KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
LEMBAGA PENGEMBANGAN DAN PEMBERDAYAAN PENDIDIK DAN TENAGA KEPENDIDIKAN
BIDANG KELAUTAN PERIKANAN TEKNOLOGI INFORMASI DAN KOMUNIKASI
(LPPPTK KPTK) GOWA
2018

RPL

REKAYASA PERANGKAT LUNAK

Mengimplementasikan Pemrograman Berorientasi Objek

Penulis:
Alun Sujjadah, S.Kom., MT.

Modul,
Pelatihan
Berbasis
Kompetensi

BUKU INFORMASI

MENGIMPLEMENTASIKAN PEMROGRAMAN BERORIENTASI OBJEK J.620100.018.02

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN R.I.
DIREKTORAT JENDERAL GURU DAN TENAGA KEPENDIDIKAN
LEMBAGA PENGEMBANGAN DAN PEMBERDAYAAN PENDIDIK DAN TENAGA KEPENDIDIKAN
BIDANG KELAUTAN, PERIKANAN,
DAN TEKNOLOGI INFORMASI DAN KOMUNIKASI
GOWA

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR LISTING

DAFTAR TABEL

BAB I

PENDAHULUAN

A. TUJUAN UMUM

Setelah mempelajari modul ini peserta diharapkan mampu membuat perangkat lunak aplikasi dalam bahasa pemrograman berorientasi objek

B. TUJUAN KHUSUS MENGIMPLEMENTASIKAN PEMROGRAMAN BERBASIS OBYEK

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Mengimplementasikan Pemrograman Berorientasi Objek ini guna memfasilitasi peserta sehingga pada akhir diklat diharapkan memiliki kemampuan sebagai berikut:

1. Membuat program berorientasi objek dengan memanfaatkan class
2. Menggunakan tipe data dan control program pada metode atau operasi dari suatu kelas
3. Membuat program dengan konsep berbasis objek
4. Membuat program object oriented dengan interface dan paket
5. Mengkompilasi program

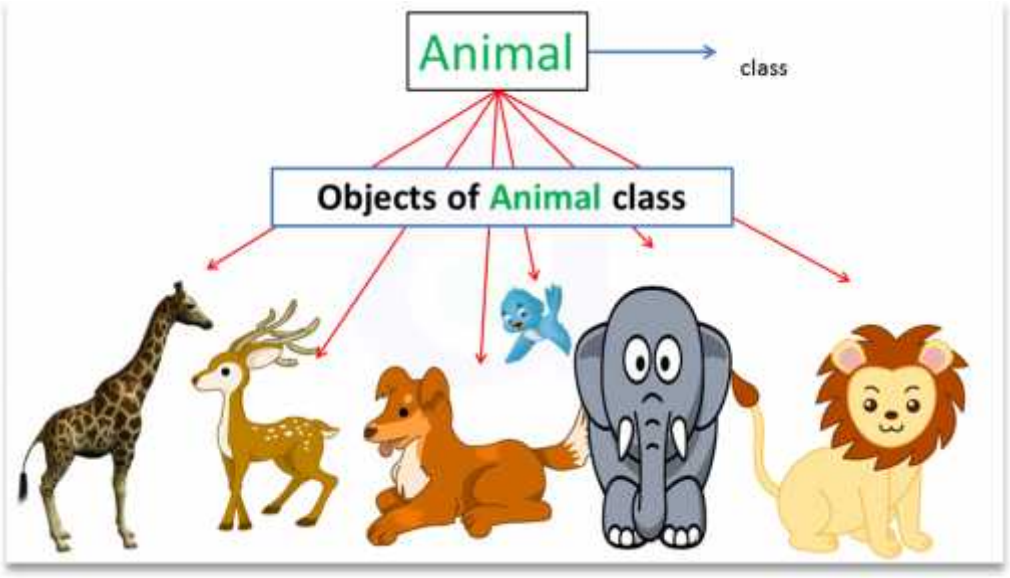
BAB I I

MEMBUAT PROGRAM BERORIENTASI OBJEK DENGAN MEMANFAATKAN CLASS

A. Pengetahuan yang Diperlukan dalam Membuat Program Berorientasi Objek dengan Memanfaatkan Class

1. Konsep Class dan Object

Paradigma pemrograman berorientasi objek adalah merepresentasikan sebuah permasalahan kedalam bentuk class dan object. Seorang programmer diupayakan untuk membuat program lebih dekat dengan memodelkan cara orang berpikir seperti dunia nyata. Class dan object adalah sebuah konsep yang saling berkaitan erat dan tidak dapat dipisahkan. Class merupakan cetak biru atau kerangka dalam pembuatan program, sedangkan object adalah hasil instance atau penciptaan dari sebuah class. Class merupakan prototipe yang mendefinisikan attribute dan behavior secara umum. Saat implementasi kedalam sebuah program attribute dimodelkan sebagai variabel dan behavior dimodelkan sebagai method. Sebagai analogi untuk mempermudah memahami konsep tentang class dan object, perhatikan contoh gambar 2.1



Gambar 2.1 Analogi class dan objek

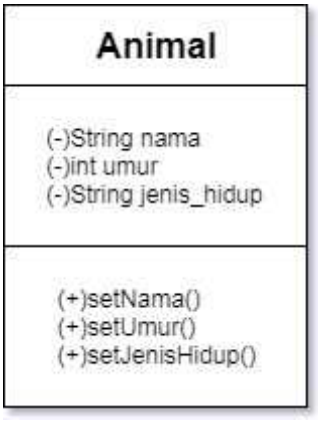
Terdapat class dengan nama Animal, dimana didalam sebuah class akan terdapat attribute (variabel) dan method (behavior) yang akan dibahas pada poin berikutnya. Pada sebuah class didefinisikan beberapa hal yang bersifat umum, sehingga akan dapat digunakan untuk penciptaan object dari class tersebut. Sebagai contoh didalam class Animal mempunyai variabel nama, umur dan jenis_hidup. Ketiga variabel tersebut digunakan karena bersifat umum, setiap Animal pasti mempunyai nama, umur dan jenis hidupnya di air, darat atau udara.

Selain itu didalam class terdapat beberapa method seperti setNama, setUmur, setJenisHidup yang berfungsi untuk memberikan nilai kepada variabel. Ketika sudah tercipta sebuah class yang merupakan kerangka atau cetak biru, maka akan dapat dibuat sebuah object seperti jerapah, kijang, anjing, burung, gajah dan singa karena objek-objek tersebut mempunyai nama, umur dan jenis hidup. Tiap-tiap object akan memiliki nilai variabel yang berbeda-beda tergantung dengan nilai yang dilewatkan pada method seperti pada contoh tabel 2.1

Tabel 2.1 Nilai atau isi dari masing-masing atribut

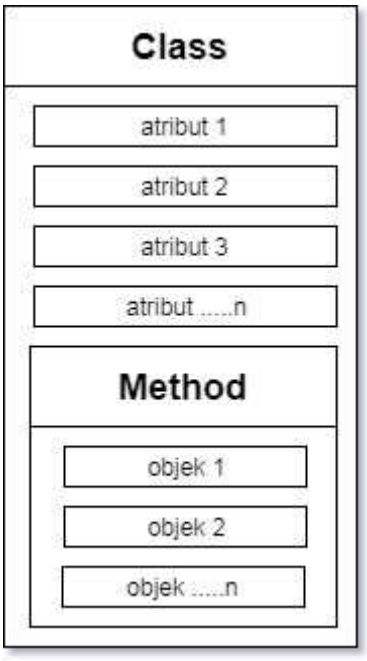
Objek	Nama	Umur	Jenis Hidup
Objek 1	Singa	10 tahun	Darat
Objek 2	Gajah	7 tahun	Darat
Objek 3	Burung	2 tahun	Udara
Objek 4	Ikan	2 tahun	Air

Pada pemrograman berorientasi objek maka setiap permasalahan akan dimodelkan dalam bentuk diagram class. Adapun contoh dari pemodelannya seperti pada gambar 2.2



Gambar 2.2 Contoh diagram class

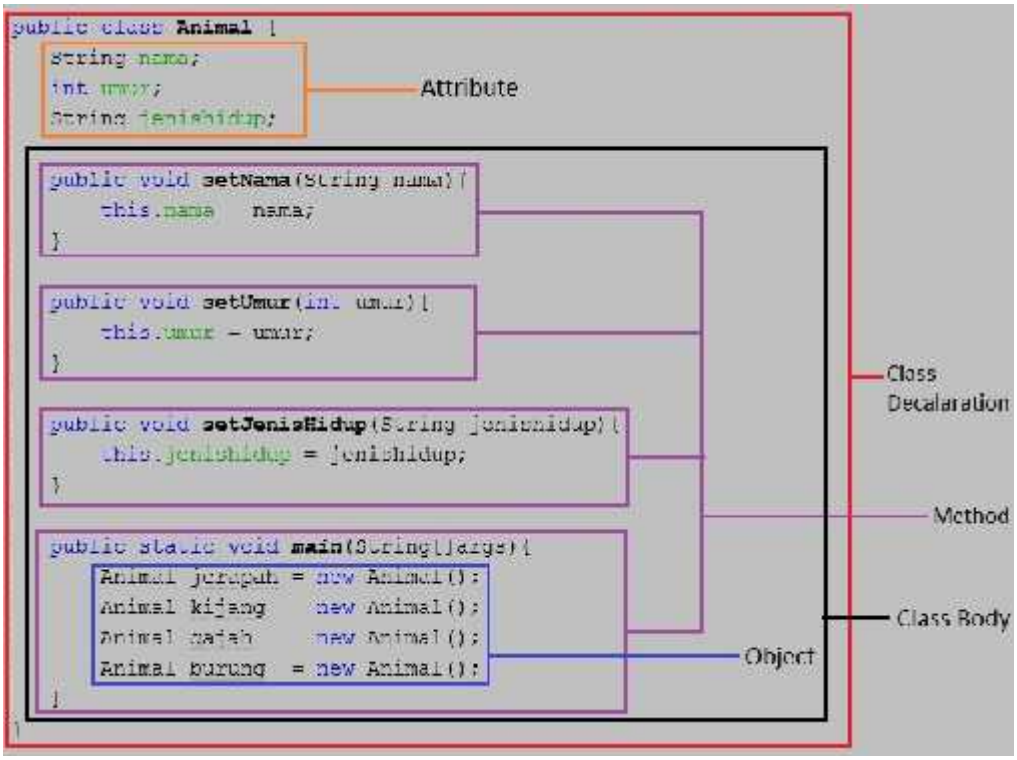
Ruang lingkup (scope) pembuatan class dan object adalah class sebagai pembungkus sebuah object dan method. Pembuatan sebuah object pada umumnya dilakukan didalam method. Untuk memudahkan pemahaman, perhatikanlah gambar 2.3



Gambar 2.3 Ruang lingkup

Deklarasi pembuatan class menggunakan kata kunci class, sedangkan penciptaan object dari sebuah class atau yang lebih dikenal dengan proses instantiate dalam sebuah program yaitu menggunakan operator new dengan sintaks yaitu nameOfClass nameOfObject =

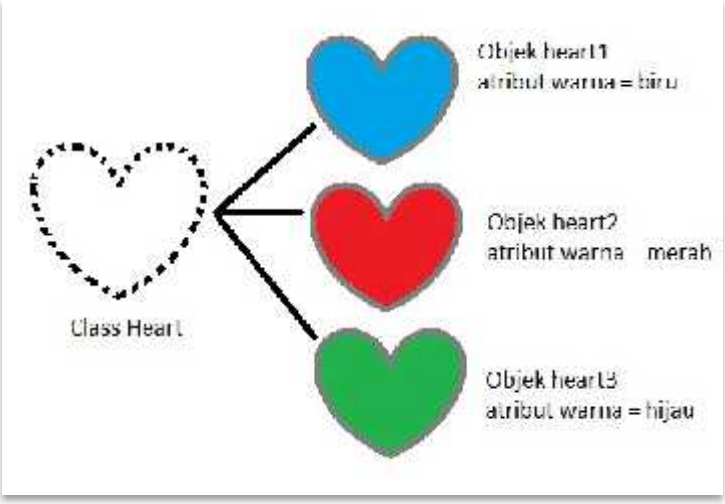
new nameOfClass(). Contoh penerapan pembuatan class dan object menggunakan bahasa pemrograman JAVA seperti pada gambar 2.4



Gambar 2.4 Pembuatan class dan object

Secara garis besar, susunan deklarasi class pada JAVA terdiri dari 2 bagian utama yaitu class declaration dan class body. Class declaration mendefinisikan nama class beserta beberapa variabelnya, sedangkan class body mendefinisikan method dan variabel yang ada didalamnya. Setiap penciptaan sebuah object, maka object tersebut akan memiliki semua variabel dan method yang dimiliki oleh sebuah class sesuai dengan visibilitas akses modifier seperti public, private, protected ataupun default yang akan dibahas pada poin selanjutnya. Berdasarkan contoh gambar 2.4 ketika diciptakan object jerapah dari class Animal dengan menggunakan kode program Animal jerapah = new Animal() maka object jerapah akan memiliki variabel nama, umur dan jenishidup. Selain itu object jerapah juga memiliki method setNama, setUmur dan setJenisHidup. Cara mengakses sebuah method yaitu dengan sintaks objek.namaMethod([parameter]), misalnya jerapah.setNama("Jerapah Asia") , begitu juga dengan objek-objek dan method lainnya. Perhatikan

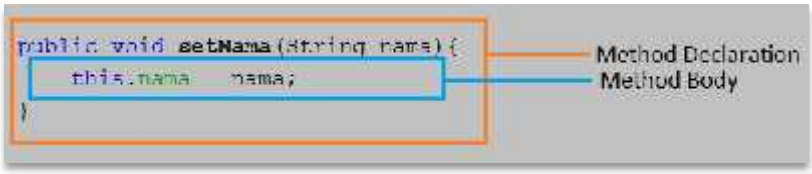
ilustrasi gambar 2.5 untuk lebih memahami konsep pemrograman berorientasi objek.



Gambar 2.5 Ilustrasi class dan objek

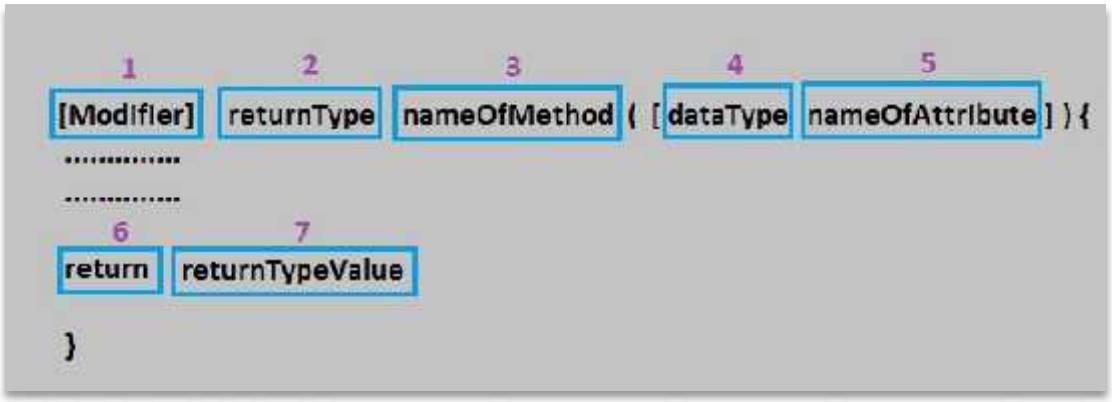
2. Method

Elemen penting yang termasuk dalam pembuatan class dan object adalah method. Pada umumnya method dapat berupa fungsi ataupun prosedur yang digunakan untuk melakukan output, pengisian ataupun mengambil nilai dari variabel. Perbedaan antara fungsi dan prosedur adalah pada sebuah fungsi yang digunakan dalam program akan membutuhkan return value (nilai balik) untuk digunakan pada proses selanjutnya, sedangkan pada prosedur tidak membutuhkannya. Dalam istilah lain method terbagi menjadi Setter dan Getter. Method Setter (prosedur) digunakan untuk memberikan nilai ataupun untuk menampilkan nilai dari variabel, sehingga tidak memerlukan return value, sedangkan method getter (fungsi) digunakan untuk mengambil nilai dari variabel, sehingga membutuhkan return value. Sama halnya dengan class, method terdiri dari 2 bagian untuk membuatnya yaitu method declaration dan method body seperti pada gambar 2.6



Gambar 2.6 Bagian method

Keyword this digunakan jika terdapat parameter yang namanya sama dengan nama variabel class. Seperti pada contoh gambar 2.4 terdapat atribut nama dengan tipe data String dan pada method setNama mempunyai parameter nama dan juga bertipe data String, sehingga untuk memberikan nilai variabel nama menggunakan keyword this, yang berarti variabel nama diberi nilai oleh parameter nama. Sintaks umum untuk pembuatan method adalah seperti pada gambar 2.7



Gambar 2.7 Sintaks penulisan method

Untuk pemahaman sintaks pembuatan method yaitu bagian yang ditulis dalam kurung siku [ ] adalah bersifat opsional, boleh digunakan maupun tidak, tergantung permasalahan yang akan diselesaikan.

Penjelasan gambar 2.7:

1. Modifier bersifat opsional, boleh digunakan atau tidak sesuai permasalahan. Modifier adalah keyword yang digunakan untuk mengatur hak akses atau tingkat visibilitas yang dapat diterapkan pada sebuah method, class, variabel ataupun object. Modifier dapat bernilai public, private, protected ataupun jika tidak digunakan berarti secara otomatis menggunakan modifier default. Pembahasan modifier akan terdapat pada poin selanjutnya.
2. returnType dapat diisikan tipe data dari return value yang dihasilkan oleh method. Jenis-jenis Tipe data tersebut adalah String (satu atau kumpulan dari beberapa karakter), int (angka bulat), float(angka pecahan), boolean(true atau false) dan lain-lain. Jika

method yang dibuat tidak menghasilkan return value, maka diisikan dengan keyword void.

3. nameOfMethod diperlukan untuk menuliskan pengenal yaitu nama dari method. Pemberian nama method mempunyai aturan yang sama dengan penamaan variabel diantaranya adalah
   a. Tidak diperbolehkan menggunakan keyword dari JAVA seperti exit, return, switch, main dan lain-lain.
   b. Diperbolehkan menggunakan angka (0-9), garis bawah (underscore), simbol dollar ($).
4. dataType berisi tipe data yang bersifat opsional, diperlukan jika sebuah method membutuhkan parameter untuk melewatkan nilai variabel seperti String, int, float, Boolean dan lain-lainnya.
5. nameOfAttribute adalah nama dari variabel yang terdapat dalam parameter dan juga bersifat opsional jika sebuah method membutuhkan parameter.
6. return adalah keyword yang digunakan untuk menghasilkan nilai balik. Dibutuhkan ketika method yang dibuat tidak menggunakan returnType void (prosedur).
7. returnTypeValue adalah nilai yang akan di jadikan nilai balik.

Listing 2.1 dan 2.2 berikut ini adalah contoh untuk penulisan method yang berupa fungsi dan prosedur

```
public void cetakData(){

    System.out.println("Ini adalah Method");
    System.out.println("Method berupa prosedur tanpa parameter");
    System.out.println("karena tidak mempunyai return value");
}
```

Listing 2.1 Method berupa prosedur tanpa parameter

```
public int tambahDuaBilangan(int bil1, int bil2){

    int hasil  = bil1 + bil2;
    return hasil;
}
```

Listing 2.2 Method berupa fungsi dengan parameter dan return value integer

3. Data Variabel

Berdasarkan scope atau ruang lingkupnya, maka data variabel dibedakan menjadi 2 jenis yaitu:

a. Member Variabel /Class Variabel

Yaitu variabel yang dimiliki oleh class, dimana deklarasinya diletakkan didalam class, sehingga variabelnya akan dapat diakses oleh semua method dan minimal didalam class yang mendefinisikannya.

b. Local Variabel

Yaitu variabel yang dimiliki oleh method, dimana deklarasinya diletakkan didalam method, sehingga variabelnya hanya dapat diakses didalam method yang mendeklarasikannya sendiri.

Untuk lebih memperjelas pengetahuan tentang Member Variabel dan Local Variabel, perhatikanlah contoh gambar 2.8 berikut ini:

```
public class Guru {
    private String nip;
    private String nama;
    private boolean gender;
    private String alamat;

    public void setTitelNamaGuruRPL (String nama ){
        this.nama = nama + ",S.Kom";
    }

    public String getTitelNamaGuruRPL(){
        return nama;
    }
    void setGajiGuru(){
        int gaji3B = 3000000;
    }
    public static void main(String[]args){
        Guru guru1 = new Guru();
        guru1.setTitelNamaGuruRPL("Helika Fidi Titimonno ");
        System.out.println(guru1.getTitelNamaGuruRPL());
        int lembur = 1000000;
        int totalgaji = gaji3B + lembur;
    }
}
```

Member Variable/Class Variable

Local Variable

Gambar 2.8 Member Variabel dan Local Variabel

Pada class Guru terdapat member variabel nip, nama, gender dan alamat dimana ruang lingkupnya dapat diakses disemua method, sehingga ketika variabel nama diakses oleh method setTitelNamaGuruRPL, maka kode program akan berjalan dengan benar. Sedangkan pada local variabel gaji3B yang didefinisikan didalam method setGajiGuru, maka variabel tersebut tidak dapat diakses didalam method lain, terbukti ketika diakses didalam method main muncul garis merah yang menandakan kode program dalam keadaan error.

4. Modifier

Dalam pemrograman berorientasi objek dikenal istilah modifier, yaitu sebuah keywords yang digunakan untuk mengatur hak akses sebuah variabel, method ataupun class dengan tujuan untuk menjaga integritas data ketika akan diakses oleh object lain. Sebagai analogi bentuk nyata adalah ketika kita mempunyai sebuah barang, maka ada beberapa barang yang hanya boleh kita gunakan sendiri dan tidak boleh dipinjamkan kepada orang lain misalnya sikat gigi, berarti kalau diimplementasikan kedalam bahasa pemrograman JAVA maka modifier yang digunakan adalah private. Selain itu ada barang-barang yang boleh dipinjamkan kepada orang lain misalnya buku, berarti implementasinya menggunakan modifier public. Atau barang-barang yang kita punyai hanya dapat dipinjam oleh saudara-saudara dirumah kita sendiri, maka disini menggunakan modifier protected.

Modifier terbagi menjadi 2 yaitu

a. Access Modifier

Terdapat 4 macam access modifier yaitu

i. public yang berarti class, method maupun atribut yang mempunyai access modifier public memungkinkan untuk diakses dari manapun dan oleh kelas apapun. Access modifier public mempunyai hak akses paling luas dibanding

yang lainnya. Penggunaan modifier public seperti pada listing 2.3 berikut ini

```
public class Mobil {

    public String merk;
    public int tahun_pembuatan;

    public void setDataMobil(String merk, int tahun_pembuatan){
        this.merk = merk;
        this.tahun_pembuatan = tahun_pembuatan;
    }
    public String getMerk(){
        return merk;
    }

    public int getTahun(){
        return tahun_pembuatan;
    }
    public static void main(String[]args){
        Mobil mobilku = new Mobil();
        mobilku.setDataMobil("AVANZA",2000);
        System.out.println("Merk Mobil : " + mobilku.getMerk());
        System.out.println("Tahun Mobil :" + mobil_ku.getTahun());
    }
}
```

Listing 2.3 Penggunaan modifier public

ii. private yang berarti class, method maupun variabel hanya dapat diakses secara langsung didalam class itu sendiri. Modifier private mempunyai akses tertutup, sesuai dengan konsep pemrograman berorientasi objek, setiap data harus disembunyikan. Untuk mengaksesnya diperlukan method setter dan getter. Penggunaan modifier private seperti pada listing 2.4 berikut ini

```
class Kendaraan{
    private String merk;
    private int tahun_pembuatan;

    void setDataMobil(String merk, int tahun_pembuatan){
        this.merk = merk;
        this.tahun_pembuatan = tahun_pembuatan;
    }
    String getMerk(){
        return merk;
    }
    int getTahun(){
        return tahun_pembuatan;
    }
}

public class Mobil {
    public static void main(String[]args){
        Kendaraan mobilku = new Kendaraan();
        mobilku.setDataMobil("PAJERO",2017);
        System.out.println("Merk Mobil : " + mobilku.getMerk());
        System.out.println("Tahun Mobil :" + mobilku.getTahun());
    }
}
```

Listing 2.4 Penggunaan dan pengaksesan modifier private

iii. protected yang berarti bahwa class, method maupun variabel tersebut dapat diakses oleh kelas yang sama, package yang sama, dan kelas turunannya (subclass). Access modifier protected pada umumnya digunakan untuk mewariskan variabel yang ada di super class terhadap child class. Penggunaan modifier protected seperti pada listing 2.5 berikut ini

```
class Pelajar{
    protected String nis,nama;
    protected double nilai1,nilai2,nilai3;
    double rata;
    protected void setDataSiswa(String nis, String nama_siswa){
        this.nis = nis;
        nama     = nama_siswa;
    }
    public void  setNilaiSiswa(double n1,double n2, double n3){
        nilai1 = n1;
        nilai2 = n2;
        nilai3 = n3;
    }
    public double getNilaiRata(){
        rata = (nilai1 + nilai2 + nilai3)/3;
        return rata;
    }
}

public class Siswa {
    public static void  main(String[]args){
        Pelajar siswa1 = new Pelajar();
        siswa1.setDataSiswa("1102020","Andi");
        siswa1.setNilaiSiswa(90, 85, 200);
        System.out.println(siswa1.getNilaiRata());
    }
}
```

Listing 2.5 Penggunaan modifier protected

iv. default atau disebut juga no access modifier yaitu class, method dan variabel dapat diakses disemua class manapun asal masih dalam satu package yang sama. Penulisannya tanpa munggunakan keyword.

b. Non-Access Modifier

Java menyediakan beberapa macam Non-Access Modifier, yaitu:

i. static

Static adalah salah satu jenis modifier di Java yang digunakan agar suatu atribut atau pun method dapat diakses oleh kelas atau objek tanpa harus melakukan instansiasi terhadap kelas tersebut. Method main adalah salah satu contoh method yang mempunyai modifier static.

ii. final

Final adalah salah satu modifier yang digunakan agar suatu atribut atau method bersifat final atau tidak bisa diubah nilainya. Modifier ini digunakan untuk membuat konstanta di Java.

iii. abstract

Abstract adalah modifier yang digunakan untuk membuat kelas dan method abstrak

iv. synchronized

Synchronized adalah modifier yang digunakan dalam aplikasi Java berbasis thread. Modifier ini menspesifikasikan bahwa method merupakan thread safe. Artinya bahwa hanya ada satu jalur eksekusi pada method yang menggunakan modifier jenis ini dan memaksa thread thread lain menunggu giliran.

v. native

Modifier Native digunakan untuk spesifikasi method dengan implementasi di bahasa lain, seperti C, C++.

vi. transient

Modifier ini digunakan agar suatu variabel tidak bisa di serialisasi. Serialization adalah konsep dimana sebuah objek dapat ditransfer dari suatu aplikasi ke aplikasi lainnya atau dari suatu workstation ke workstation lainnya. Konsep ini sangat diperlukan ketika membuat aplikasi client server. Salah satu tujuan serialization adalah bahwa tidak boleh ada perubahan terhadap atribut pada saat objek di transformasikan menjadi stream.

vii. volatile

viii. implements

Digunakan untuk mengimplementasikan sebuah interface.

ix. Extends

Digunakan untuk proses pewarisan class.

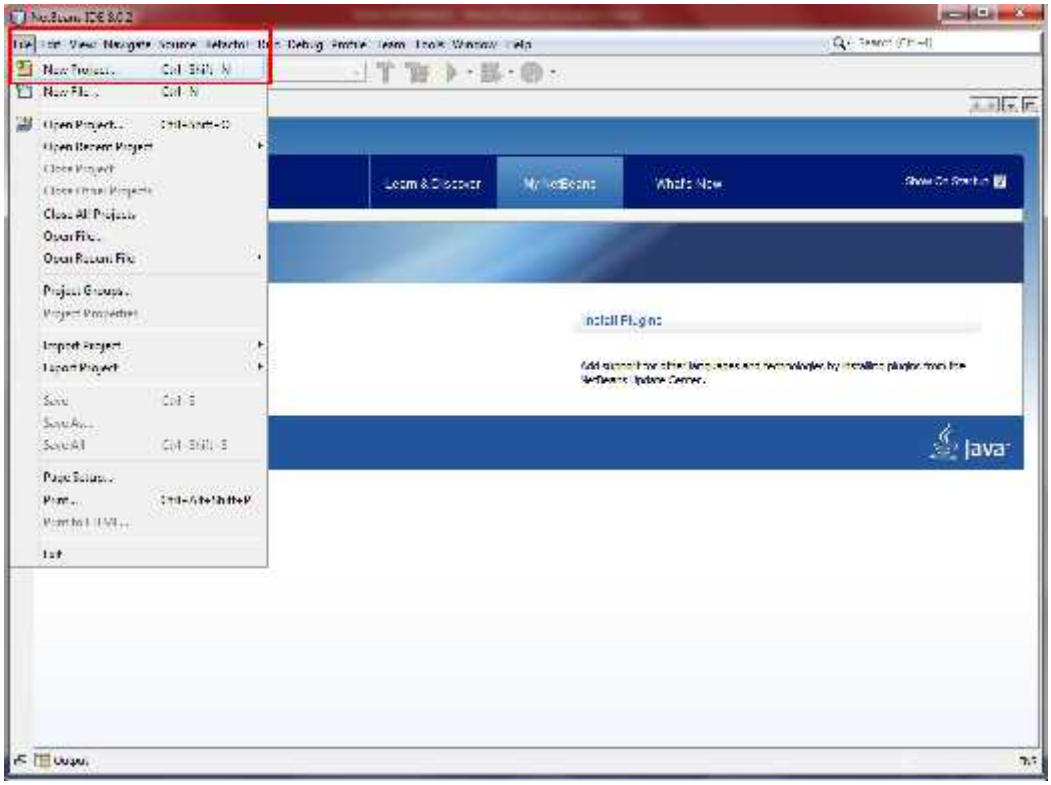
Tabel 2.2 berikut ini menyatakan scope akses modifier

Access Modifier	Same Class	Same Package	Sub Class	Other Package
public	√	√	√	√
protected	√	√	√	X
default	√	√	X	X
private	√	X	X	X

5. Implementasi Pembuatan Class dan Object

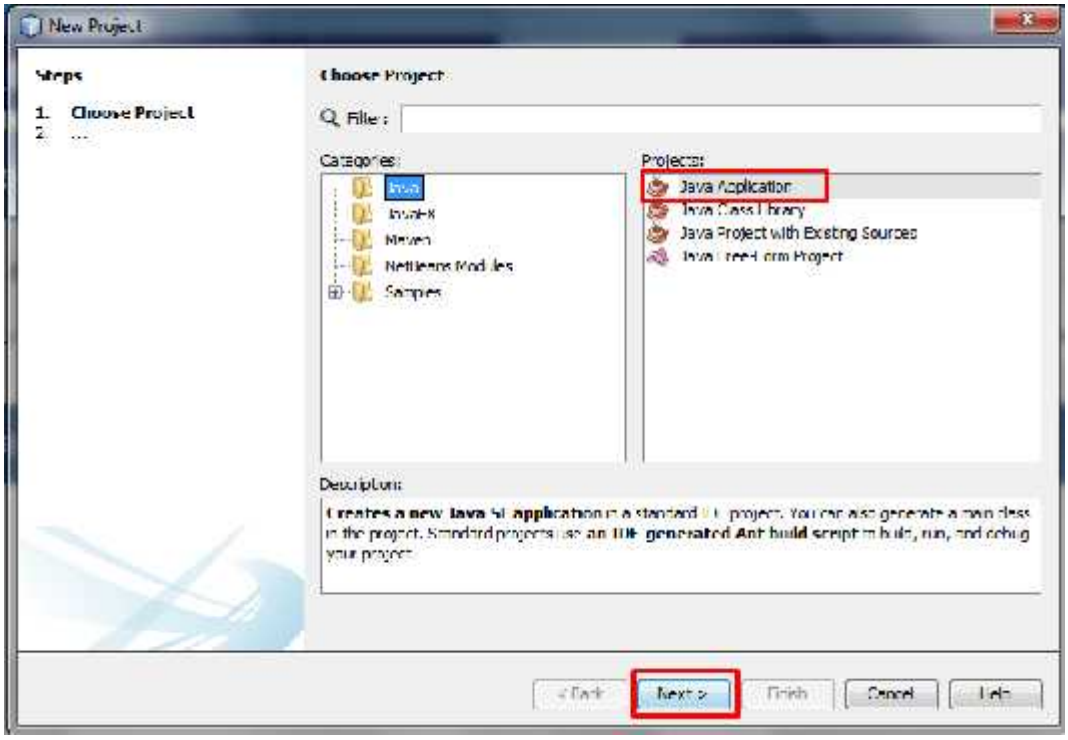
Pembuatan class dan object pada pembahasan ini adalah menggunakan bahasa pemrograman JAVA dengan menggunakan software Netbeans. Sebagai contoh kasus permasalahan adalah program untuk melakukan operasi aritmatika penjumlahan, pengurangan, perkalian dan pembagian antara 2(dua) bilangan. Langkah-langkahnya adalah sebagai berikut:

1. Bukalah editor software Netbeans yang telah terinstall di komputer. Kemudian lanjutkan dengan melakukan klik File → New Project seperti pada gambar 2.9



Gambar 2.9 Tampilan IDE Netbeans

2. Jika berhasil maka akan muncul tampilan dialog untuk menentukan jenis project seperti pada gambar 2.10, lanjutkan dengan memilih Java Application dan klik Next



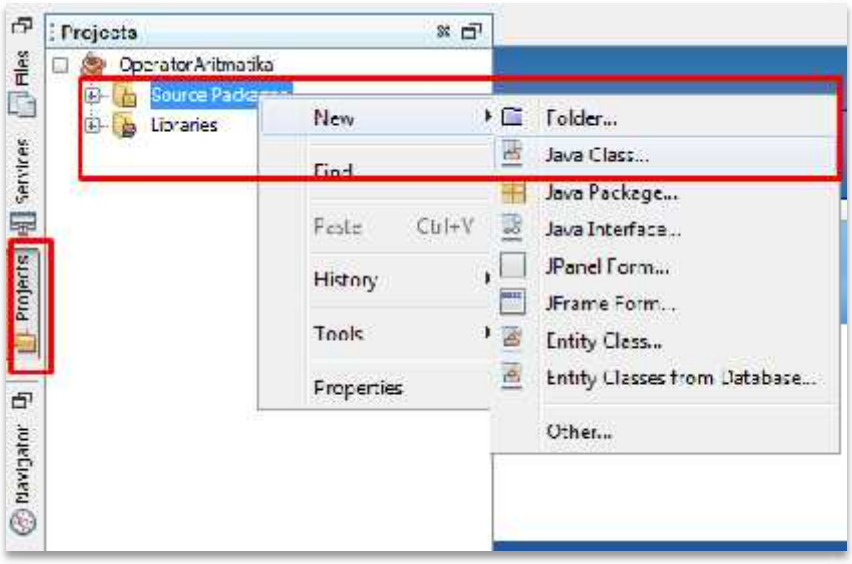
Gambar 2.10 Penentuan jenis project

3. Isikan parameter yang dibutuhkan dalam project seperti pada gambar 2.11

Gambar 2.11 Pengisian parameter project

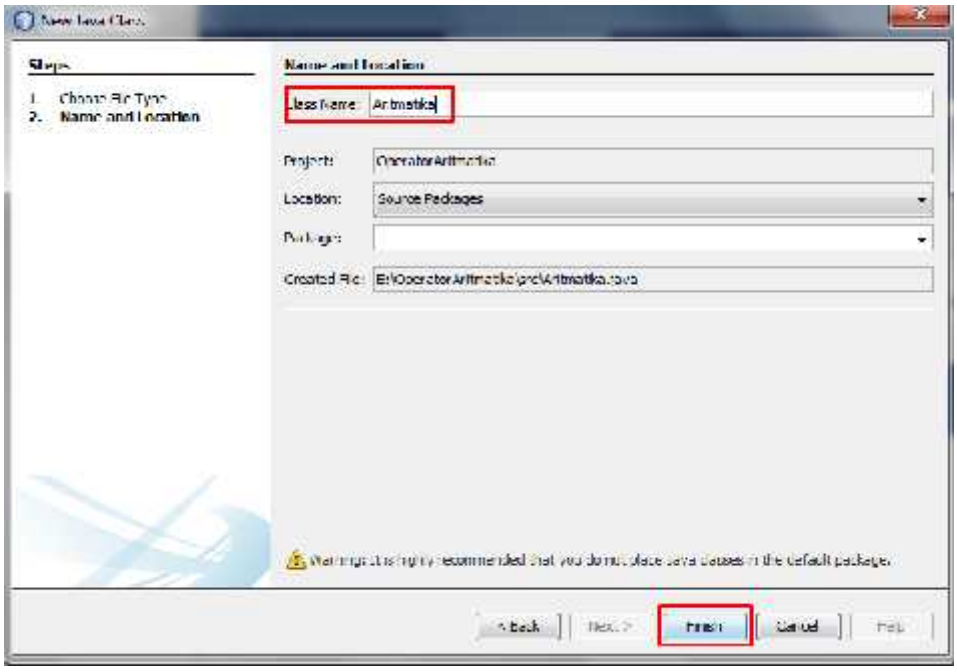
Pada contoh tersebut field Project Name diisikan OperatorAritmatika, kemudian tempat penyimpanan file di drive E:\ sesuaikan dengan komputer masing-masing, dan hilangkan centang pada field Create Main Class.

4. Klik pada tab projects, kemudian klik kanan pada bagian source packages dan pilih New → Java Class seperti pada gambar 2.12



Gambar 2.12 Pembuatan class

5. Isikan class name yang akan dibuat, misal nama class Aritmatika, kemudian lanjutkan dengan klik next seperti pada gambar 2.13



Gambar 2.13 Pengisian Class Name

6. Setelah berhasil membuat Java Class, maka akan otomatis terbentuk sebuah class dengan nama Aritmatika, kemudian lanjutkan dengan menambahkan Member Variabel bilangan1, bilangan2 dan hasil dengan tipe data float menggunakan access modifier private seperti pada listing 2.6

```
public class Aritmatika {
    private float bilangan1;
    private float bilangan2;
    private float hasil;
}
```

Listing 2.6 Deklarasi Member Variabel

7. Tambahkan Setter Method setDataBilangan dengan 2(dua) parameter bilangan1 dan bilangan2, yang digunakan untuk memberikan nilai pada 2(dua) bilangan seperti pada listing 2.7

```
public class Aritmatika {
    private float bilangan1;
    private float bilangan2;
    private float hasil;

    public void setDataBilangan(int bilangan1, int bilangan2){
        this.bilangan1 = bilangan1;
        this.bilangan2 = bilangan2;
    }
}
```

Listing 2.7 Pembuatan Setter Method

8. Tambahkan Getter Method getHasilJumlah, getHasilKurang, getHasilKali dan getHasilBagi dengan return type value adalah float seperti pada listing 2.8

```
public class Aritmatika {
    private float bilangan1;
    private float bilangan2;
    private float hasil;

    public void setDataBilangan(int bilangan1, int bilangan2){
        this.bilangan1 = bilangan1;
        this.bilangan2 = bilangan2;
    }

    public float getHasilJumlah(){
        hasil =  bilangan1+bilangan2;
        return hasil;
    }
    public float getHasilKurang(){
        hasil = bilangan1 - bilangan2;
        return hasil;
    }
    public float getHasilKali(){
        hasil = bilangan1 * bilangan2;
        return hasil;
    }
    public float getHasilBagi(){
        hasil =  bilangan1 / bilangan2;
        return hasil;
    }
}
```

Listing 2.8 Getter Method

9. Tambahkan main method dimana method tersebut akan dijalankan pertama kali oleh compiler program, kemudian didalamnya tambahkan proses pembuatan object dari dari class Aritmatika, dan proses terakhir adalah object yang telah dibuat mengakses semua method yang terdapat dalam class, seperti pada listing 2.9

```
public static void main(String []args){
    Aritmatika arit1 = new Aritmatika();
    arit1.setDataBilangan(45,6);
    System.out.println(arit1.getHasilJumlah());
    System.out.println(arit1.getHasilKurang());
    System.out.println(arit1.getHasilKali());
    System.out.println(arit1.getHasilBagi());
}
```

Listing 2.9 Pembuatan object didalam main method

10. Jalankan program dengan menekan tombol Shift + F6

B. Keterampilan yang diperlukan dalam Membuat Program Berorientasi Objek dengan Memanfaatkan Class

1. Mengoperasikan software IDE (Integrated Development Environment) untuk bahasa pemrograman JAVA seperti Netbeans, GEL, Eclipse dan lain-lain
2. Menyusun rancangan class, method dan variabel yang digunakan dalam program.
3. Membuat method main, method setter dan getter, variabel beserta tipe data yang digunakan.
4. Menentukan access dan non access modifier
5. Melakukan proses instantiate objek dari sebuah class.
6. Memanggil method dari sebuah object atau lebih.

C. Sikap kerja yang diperlukan dalam Membuat Program Berorientasi Objek dengan Memanfaatkan Class

Harus bersikap secara:

1. Cermat dan teliti dalam menganalisis data.
2. Tekun dalam proses pemahaman sintaks.
3. Sesuai dengan kaidah-kaidah bahasa pemrograman.
4. Berpikir analitis serta evaluatif ketika melakukan trial and error.

BAB III

Menggunakan Tipe Data dan Control Program Pada Metode Atau Operasi Dari Suatu Kelas

A. Pengetahuan yang diperlukan dalam Menggunakan Tipe Data dan Control Program pada Metode atau Operasi dari Suatu Kelas

1. Variabel, Tipe Data dan konstanta

Variabel merupakan container yang digunakan untuk menyimpan suatu nilai secara sementara pada sebuah program dengan tipe tertentu. Sementara berarti nilainya dapat diubah sesuai keperluan, dan ketika program dihentikan maka nilai dari variabel akan hilang. Pada dasarnya ada dua macam tipe data variabel yaitu tipe primitive dan tipe reference.

Tipe primitive meliputi:

a. Tipe Boolean adalah tipe data logika yang hanya mempunyai 2 nilai yaitu true dan false.
b. Tipe Numerik yang meliputi:
    i. byte, adalah tipe data Integral 8-bit. Memiliki rentang nilai antara -2^7 sampai $2^7 – 1$ atau dari -128 sampai 127
    ii. short, adalah tipe data Integral 16-bit. Memiliki rentang nilai antara -2^{15} sampai $2^{15} – 1$ atau dari -32768 sampai 32768.
    iii. int, adalah tipe data Integral 32-bit. Memiliki rentang nilai antara -2^{31} sampai $2^{31} –1$ atau dari -2,147,483,648 sampai 2,147,483,647.
    iv. long, adalah tipe data Integral 64-bit. Memiliki rentang nilai antara -2^{63} sampai $2^{63} – 1$ atau dari -9,223,372,036,854,775,808 sampai 9,223,372,036,854,775,807.

v. char, adalah tipe data Textual, yang merepresentasikan karakter unicode 16-bit. Nilai literalnya harus diapit dengan tanda petik tunggal (‘).

vi. float, adalah tipe data Floating Point 32-bit. Nilai literalnya mengandung pecahan (dipisahkan dengan tanda titik ‘.’)

vii. double, adalah tipe data Floating Point 64-bit. Nilai literal default untuk float dan double adalah double, kecuali diberi akhiran F

Sedangkan tipe data variabel berupa reference terdiri atas tipe variabel data:

a. Tipe class

b. Tipe array

c. Tipe interface

Untuk mendeklarasikan variabel dan tipe data menggunakan sintaks `<tipe-data> <nama variabel>`. Perhatikan contoh listing 3.1 berikut ini untuk memahami penggunaan tipe data dan variabel.

```
public class Variabel {

    public static void main(String[]args){
        byte   nilai_byte   = 70;
        short  nilai_short  = 500;
        int    nilai_int    = 263456;
        long   nilai_long   = 1200000;
        char   nilai_char   = 'A';
        float  nilai_float  = 190;
        double nilai_double = 387;
        boolean nilai_bool  = true;

        System.out.println(nilai_byte);
        System.out.println(nilai_short);
        System.out.println(nilai_int);
        System.out.println(nilai_long);
        System.out.println(nilai_char);
        System.out.println(nilai_float);
        System.out.println(nilai_double);
        System.out.println(nilai_bool);
    }
}
```

Listing 3.1 Penggunaan tipe data dan variabel

2. Control Program

Struktur control program terdiri dari 2(dua) jenis, yaitu struktur percabangan dan perulangan. Percabangan digunakan untuk memilih dan mengeksekusi blok kode ketika ada 2(dua) atau lebih kemungkinan kondisi. Kondisi diisikan dengan sebuah pernyataan yang melibatkan operator logika ==(sama dengan), !=(tidak sama dengan), !(negasi), >(lebih besar), >=(lebih besar sama dengan), <(lebih kecil), <=(lebih kecil sama dengan). Sedangkan perulangan digunakan untuk mengeksekusi program secara berulang-ulang.

2.1. Struktur percabangan

a. Pernyataan IF satu kondisi. Sintaks penggunaanya adalah

```
if (kondisi){
   pernyataan
}
```

Jika kondisi bernilai true maka program akan mengeksekusi pernyataan.

b. Pernyataan IF dua kondisi. Sintaks penggunaanya adalah

```
if (kondisi){
   pernyataan
}
else{

}
```

Jika kondisi bernilai true maka program akan mengeksekusi pernyataan, jika bernilai false maka program akan mengeksekusi bagian blok else.

c. Pernyataan IF lebih dari 2 kondisi. Sintaks penggunaanya adalah

```
if (kondisi 1){
   pernyataan 1
}
else if(kondisi 2){
   Pernyataan 2
}
else{
   Pernyataan 3
}
```

Jika kondisi 1 bernilai true, maka program akan mengeksekusi pernyataan 1, jika bernilai false, maka program akan beralih pada bagian blok else untuk selanjutnya melakukan pengecekan pada kondisi 2. Jika kondisi 2 bernilai true maka program akan mengeksekusi pernyataan 2, jika bernilai false maka program akan mengeksekusi pernyataan 3.

d. Pernyataan SWITCH, sintaks penggunaanya adalah

```
switch(kondisi){
      case kondisi_1 :
                    pernyataan 1;
                    break;
      case kondisi_2 :
                    pernyataan 2;
                    break;
      case kondisi_n :
                    pernyataan n;
                    break;
}
```

Program akan melakukan pengecekan nilai kondisi, kemudian program akan menyesuaikan nilai tersebut termasuk dalam case sesuai kondisi masing-masing, untuk mengerjakan baris pernyataan sampai break

Contoh penggunaan struktur percabangan IF seperti pada listing 3.2

```
public class Student {
    private String nama_siswa;
    private int nilai_siswa;

    public void setDataStudent(String nama_siswa, int nilai){
        this.nama_siswa = nama_siswa;
        nilai_siswa = nilai;
    }
    public String getNilaiHuruf(){
        String nilai_huruf = null;
        if(nilai_siswa >= 80){
            nilai_huruf = "Memuaskan";
        }
        else if(nilai_siswa >=70){
            nilai_huruf = "Baik";
        }
        else if(nilai_siswa >=60){
            nilai_huruf = "Kurang";
        }

        return nilai_huruf;
    }

    public static void main(String args[]){
        Student siswa1 = new Student();
        siswa1.setDataStudent("Budi",90);
        System.out.println(siswa1.getNilaiHuruf());
    }
}
```

Listing 3.2 Struktur percabangan menggunakan IF

Contoh penggunaan struktur percabangan menggunakan switch seperti pada listing 3.3

```
public class Student {
    private String nama_siswa;
    private String gender;

    public void setDataStudent(String nama_siswa, String gender){
        this.nama_siswa = nama_siswa;
        this.gender = gender;
    }
    public String getGenderInEnglish(){
        String jk = null;
        switch(gender){
            case "L" :
                jk = "MALE";
                break;
            case "P" :
                jk = "FEMALE";
                break;
        }

        return jk;
    }

    public static void main(String args[]){
        Student siswa1 = new Student();
        siswa1.setDataStudent("Budi","P");
        System.out.println(siswa1.getGenderInEnglish());
    }
}
```

Listing 3.3 Struktur percabangan menggunakan switch

2.2. Struktur perulangan

a. Perulangan for, digunakan ketika permasalahan telah diketahui batas awal dan batas akhirnya. Sintaks penggunaannya adalah

```
for(<nilai awal>;<nilai akhir>;increment/decrement){
      pernyataan
}
```

b. Perulangan do while, akan dikerjakan selama kondisinya memenuhi persyaratan, minimal pernyataan akan dikerjakan 1x meskipun tidak memenuhi persyaratan. Sintaksnya adalah

```
do {
     pernyataan
} while(kondisi);
```

c. Perulangan While, akan dikerjakan selama kondisinya memenuhi persyaratan, minimal dikerjakan 0x. Sintaks penggunaannya adalah

```
while(kondisi){
      pernyataan
}
```

Contoh penggunaan perulangan seperti pada listing 3.4 berikut ini

```
public class Perulangan {

    public void setPerulanganFor(){
        for(int i=0;i<10;i++){
            System.out.println("Nilai i adalah " + i);
        }
    }
    public void setPerulanganDoWhile(){
        int a = 0;
        do{
            System.out.println("Nilai a adalah" + a);
            a = a + 1;
        }while(a < 10);
    }

    public void setPerulanganWhile(){
        int c = 0;
        while(c<20){
            System.out.println("Nilai c adalah " + c);
        }
    }
    public static void main(String[]args){
        Perulangan p1 = new Perulangan();
        p1.setPerulanganFor();
        p1.setPerulanganDoWhile();
        p1.setPerulanganWhile();
    }
}
```

Listing 3.4 Contoh perulangan

B. Keterampilan yang diperlukan dalam Menggunakan Tipe Data dan Control Program pada Metode atau Operasi dari Suatu Kelas

1 Mengidentifikasi permasalahan yang akan dijadikan control program

2 Mengidentifikasi tipe data beserta variabel yang digunakan

3 Menyusun percabangan dan perulangan pada sebuah method

C. Sikap kerja yang diperlukan dalam Menggunakan Tipe Data dan Control Program pada Metode atau Operasi Suatu Kelas.

Harus bersikap secara:

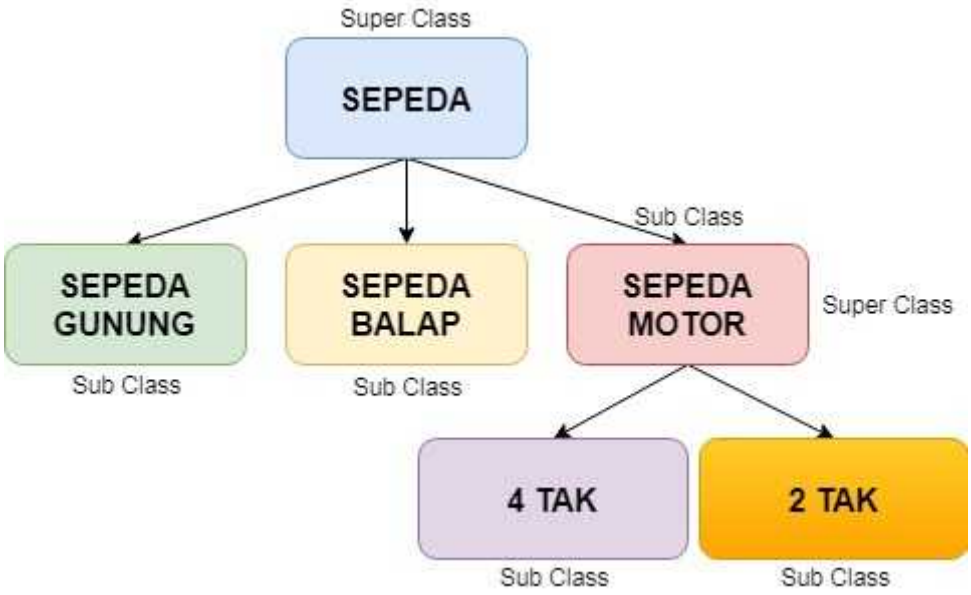
1. Cermat dan teliti dalam mengidentifikasi jenis-jenis tipe data
2. Sering melakukan percobaan dan analitis terhadap hasil program.
3. Tekun dan mau berusaha untuk mencoba berkali-kali

BAB IV
MEMBUAT PROGRAM DENGAN KONSEP BERBASIS OBJEK

A. Pengetahuan yang diperlukan dalam Membuat Program Dengan Konsep Berbasis Objek

1. Inheritance

Proses pewarisan merupakan pendefinisian sebuah class menggunakan referensi class lain yang telah terdefinisikan terlebih dahulu, dengan kata lain pewarisan adalah proses untuk menurunkan sifat dari semua variabel dan method yang dimiliki oleh Parent Class (Super Class) kepada Child Class (Sub Class). Pada konsep pemrograman berbasis objek dikenal istilah hierarki class, yaitu setiap Super Class menjadi parent dari Sub Class, dan setiap Sub Class menjadi Super Class bagi Sub Class lainnya. Pada contoh kasus class sepeda, perhatikan gambar 4.1 berikut ini



Gambar 4.1 Contoh Hierarki Class Sepeda

Pada gambar 4.1 tersebut dimungkinkan penciptaan class baru menggunakan referensi class Sepeda (Super Class), seperti class Sepeda Gunung, Sepeda Balap dan Sepeda Motor (Sub Class). Sub

Class akan mewarisi sifat-sifat dari atribut(variabel) dan behavior (method) dari Super Class, pewarisan itu misalnya class Sepeda memiliki atribut roda, maka Sepeda Balap, Sepeda Gunung dan Sepeda Motor akan memiliki atribut roda juga. Setiap Sub Class dapat menambahkan atribut dan method lain yang lebih spesifik sebagai bentuk perluasan dari Super Class, sehingga untuk membuat class pewarisan menggunakan keyword extends dengan sintaks <[modifier]> <class> <Nama Class> extends <Nama Super Class>.

Keuntungan dari konsep pewarisan adalah

a. Sub Class tidak perlu menuliskan fungsi yang berulang-ulang jika bentuk fungsi nya mempunyai kesamaan dengan class lain, cukup dengan menggunakan method yang terdapat pada Super Class sehingga program akan lebih efisien.
b. Struktur program akan lebih tertata rapi, sehingga mudah untuk dilakukan perawatan. Misal jika terdapat penambahan atau perubahan fitur program, maka cukup dilakukan modifikasi pada Super Class.

Listing 4.1, 4.2, 4.3, 4.4, 4.5 dan 4.6 berikut ini adalah implementasi dari konsep pewarisan class Sepeda, Sepeda Gunung, Sepeda Balap dan Sepeda Motor dengan menggunakan file yang terpisah satu sama lain.

```
/*
Nama File : Sepeda.java
Berfungsi sebagai Super Class
 */
public class Sepeda {
    protected String jenis_roda;
    protected String merk;
    protected int tahun_pembuatan;

    public void setDataSepeda(String jroda,String merk, int tahun){
        jenis_roda = jroda;
        this.merk = merk;
        tahun_pembuatan = tahun;
    }

    public String getJenisRoda(){
        return jenis_roda;
    }

    public String getMerk(){
        return merk;
    }

    public int getTahunBuat(){
        return tahun_pembuatan;
    }
}
```

Listing 4.1 Class Sepeda

```
/*
Nama File : SepedaGunung.java
Sub Class dari Class Sepeda
 */
public class SepedaGunung extends Sepeda{

    void cetakKeterangan(){
        System.out.println("Ini Adalah Sepeda Gunung");
    }
    public static void main(String []args){
        SepedaGunung s1 = new SepedaGunung();
        s1.cetakKeterangan();
        s1.setDataSepeda("Roda Medium","Polygon", 2015);
        System.out.println("Jenis Roda : " + s1.getJenisRoda());
        System.out.println("Merk Sepeda : " + s1.getMerk());
        System.out.println("Tahun Pembuatan : " + s1.getTahunBuat());
    }

}
```

Listing 4.2 Class SepedaGunung

```
/*
Nama File : SepedaBalap.java
Sub Class dari Class Sepeda
 */
public class SepedaBalap extends Sepeda{
    void cetakKeterangan(){
        System.out.println("Ini Adalah Sepeda Balap");
    }
    public static void main(String []args){
        SepedaBalap s1 = new SepedaBalap();
        s1.cetakKeterangan();
        s1.setDataSepeda("Roda Soft","BMX", 2000);
        System.out.println("Jenis Roda : " + s1.getJenisRoda());
        System.out.println("Merk Sepeda : " + s1.getMerk());
        System.out.println("Tahun Pembuatan : " + s1.getTahunBuat());
    }
}
```

Listing 4.3 Class SepedaBalap

```
/*
Nama File : SepedaMotor.java
Sub Class dari Class Sepeda
 */
public class SepedaMotor extends Sepeda{
    protected String mesin;

    public void setMesin(String mesin){
        this.mesin = mesin;
    }
    public String getMesin(){
        return mesin;
    }
    void cetakKeterangan(){
        System.out.println("Ini Adalah Sepeda Motor");
    }
    public static void main(String []args){
        SepedaMotor s1 = new SepedaMotor();
        s1.cetakKeterangan();
        s1.setDataSepeda("Roda Hard","Honda", 2014);
        System.out.println("Jenis Roda : " + s1.getJenisRoda());
        System.out.println("Merk Sepeda : " + s1.getMerk());
        System.out.println("Tahun Pembuatan : " + s1.getTahunBuat());
    }
}
```

Listing 4.4 Class SepedaMotor

```
/*
Nama File : Sepeda2Tak.java
Sub Class dari Class SepedaMotor
 */
public class Sepeda2Tak extends SepedaMotor{

    @Override
    void cetakKeterangan(){
        System.out.println("Ini Adalah Sepeda Motor 2 Tak");
    }
    public static void main(String[]args){
        Sepeda2Tak s2tak = new Sepeda2Tak();
        s2tak.cetakKeterangan();
        s2tak.setMesin("150 CC");
        s2tak.setDataSepeda("Roda Hard","Honda Rear", 2013);
        System.out.println("Kapasitas Mesin : " + s2tak.getMesin());
        System.out.println("Jenis Roda : " + s2tak.getJenisRoda());
        System.out.println("Merk Sepeda : " + s2tak.getMerk());
        System.out.println("Tahun Pembuatan : " + s2tak.getTahunBuat());
    }

}
```

Listing 4.5 Class Sepeda2Tak

```
/*
Nama File : Sepeda4Tak.java
Sub Class dari Class SepedaMotor
 */
public class Sepeda4Tak extends SepedaMotor{

    @Override
    void cetakKeterangan(){
        System.out.println("Ini Adalah Sepeda Motor 4 Tak");
    }
    public static void main(String[]args){
        Sepeda4Tak s4tak = new Sepeda4Tak();
        s4tak.cetakKeterangan();
        s4tak.setMesin("500 CC");
        s4tak.setDataSepeda("Roda Hard","Yamaha Vixion", 2013);
        System.out.println("Kapasitas Mesin : " + s4tak.getMesin());
        System.out.println("Jenis Roda : " + s4tak.getJenisRoda());
        System.out.println("Merk Sepeda : " + s4tak.getMerk());
        System.out.println("Tahun Pembuatan : " + s4tak.getTahunBuat());
    }

}
```

Listing 4.6 Class Sepeda4Tak

2. Constructor

Constructor adalah sebuah method yang digunakan untuk memberikan nilai default atau inisialisasi nilai pada sebuah objek ketika diciptakan. Dalam sebuah class diperbolehkan terdapat satu ataupun lebih Constructor. Sebagai ilustrasi, misalnya pembuatan objek enemy pada sebuah game, maka objek tersebut akan memiliki nilai default kekuatannya, kemampuannya seperti apa, kemudian seiiring berjalannya waktu pada game tersebut kekuatan enemy dapat bertambah ataupun berkurang. Constructor hanya dipanggil satu kali yaitu saat penciptaan sebuah objek. Pada sub class yang memanggil constructor dari super class nya dapat menggunakan keyword super, artinya sub class memanggil constructor dari super class nya. Ciri-ciri sebuah method yaitu nama method nya sama dengan nama class, tidak memiliki return value dan tidak menggunakan keyword void. Listing 4.7 berikut ini adalah contoh implementasi constructor method.

```
/**
 * Nama File : Pesawat.java
 * @author javajawara
 */
public class Pesawat {

    String jenis_pesawat;
    String nama_airlines;

    /* Constructor Method tanpa parameter */
    public Pesawat(){

    }
    public Pesawat(String jenis_pesawat, String nama_airlines){
        this.jenis_pesawat = jenis_pesawat;
        this.nama_airlines = nama_airlines;
    }

    public void infoPesawat(){
        System.out.println("Jenis Pesawat : " + this.jenis_pesawat);
        System.out.println("Nama Air Line : " + this.nama_airlines);
    }
}
```

Listing 4.7 Super class Pesawat dengan 2 (dua) constructor method

Pada class Pesawat terdapat 2(dua) buah contructor method, yaitu tanpa parameter dan menggunakan parameter. Adapun pemanggilan constructor tergantung pada objek pemanggilnya, seperti pada contoh listing 4.8 berikut ini

```
public class PesawatKomersil extends Pesawat{

    /*Constructor tanpa parameter yang memanggil constructor tanpa parameter super classnya */
    public PesawatKomersil(){
        super();
    }
    /*Constructor yang memanggil constructor dengan parameter super classnya */
    public PesawatKomersil(String jenis, String nama){
        super(jenis,nama);
    }

    public static void main(String []args){
        PesawatKomersil pesawatku = new PesawatKomersil();
        PesawatKomersil pesawatmu = new PesawatKomersil("BOEING", "GARUDA AIRLINES");
        System.out.println("PESAWATKU");
        pesawatku.infoPesawat();
        System.out.println("PESAWATMU");
        pesawatmu.infoPesawat();
    }
}
```

Listing 4.8 Sub class PesawatKomersil

Jika kode program tersebut dijalankan maka akan tampil keluaran seperti pada gambar 4.2 berikut ini

```
PESAWATKU
Jenis Pesawat : null
Nama Air Line : null
PESAWATMU
Jenis Pesawat : BOEING
Nama Air Line : GARUDA AIRLINES
```

Gambar 4.2 Hasil output program

Hasil jenis pesawat dan nama air line dari objek pesawatku adalah null dikarenakan objek pesawatku memanggil constructor tanpa parameter yang tidak mempunyai implementasi program, sehingga tidak ada nilai yang diberikan pada constructor method. Sedangkan pada pesawatmu memanggil constructor dengan parameter, sehingga nilai BOEING dan GARUDA AIRLINES diberikan pada masing-masing parameter.

3. Polymorphism dan Overriding

Secara harfiah arti dari polymorphism adalah mempunyai banyak bentuk. Polymorphism adalah sebuah konsep yang sangat penting dari pemrograman berorientasi objek. Pada bahasa pemrograman JAVA konsep ini dikenali dengan penggunaan lebih dari satu method yang memiliki nama yang sama. Sebagai contoh misalkan seorang guru mempunyai kewajiban untuk melakukan pengajaran, akan tetapi tiap-tiap guru tersebut mempunyai perbedaan matakuliah apa yang diajarkan, begitu juga dengan jumlah jam mata pelajaran, sehingga dapat diartikan bahwa satu method mengajar mempunyai banyak bentuk dalam implementasinya. Polymorphism berkaitan erat dengan proses pewarisan (inheritance), karena proses pembentukannya melalui super class dan sub class. Penggunaan method dengan nama yang sama dapat diimplementasikan dengan method overriding. Overriding method adalah fungsi yang digunakan untuk menimpa implementasi dari program dengan menggunakan nama method yang sama. Berikut ini adalah contoh listing untuk lebih memahami penggunaan polymorphism dan overriding.

```
/**
 * Nama file : Guru.java
 * @author javajawara
 */
public class Guru {

    protected String nama_guru;

    public Guru(String nama_guru){
        this.nama_guru = nama_guru;
    }

    public void infoGuru(){
        System.out.println("Saya adalah seorang Guru");
    }
}
```

Listing 4.9 Super Class Guru

Class Guru merupakan super class yang akan di extends pada class lainnya sehingga properti dan method yang terdapat didalam class Guru dapat digunakan pada class lain yang menjadi turunan dari class Guru. Berikutnya adalah menambahkan kode pada class Mapel seperti pada listing 4.10 berikut ini

```
/**
 *
 * @author javajawara
 */
public class Mapel extends Guru {

    protected String nama_mapel;

    public Mapel(String nama_mapel, String nama_dosen){
        super(nama_dosen);
        this.nama_mapel = nama_mapel;
    }

    public void infoGuru(){
        System.out.println("Nama Guru : " + super.nama_guru);
        System.out.println("Mengajar Mata Pelajaran : " + this.nama_mapel);
    }
}
```

Listing 4.10 Sub Class Mapel

Pada class Mapel diatas terlihat bahwa properti yang terdapat pada super class (Guru) yaitu nama_guru ikut dikonstruksi bersamaan dengan properti dari class Mapel, selain itu terdapat pula penggunaan overriding method, karena terdapat method yang sama dengan yang ada pada super class yaitu infoGuru, perbedaannya method pada class Mapel memiliki output yang akan ditampilkan ketika program dieksekusi. Selanjutnya adalah pembuatan class JamMengajar, seperti pada listing 4.11 berikut ini

```
/**
 *
 * @author javajawara
 */
public class JamMengajar extends Guru {

    protected int jml_jam;

    public JamMengajar(int jml_jam, String nama_dosen){
        super(nama_dosen);
        this.jml_jam = jml_jam;
    }

    public void infoGuru(){
        System.out.println("Nama Guru : " + super.nama_guru);
        System.out.println("Jumlah Jam Mengajar : " + this.jml_jam + " CAM");
    }
}
```

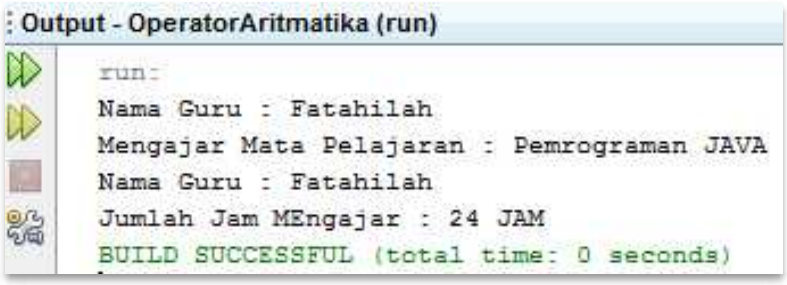
Listing 4.11 Sub class JamMengajar

Pada sub class JamMengajar, memiliki penjelasan yang sama dengan class Mapel. Hanya saja pada output menampilkan jumlah jam mengajar. Kemudian diakhir adalah pembuatan class Sekolah dimana terdapat method main yang merupakan fungsi untuk penciptaan beberapa objek seperti pada listing 4.12 berikut ini

```
/**
 *
 * @author javajawara
 */
public class Sekolah {

    public static void main(String[] args) {
    // TODO code application logic here
    Guru guruku = null;

    Mapel mapel = new Mapel("Pemrograman JAVA", "Fatahilah");
    JamMengajar jml_jam = new JamMengajar(24, "Fatahilah");
    guruku = mapel;
    guruku.infoGuru();

    guruku = jml_jam;
    guruku.infoGuru();
    }
}
```

Listing 4.12 Class Sekolah

Gambar 4.3 adalah hasil output ketika program dijalankan



Gambar 4.3 Hasil Output

B. Keterampilan yang diperlukan dalam Membuat Program Dengan Konsep Berbasis Objek

1. Mengidentifikasi penggunaan Inheritance
2. Menulis class 2 (dua) atau lebih pada file yang berbeda.
3. Mengorganisasikan Super class dan Sub class.
4. Mengimplementasikan polymorphism
5. Mengimplementasikan overrding method dan overloading method.
6. Menggunakan keyword super

C. Sikap kerja yang diperlukan dalam Membuat Program Dengan Konsep Berbasis Objek

Harus bersikap secara:

1. Cermat dan teliti dalam mengorganisasikan super class dan sub class
2. Sering melakukan uji coba kode program.
3. Sesuai dengan kaidah penulisan sintaks bahasa pemrograman JAVA.
4. Berpikir analitis serta evaluatif waktu melakukan uji coba kode program.

BAB V
MEMBUAT PROGRAM OBJECT ORIENTED DENGAN INTERFACE DAN PAKET

A. Pengetahuan yang diperlukan dalam Membuat Program Dengan Konsep Berbasis Objek

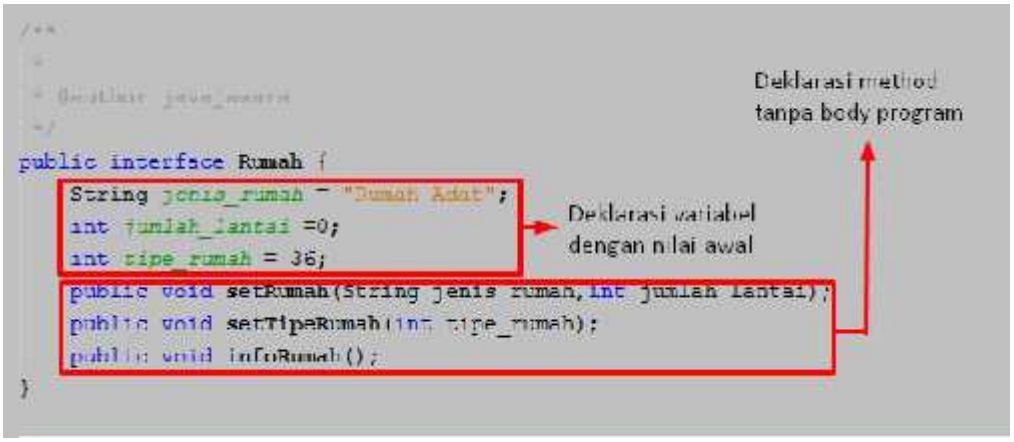
1. Interface

Interface digunakan untuk membuat kesamaan pada nama method ataupun variabel. Misal pada dunia nyata sebuah Mobil mempunyai cara untuk mengubah porsenelling, akan tetapi antara mobil manual dan mobil automatic akan berbeda cara penggunaanya. Interface terlihat seperti class, tapi bukan class, karena keyword yang digunakan adalah Interface bukan class. Interface merupakan kumpulan deklarasi method tanpa body program. Selain itu didalam interface diperbolehkan mendeklarasikan variabel dan harus diberi nilai awal sehingga mempunyai bentuk final (tidak dapat diubah nilainya).

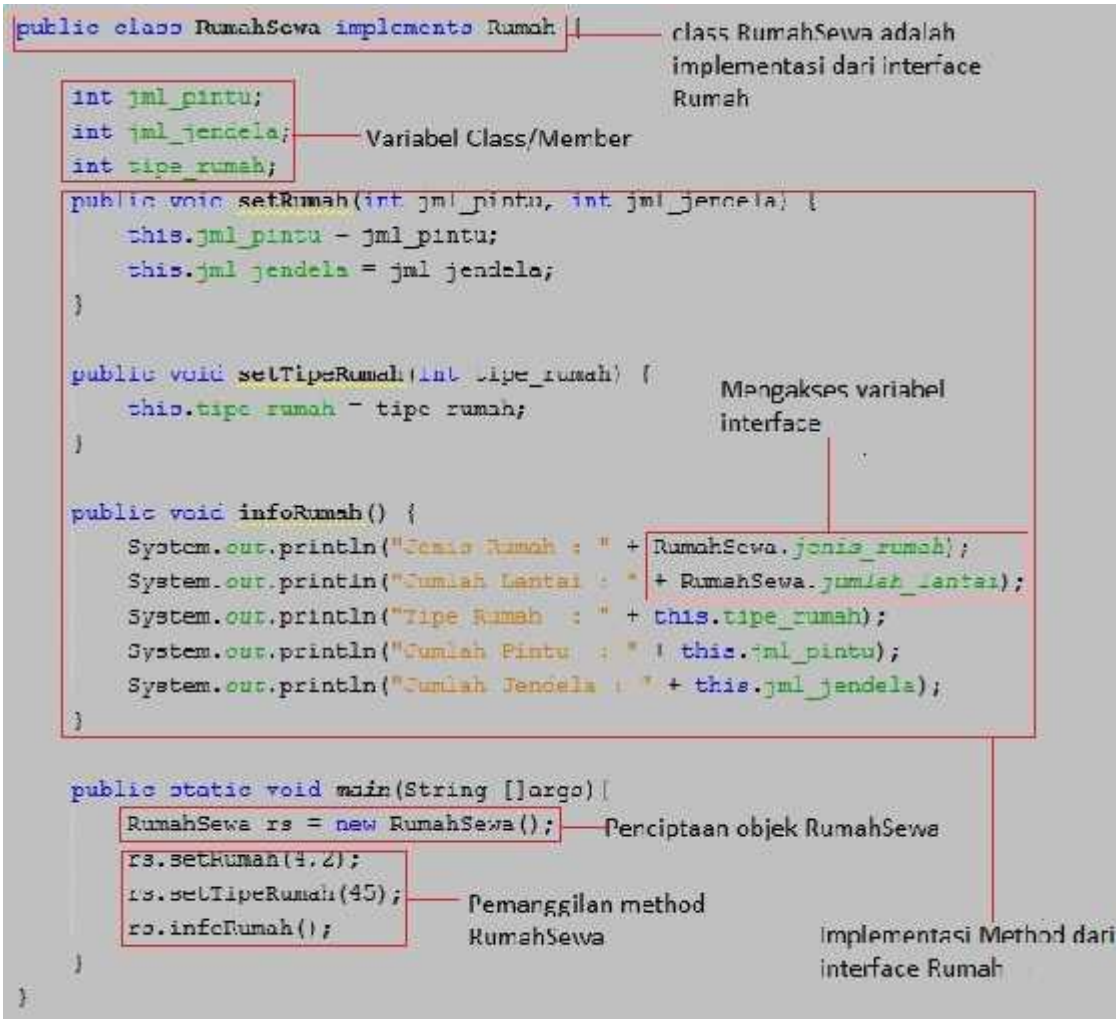
Interface diperlukan karena seperti dalam bahasa pemrograman C/C++ terdapat konsep multiple inheritance, artinya satu sub class atau child class dapat memiliki banyak super class/parent class. Jadi pada pemrograman C/C++ diijinkan sebuah class anak mempunyai banyak class orang tua. Pada bahasa pemrograman JAVA hal tersebut tidak diperbolehkan karena bertentangan dengan konsep dunia nyata bahwa seorang anak pasti hanya mempunyai satu orang tua.

Untuk mengatasi permasalahan tersebut maka digunakan konsep interface. Sintaks penggunaan interface adalah <[modifier]> interface <nama interface>. Untuk class yang mengimplentasikan interface diharuskan mengimplementasikan semua method yang terdapat dalam interface, jadi tidak diperbolehkan hanya mengimplementasikan sebagian saja. Sintak untuk melakukan implementasi interface adalah menggunakan keyword implements yaitu <[modifier]> class <nama class> [extends <super class>] implements <nama interface>. Jika

mengimplementasikan banyak interface, maka tiap-tiap interface dipisahkan dengan tanda koma(,). Gambar 5.1 dan 5.2 berikut adalah contoh interface dan class yang mengimplementasikannya.



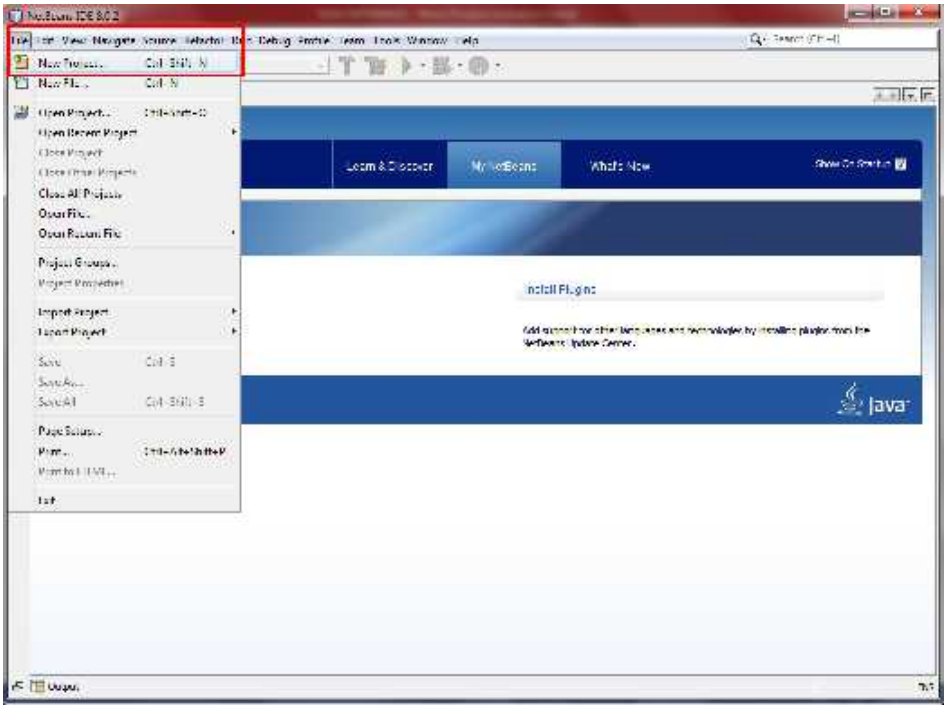
Gambar 5.1 Contoh interface



Gambar 5.2 Contoh class yang mengimplementasikan interface

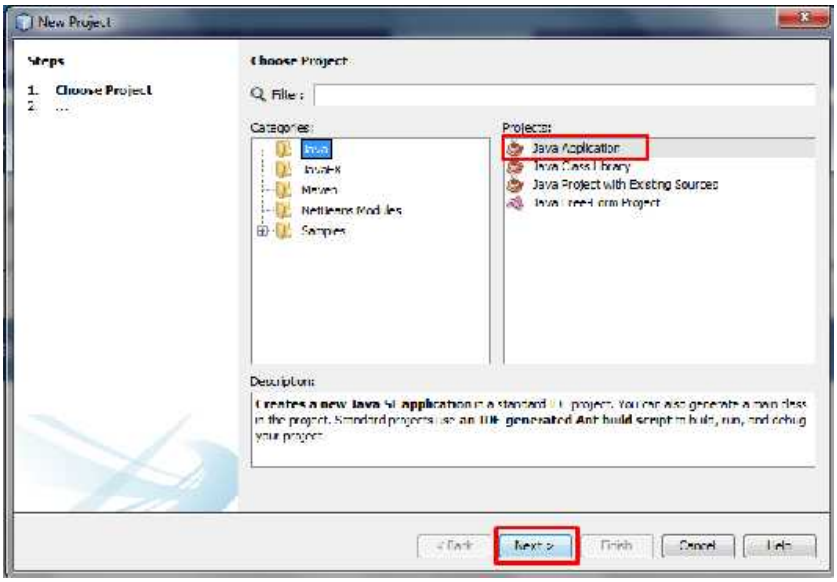
2. Langkah Pembuatan Interface

1. Bukalah editor software Netbeans yang telah terinstall di komputer. Kemudian lanjutkan dengan melakukan klik File → New Project seperti pada gambar 5.3



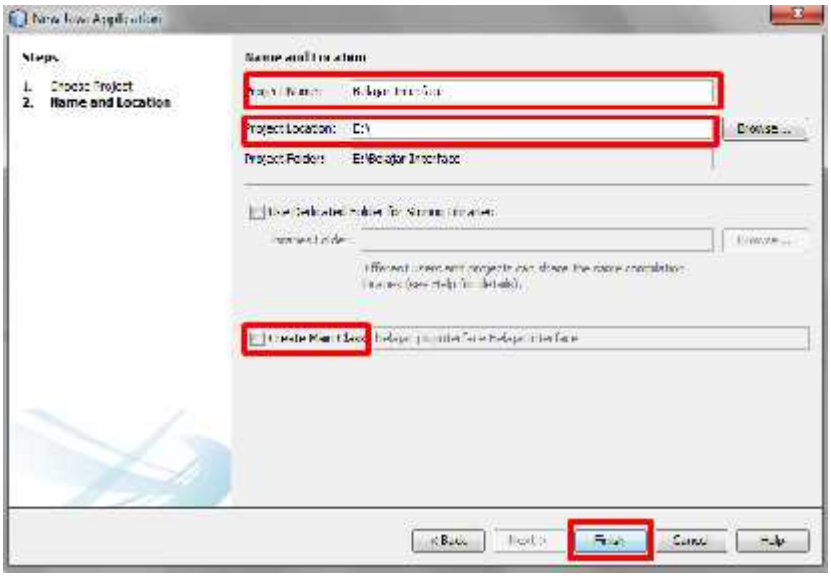
Gambar 5.3 Tampilan IDE Netbeans

2. Jika berhasil maka akan muncul tampilan dialog untuk menentukan jenis project seperti pada gambar 5.4, lanjutkan dengan memilih Java Application dan klik Next



Gambar 5.4 Penentuan jenis project

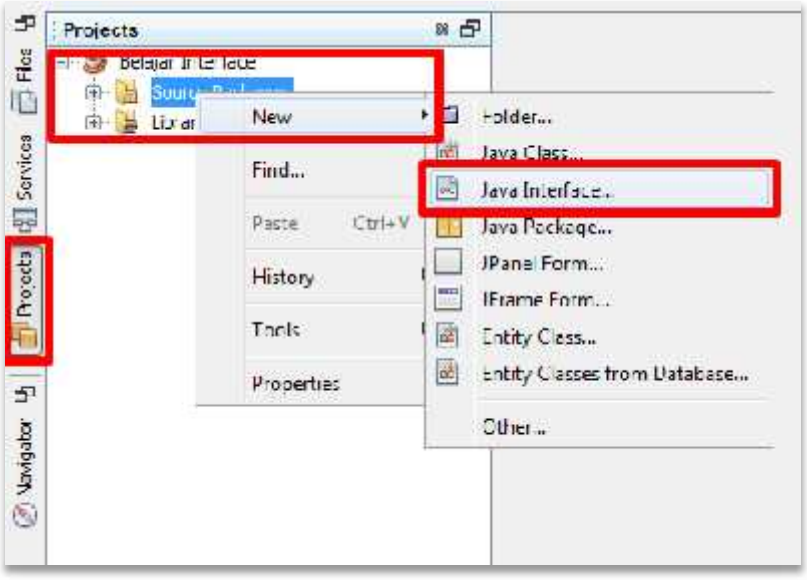
3. Isikan parameter yang dibutuhkan dalam project seperti pada gambar 5.5



Gambar 5.5 Pengisian parameter project Interface

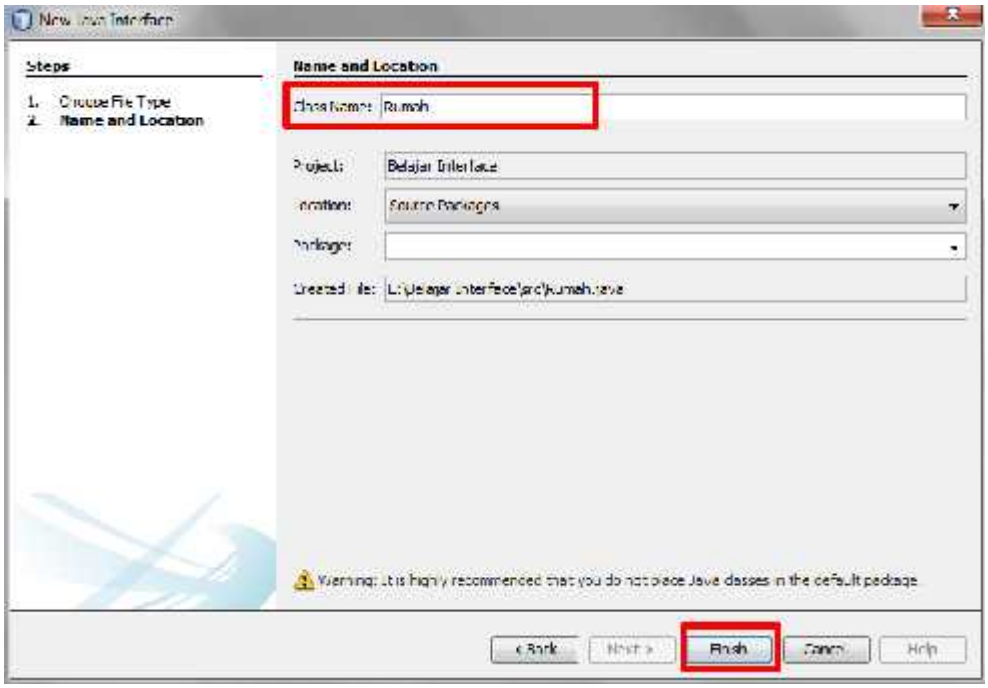
Pada contoh tersebut field Project Name diisikan Belajar Interface, kemudian tempat penyimpanan file di drive E:\ sesuaikan dengan komputer masing-masing, dan hilangkan centang pada field Create Main Class.

4. Klik pada tab projects, kemudian klik kanan pada bagian source packages dan pilih New → Java Interface seperti pada gambar 5.6



Gambar 5.6 Pembuatan interface

5. Isikan class name yang akan dibuat, misal nama class Rumah, kemudian lanjutkan dengan klik next seperti pada gambar 5.7



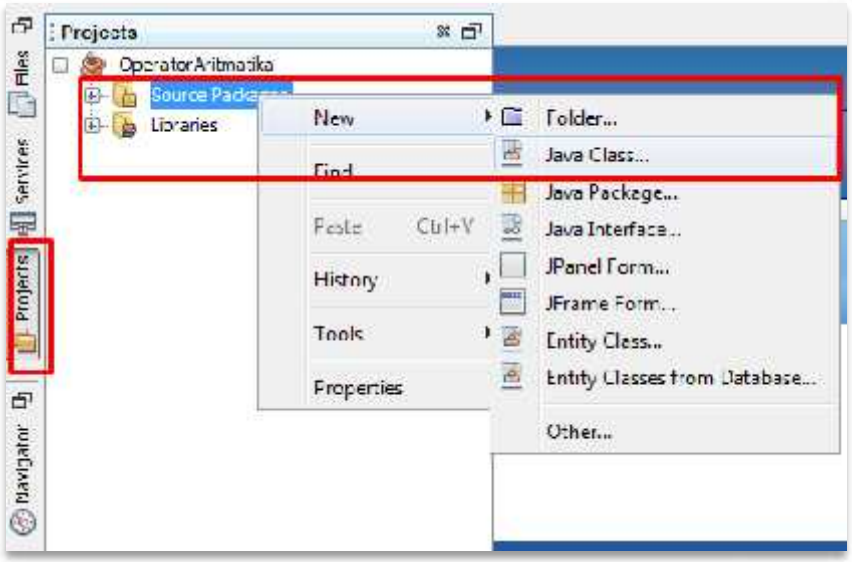
Gambar 5.7 Pengisian Class Name

6. Isikan file Rumah.java dengan listing program seperti pada listing 5.1 berikut ini

```
public interface Rumah {
    String jenis_rumah = "Rumah Adat";
    int jumlah_lantai =2;
    public void setRumah(int jml_pintu,int jml_jendela);
    public void setTipeRumah(int tipe_rumah);
    public void infoRumah();
}
```

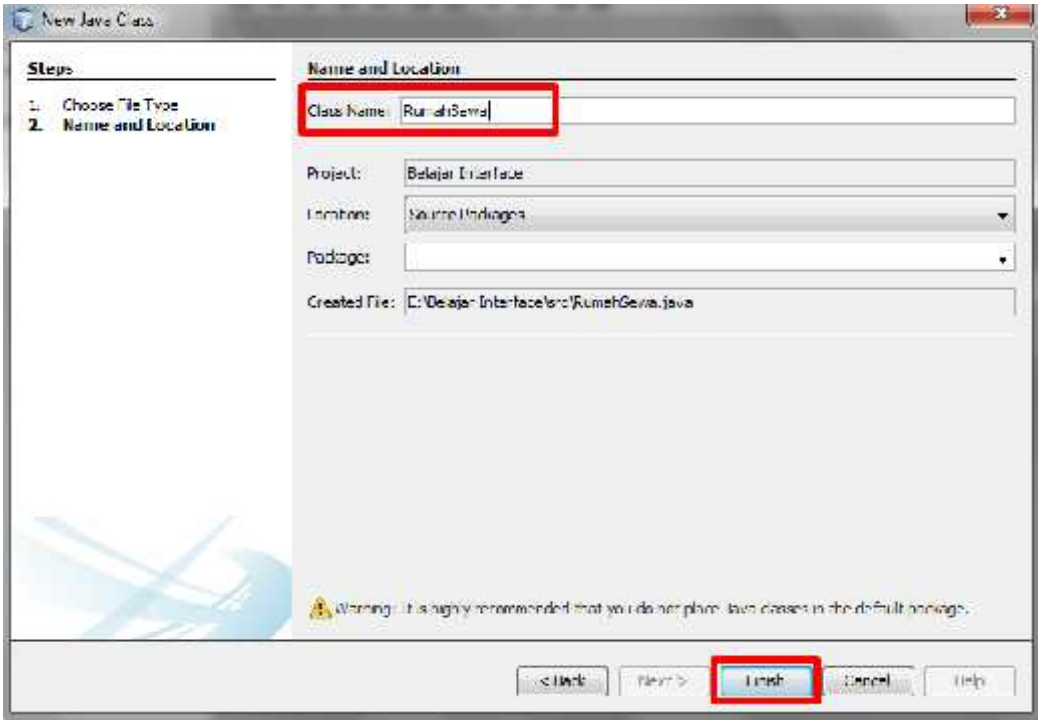
Listing 5.1 Interface Rumah

7. Klik pada tab projects, kemudian klik kanan pada bagian source packages dan pilih New → Java Class seperti pada gambar 5.8



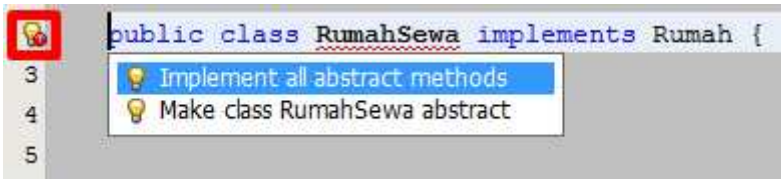
Gambar 5.8 Pembuatan Java Class

8. Isikan nama class dengan RumahSewa seperti pada gambar 5.9 berikut ini

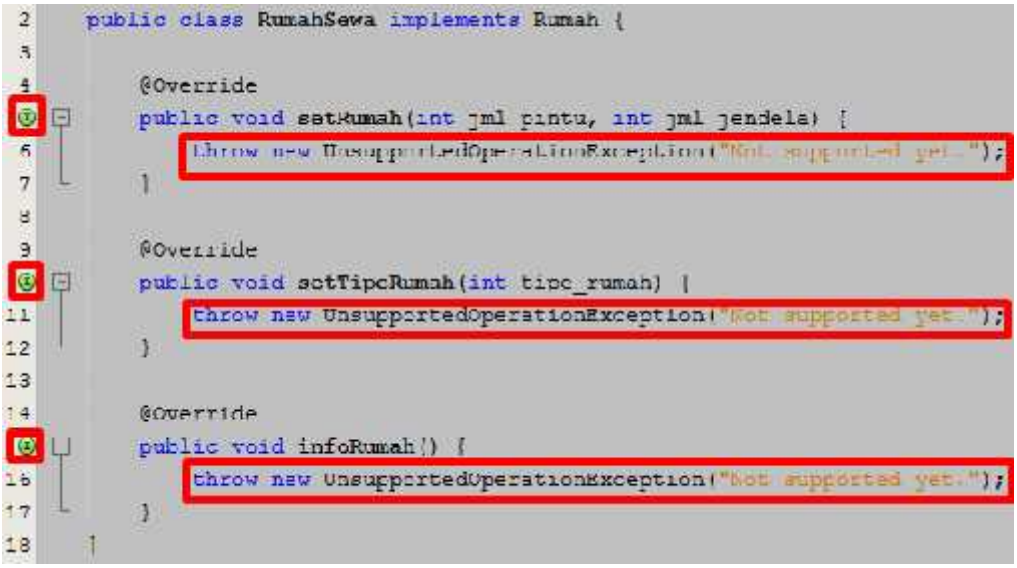


Gambar 5.9 Pengisian nama class

9. Isikan file RumahSewa.java seperti pada listing 5.2 berikut ini, hasil dari kode program tersebut akan error dikarenakan class tersebut belum mengimplementasikan semua method yang dimiliki oleh interface Rumah. Solusinya adalah melakukan klik pada tanda icon lampu disebelah kiri editor, dan pilih menu Implement all abstract methods, sehingga akan muncul listing program yang di generate oleh Netbeans seperti pada gambar 5.10



Listing 5.2 Kode program class RumahSewa



Gambar 5.10 Hasil generate listing Netbeans

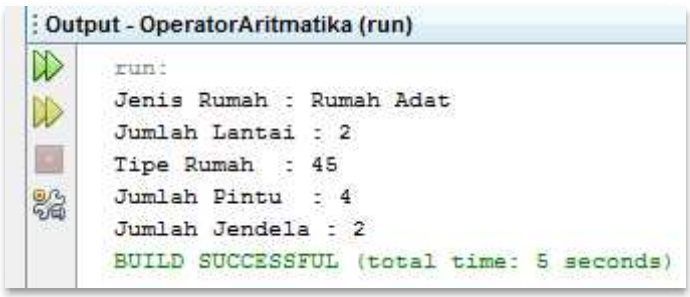
Tanda icon berwarna hijau disebelah kiri editor adalah menandakan bahwa method tersebut melakukan override dari interface Rumah. Hapus listing program “throw new UnsupportedOperationException(“Not supported yet.”);”

10. Kemudian tambahkan kode program seperti pada listing 5.3 berikut ini

```
public class RumahSewa implements Rumah {

    int jml_pintu;
    int jml_jendela;
    int tipe_rumah;
    public void setRumah(int jml_pintu, int jml_jendela) {
        this.jml_pintu = jml_pintu;
        this.jml_jendela = jml_jendela;
    }

    public void setTipeRumah(int tipe_rumah) {
        this.tipe_rumah = tipe_rumah;
    }

    public void infoRumah() {
        System.out.println("Jenis Rumah : " + RumahSewa.jenis_rumah);
        System.out.println("Jumlah Lantai : " + RumahSewa.jumlah_lantai);
        System.out.println("Tipe Rumah  : " + this.tipe_rumah);
        System.out.println("Jumlah Pintu  : " + this.jml_pintu);
        System.out.println("Jumlah Jendela : " + this.jml_jendela);
    }

    public static void main(String []args){
        RumahSewa rs = new RumahSewa();
        rs.setRumah(4,2);
        rs.setTipeRumah(45);
        rs.infoRumah();
    }
}
```

Listing 5.3 Class RumahSewa

11. Jalankan program dengan menekan tombol Shift + F6, dan hasil output nya adalah seperti gambar 5.11 berikut ini

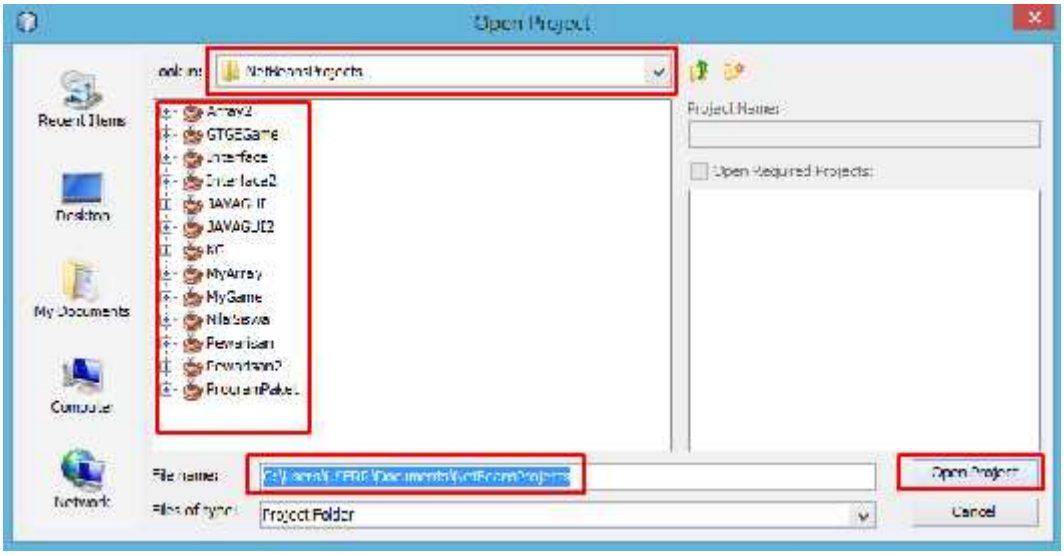


Gambar 5.11 Hasil output program

3. Membuat Paket dengan Program

Sebuah program yang telah siap pakai, maka harus di distribusikan dalam bentuk paket. Fungsi dari pembuatan paket ini adalah agar program tersebut dapat dijalankan di semua komputer, tanpa melalui editor pemrograman seperti Netbeans. Bentuk distribusinya berupa paket installer dimana fungsinya adalah melakukan registrasi setiap library yang digunakan dalam program seperti pada program menggunakan JAVA, maka setiap komputer yang menjalankannya memerlukan JRE (Java Runtime Environment) . Terdapat beberapa tahapan dalam pembuatan paket yaitu

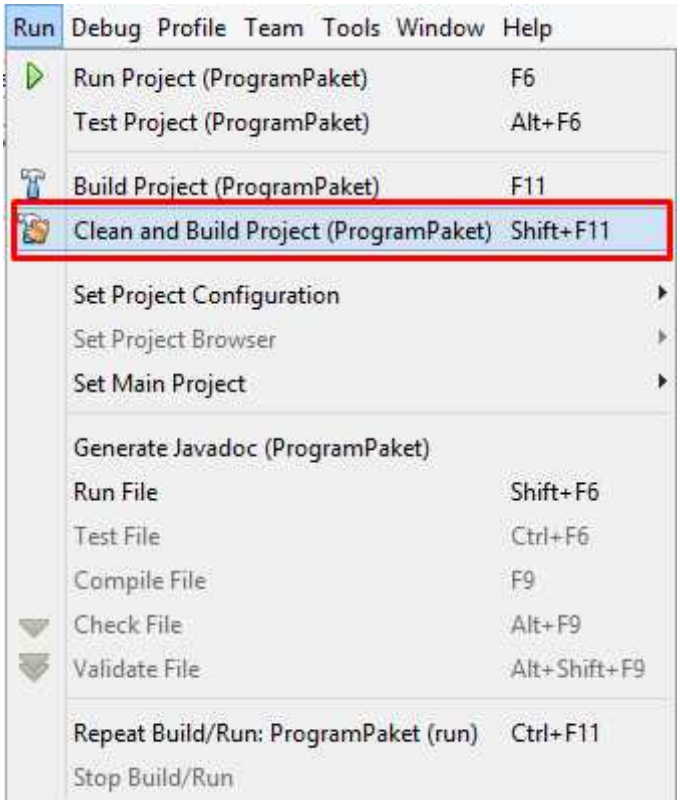
1. Membuat aplikasi berbentuk JAR (Java Archieve) kusus untuk aplikasi yang dibuat menggunakan Bahasa pemrograman JAVA. Cara untuk membuatnya adalah jalankan editor Netbeans, kemudian pilih menu File → Open Project maka akan tampil dialog box dan kemudian pilih project program yang akan dijadikan JAR menggunakan, seperti pada gambar 5.12 berikut ini



Gambar 5.12 Tampilan Dialog box Open Project

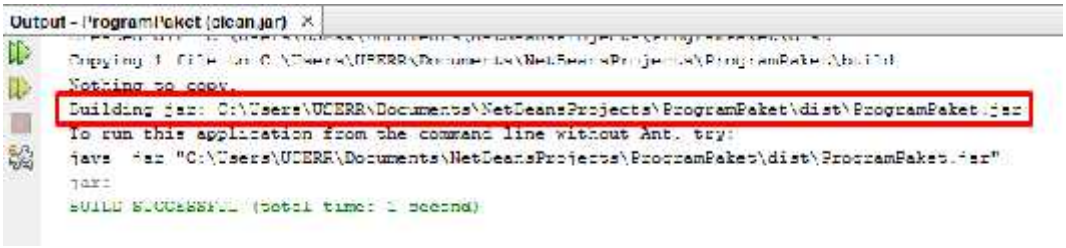
Pilihlah lokasi folder penyimpanan project yang telah dibuat. Jika belum mempunyai project yang akan dibuat, dapat mengunduh contoh program pada alamat url http://bit.ly/2EQlhc0,

lakukan pengekstrakan file tersebut menggunakan program archiever seperti Win Rar atau Winzip, kemudian buka project melalui netbeans. Proses selanjutnya adalah membuat file JAR dengan memilih menu Run → Clean and Build Project seperti pada gambar 5.13 berikut ini



Gambar 5.13 Menu Clean and Build Project

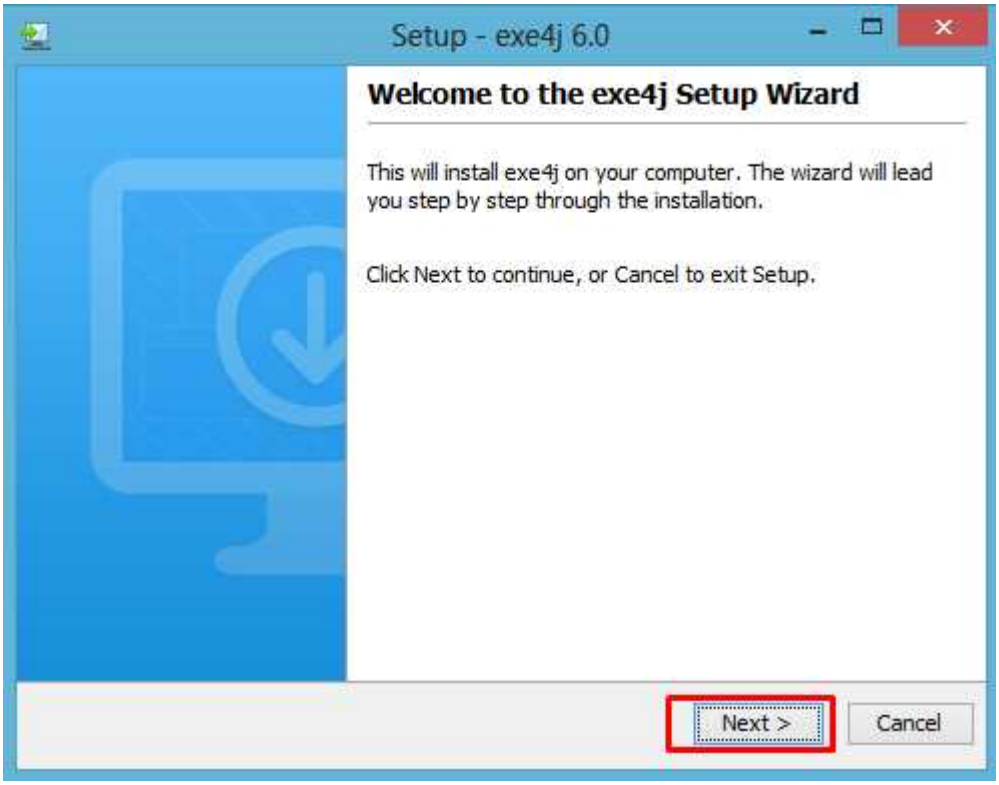
Jika proses berhasil, maka perhatikan pada tampilan output disisi bawah editor terdapat keterangan bahwa file JAR telah sukses dibuat, beserta letak lokasi, pada umumnya file tersebut akan berada pada folder dist, seperti pada gambar 5.14 berikut ini



Gambar 5.14 Hasil dari Pembuatan file JAR

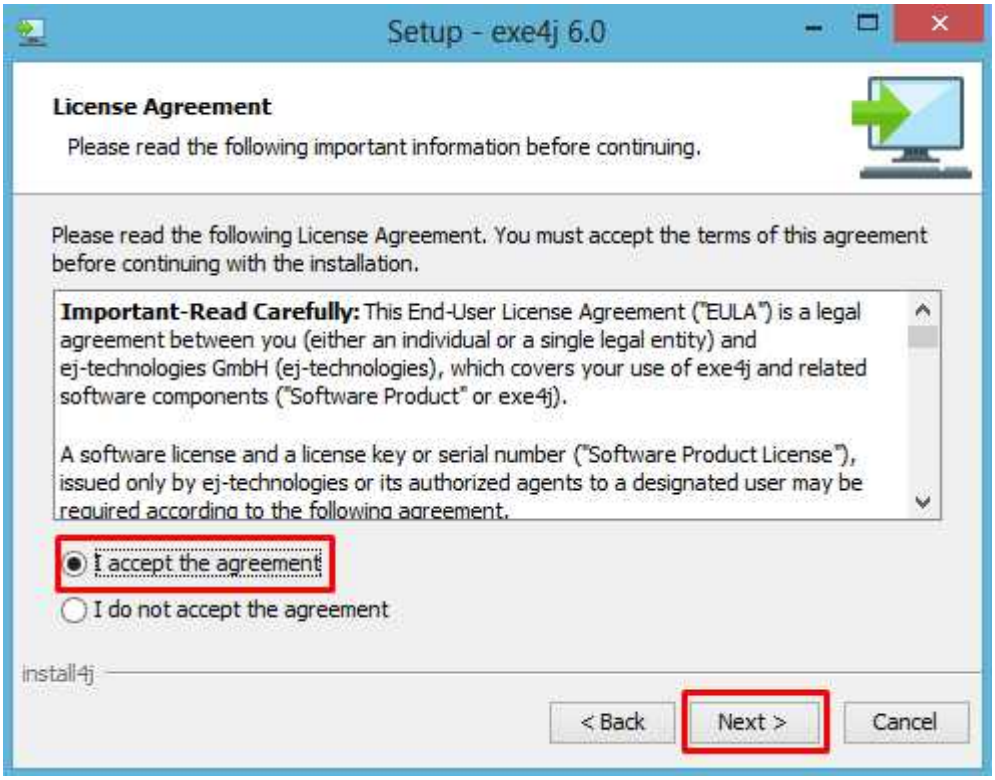
2. Mengkonversi file JAR (Java Archieve) menjadi file exe (executable)

   Pada proses ini, dibutuhkan software tambahan yang mungkin sangat banyak di internet, akan tetapi pada modul ini menggunakan produk EXE4J yang dapat anda unduh di alamat url http://bit.ly/2GpCqq5 . Lakukan pengunduhan dan sesuaikan dengan sistem operasi pada komputer masing-masing, kemudian jalankan software EXE4J dengan melakukan double klik pada file yang telah diunduh, maka akan tampil seperti pada gambar 5.15 berikut ini dan klik Next



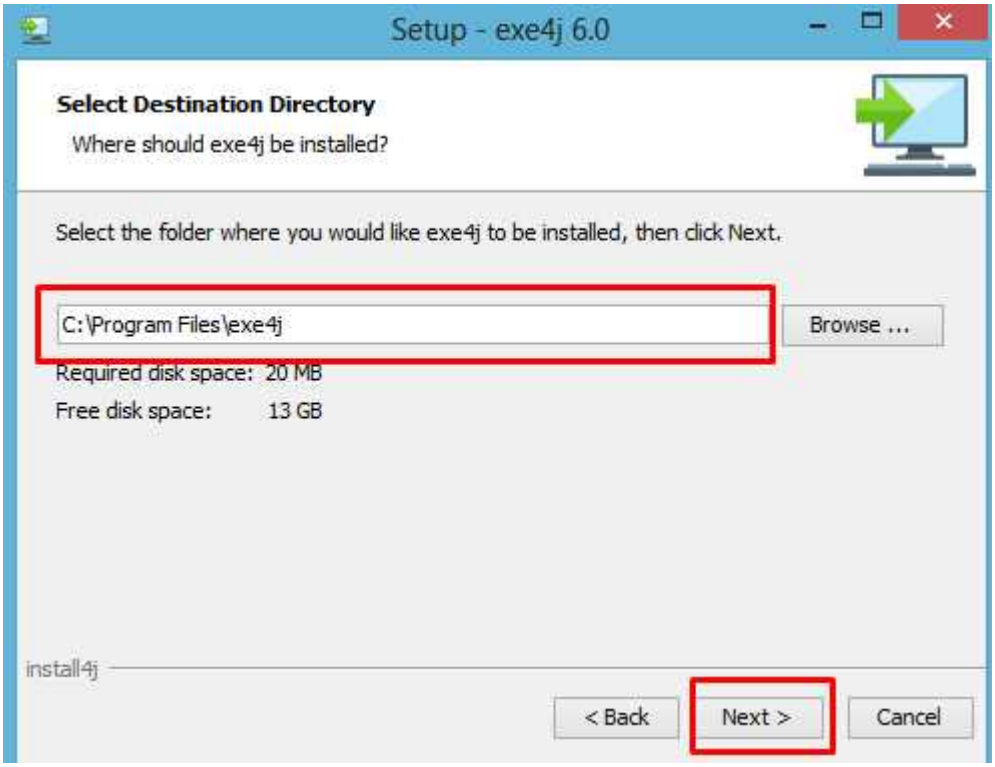
Gambar 5.15 Setup EXE4J

Proses berikutnya adalah tampil dialog konfirmasi persetujuan untuk menyatakan bahwa pengguna telah menyetujui aturan lisensi yang telah ditetapkan oleh pembuat software EXE4J. Centang pilihan "I accept the agreement" seperti pada gambar 5.16



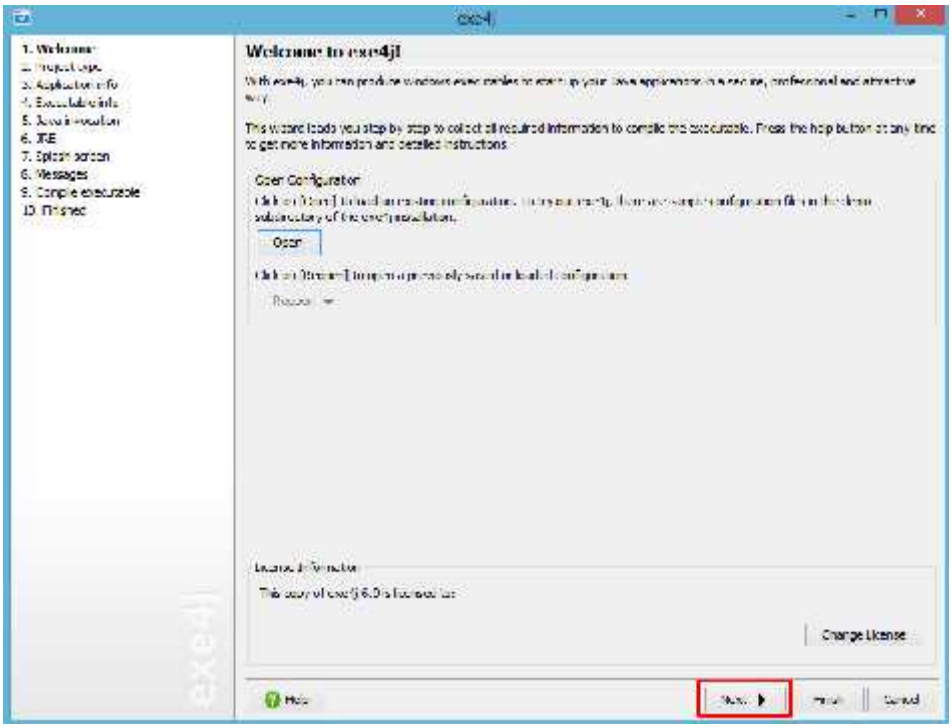
Gambar 5.16 Persetujuan Lisensi

Kemudian tampil pilihan lokasi folder untuk hasil instalasi, kemudian pilih tombol Next seperti pada gambar 5.17



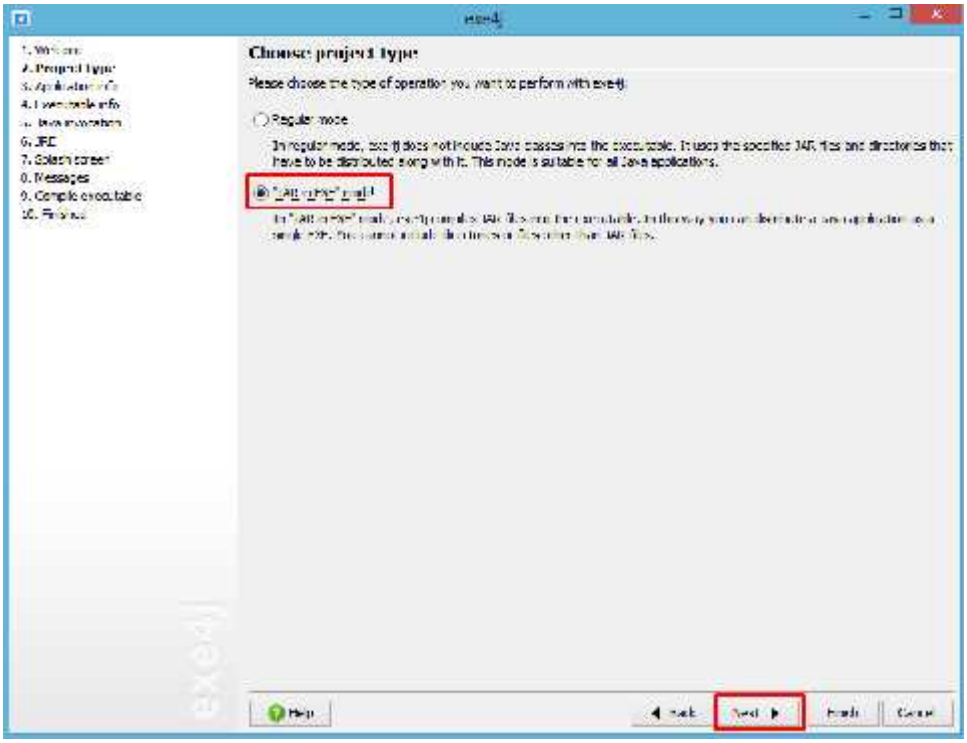
Gambar 5.17 Pemilihan direktori instalasi

Tunggu sejenak untuk menyelesaikan proses instalasi, setelah selesai lakukan klik Run dan akan tampil seperti pada gambar 5.18 berikut ini



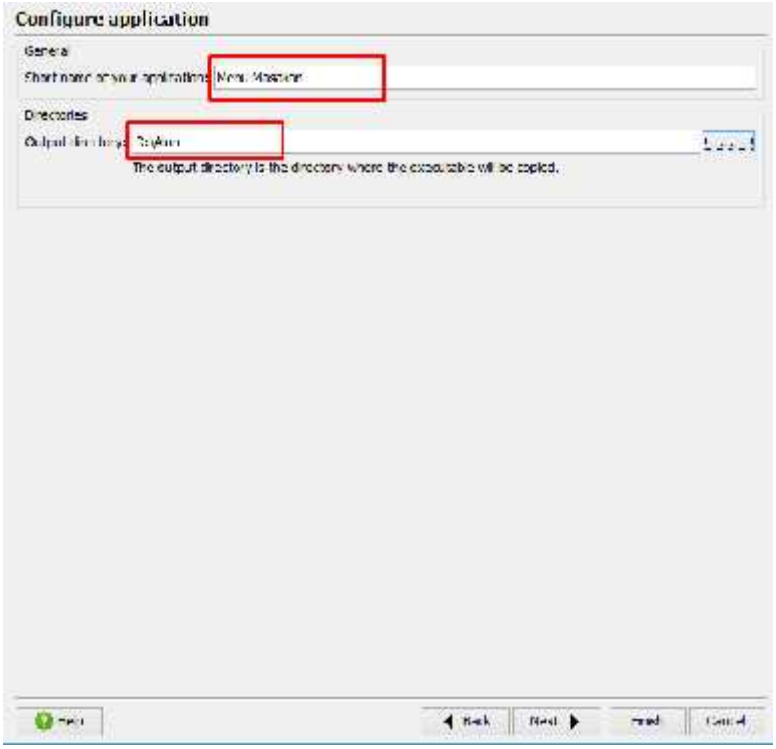
Gambar 5.18 Tampilan awal EXE4J

Kemudian klik tombol Next, maka akan tampil seperti pada gambar 5.19



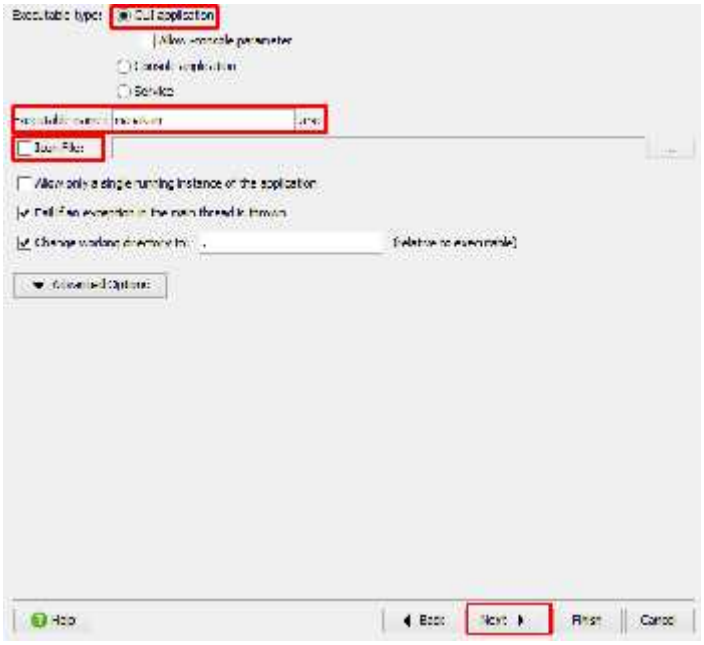
Gambar 5.19 Pemilihan Tipe project

Pilihlah opsi JAR in EXE mode, dan lanjutkan dengan melakukan klik pada tombol Next dan akan tampil seperti pada gambar 5.20 berikut ini



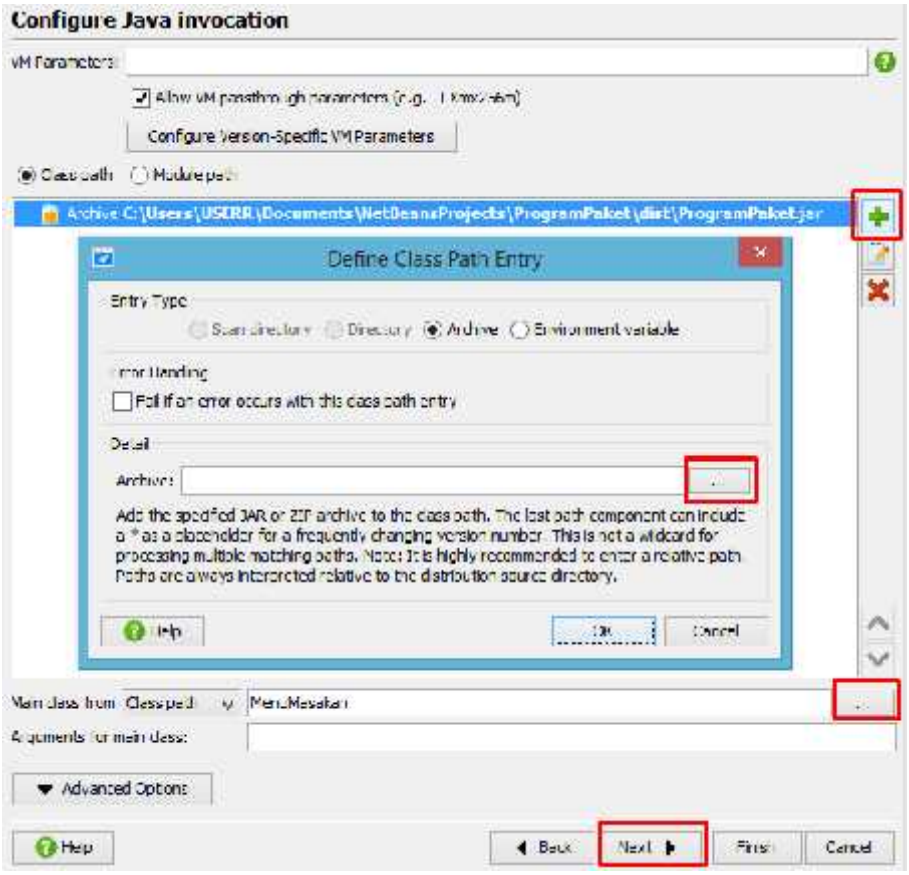
Gambar 5.20 Pengisian nama aplikasi dan output direktori

Isikan nama dari aplikasi yang akan dibuat beserta lokasi direktori hasil pembuatan installer, kemudian lanjutkan menekan tombol Next dan akan tampil seperti gambar 5.21 berikut ini



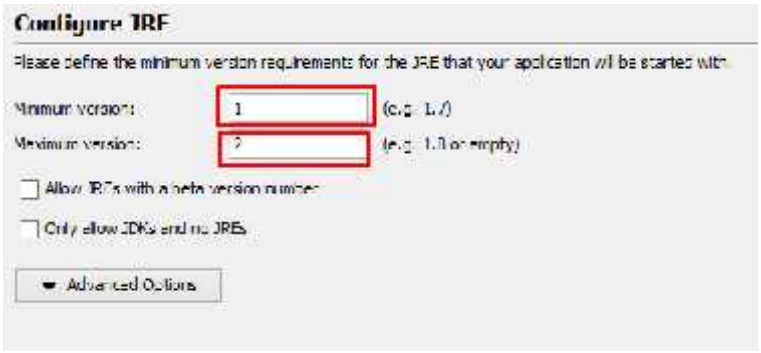
Gambar 5.21 Tipe executable

Proses berikutnya adalah pemilihan file JAR yang telah dibuat pada tahapan sebelumnya dengan cara melakukan klik pada tombol +, kemudian isikan lokasi dari file JAR dan isikan main method untuk setting Class path, lanjutkan dengan klik pada tombol Next seperti pada gambar 5.22 berikut ini



Gambar 5.22 Pengisian file JAR dan Main Class

Tahapan berikutnya adalah untuk konfigurasi JRE, isikan nilai minimum = 1 dan maximum =2 dan lanjutkan dengan klik pada tombol Next seperti pada gambar 5.23 berikut ini

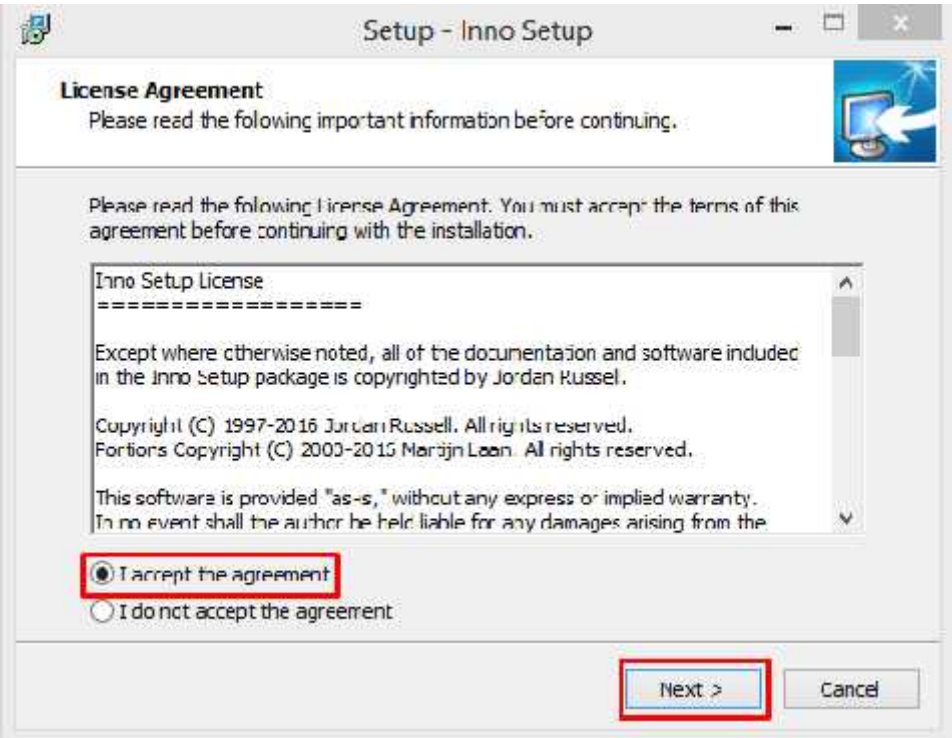


Gambar 5.23 Konfigurasi JRE

Selanjutnya adalah hanya melakukan klik Next jika tanpa gambar splash screen, tetapi jika akan menggunakan isikan file gambar yang akan digunakan, kemudian klik Next sekali lagi dan proses akan berjalan membuat file executable.

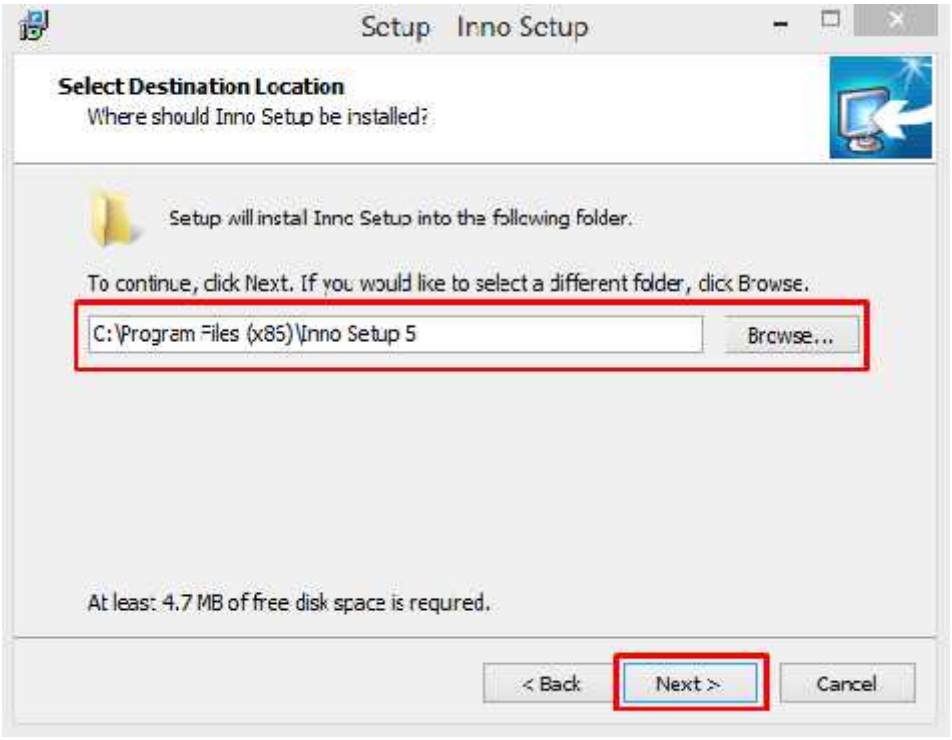
3. Pembuatan installer menggunakan program

   Pada tahapan ini menggunakan software Inno Setup, yang dapat diunduh pada alamat url http://www.jrsoftware.org/isdl.php . Setelah berhasil melakukan pengunduhan maka jalankan file tersebut maka akan tampil persetujuan lisensi, pilih I accept the agreement dan klik tombol Next seperti pada gambar 5.24 berikut ini



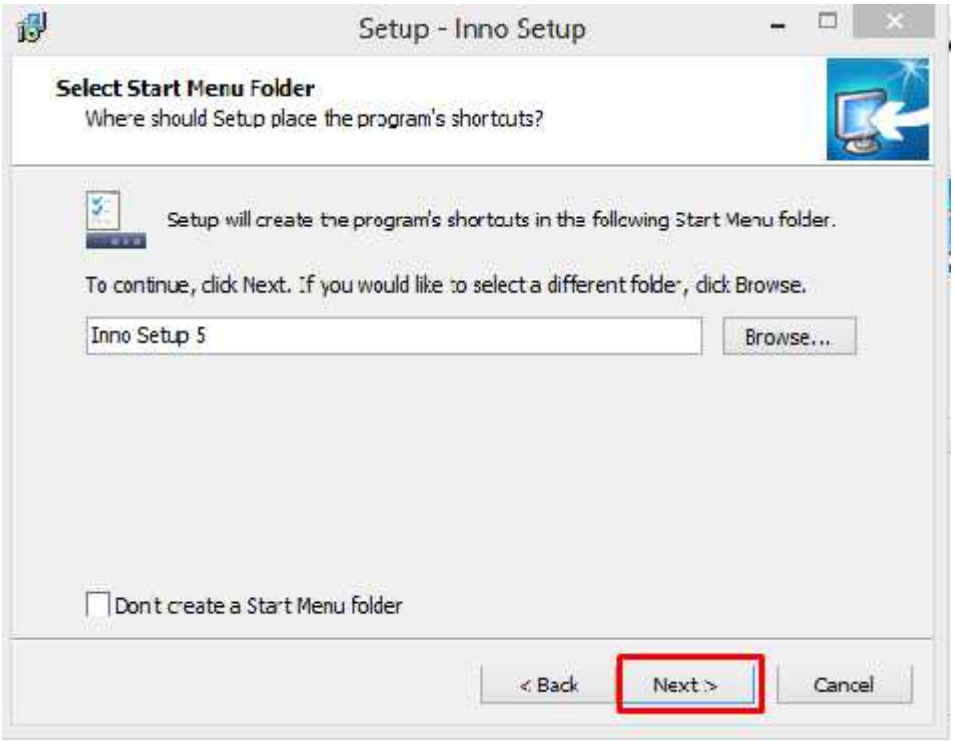
Gambar 5.24 Persetujuan lisensi Inno Setup

Proses selanjutnya adalah menentukan letak direktori hasil instalasi Inno Setup dan lanjutkan klik tombol Next seperti pada gambar 5.25 berikut ini



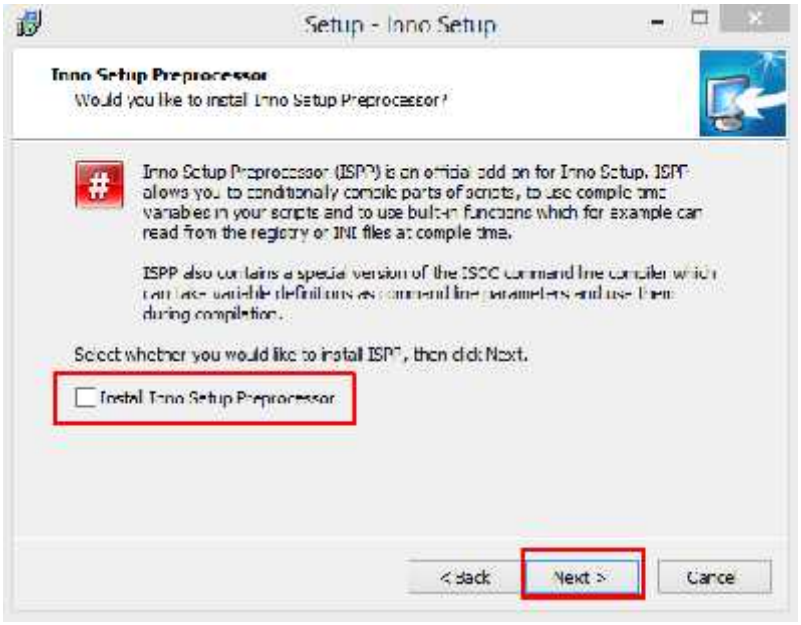
Gambar 5.25 Pemilihan folder instalasi

Proses berikutnya adalah pembuatan shortcut untuk ditampilkan pada start menu folder, pada tahapan ini biarkan semua diatur secara default, lanjutkan dengan klik tombol Next seperti pada gambar 5.26 berikut ini



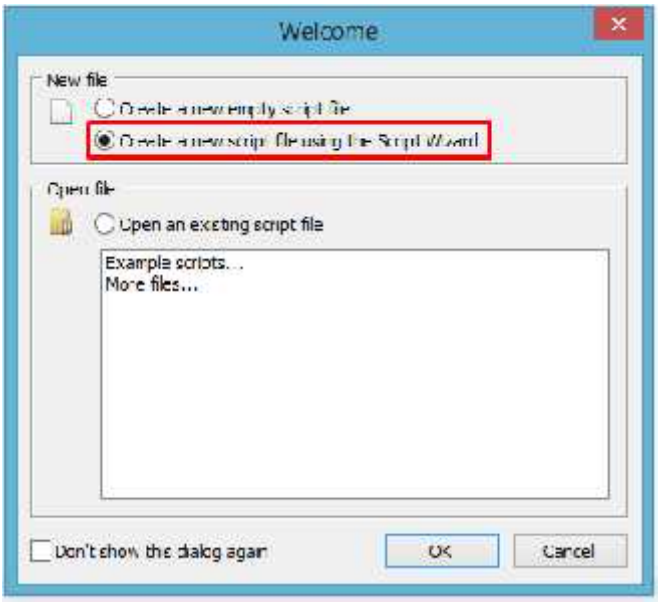
Gambar 5.26 Pembuatan shortcut

Langkah selanjutnya adalah melakukan uncheck pada pilihan Install Inno Setup Preprocessor dan klik tombol Next, artinya saat ini belum diperlukan untuk menambahkan plugin tambahan seperti pada gambar 5.27 berikut ini



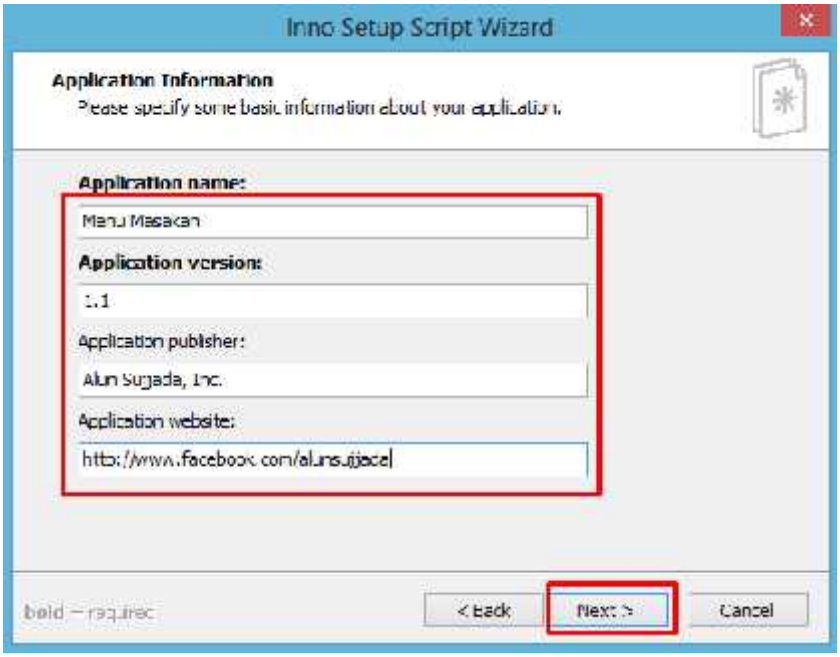
Gambar 5.27 Instalasi Inno Setup Preprocessor

Kemudian lewatkan bagian Additional task dan langsung melakukan klik tombol Next, kemudian aplikasi Inno Setup akan siap untuk di install dan lakukan klik tombol install. Setelah semua proses instalasi selesai, jalankan program Inno Setup maka akan tampil program seperti pada gambar 5.28 dan pilih opsi Create a new script file using the script wizard



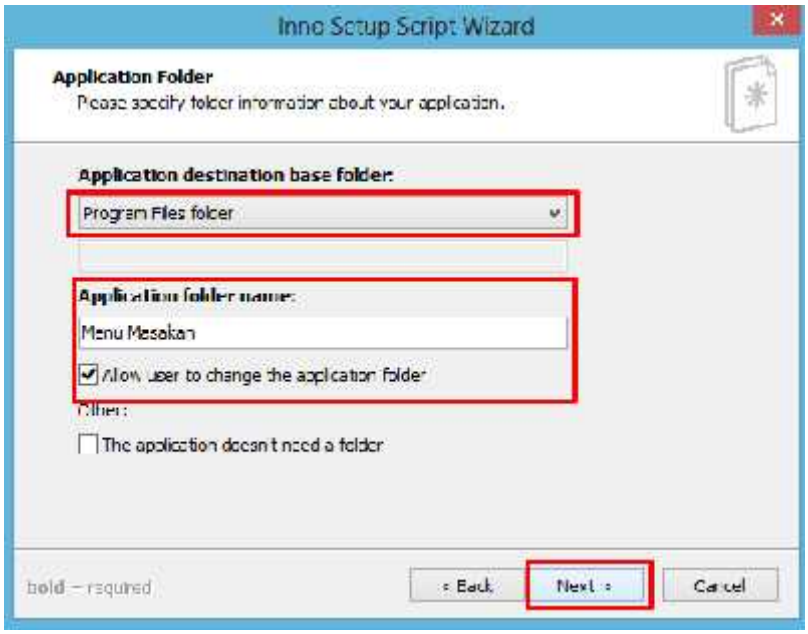
Gambar 5.28 Pemilihan Tipe installer

Setelah melakukan klik pada tombol OK maka akan muncul dialog wizard, pada tahapan ini cukup dengan klik tombol Next dan akan muncul dialog pengisian parameter seperti nama aplikasi, versi, penerbit dan alamat website seperti pada gambar 5.29 berikut ini



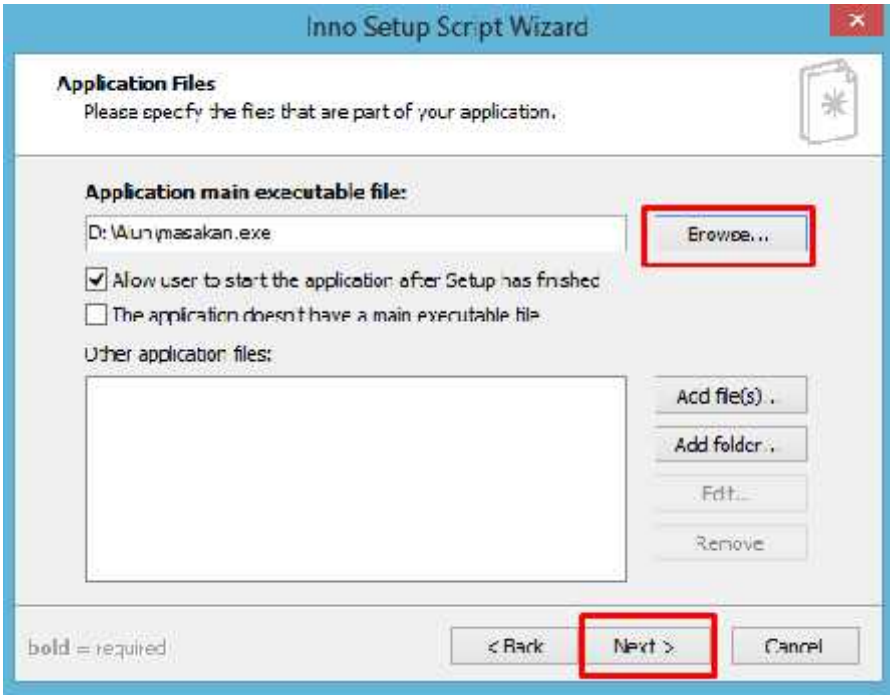
Gambar 5.29 Dialog pengisian parameter aplikasi

Setelah melakukan klik tombol Next, proses selanjutnya adalah pemilihan spesifikasi folder hasil instalasi file executable seperti pada gambar 5.30 berikut ini



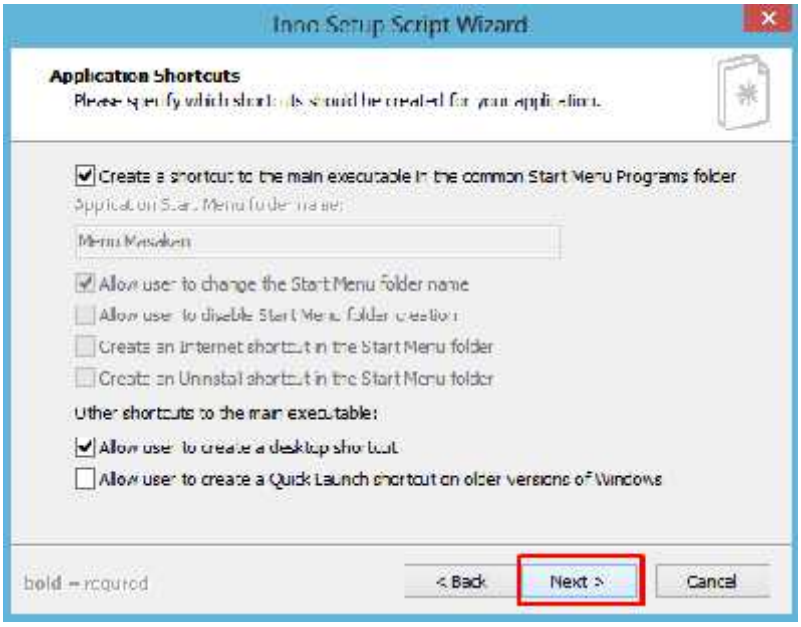
Gambar 5.30 Spesifikasi folder hasil instalasi

Tahapan selanjutnya adalah pemilihan file executable hasil dari konversi JAR yang telah dibuat pada tahapan sebelumnya dengan melakukan klik pada tombol Browse dan lanjutkan dengan klik tombol Next seperti pada gambar 5.31 berikut ini



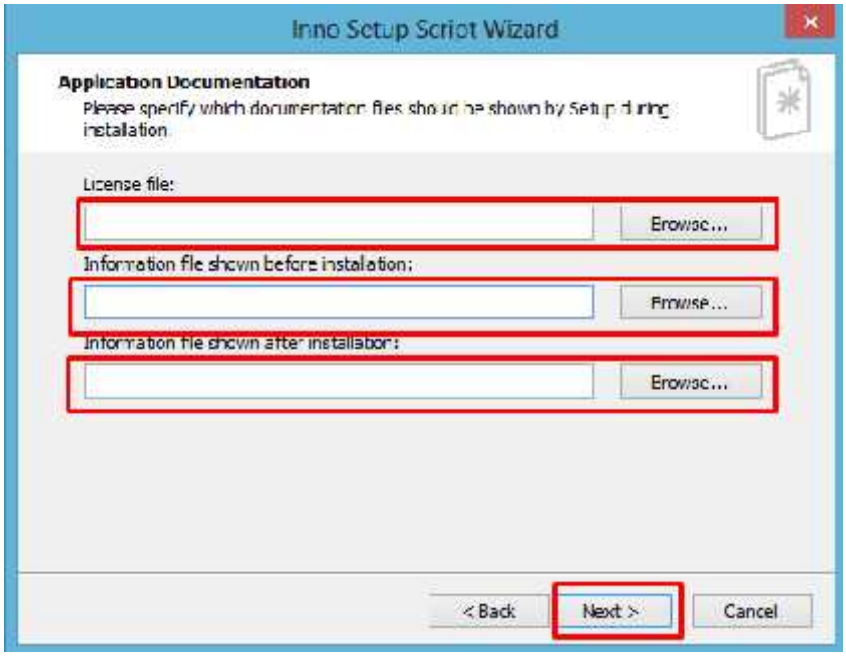
Gambar 5.31 Spesifikasi Aplikasi

Proses selanjutnya adalah melakukan klik pada dialog pemilihan shortcut dan klik tombol Next seperti pada gambar 5.32 berikut ini



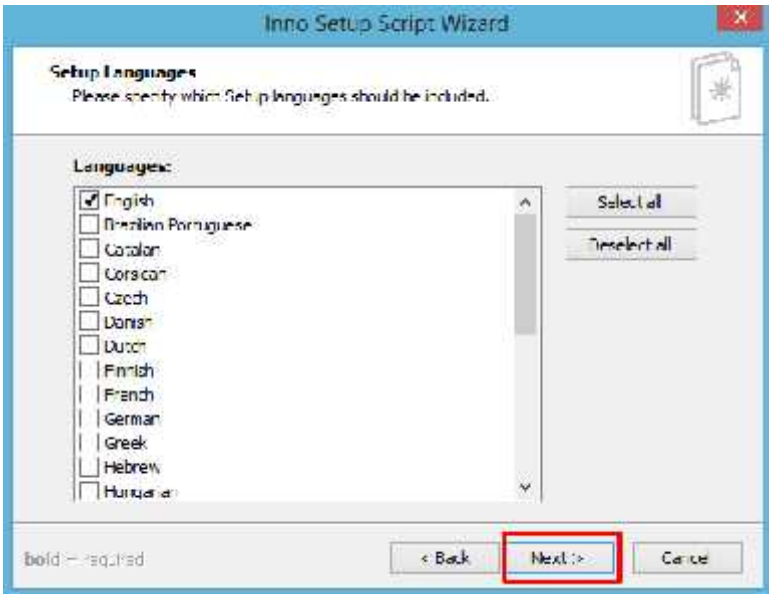
Gambar 5.32 Spesifikasi shortcut aplikasi

Tahapan berikutnya adalah pengisian dokumentasi aplikasi jika mempunyai lisensi, informasi file sebelum instalasi dan sesudah instalasi, untuk tahapan saat ini dikosongkan terlebih dahulu dan lanjutkan dengan klik tombol Next seperti pada gambar 5.33 berikut ini



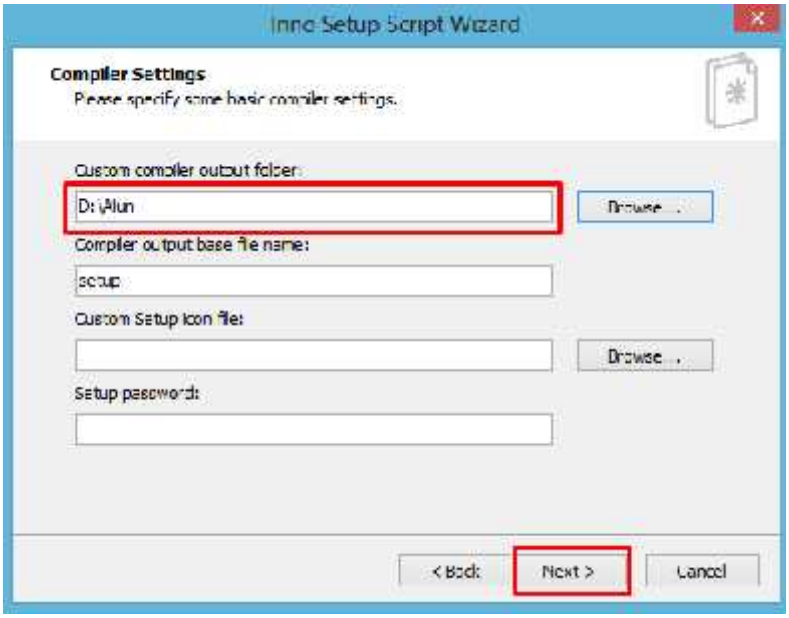
Gambar 5.33 Dokumentasi Aplikasi

Proses berikutnya adalah pemilihan Bahasa yang dipakai pada saat proses instalasi, sebagai bahasa default adalah menggunakan Bahasa Inggris, dan lanjutkan dengan klik tombol Next seperti pada gambar 5.34 berikut ini



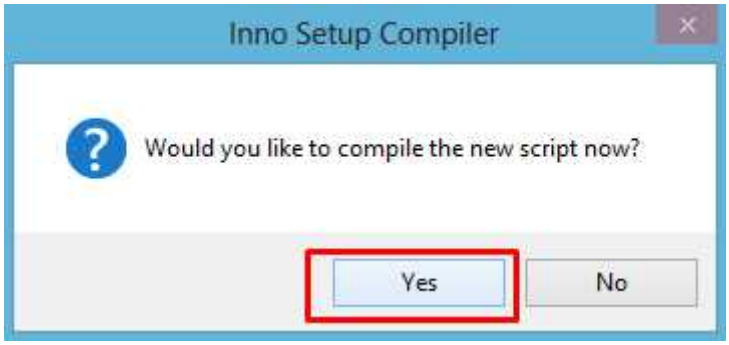
Gambar 5.34 Pemilihan Bahasa

Tahapan selanjutnya adalah menentukan letak hasil dari pembuatan installer dan lanjutkan dengan klik tombol Next seperti pada gambar 5.35 berikut ini



Gambar 5.35 Penentuan folder hasil pembuatan installer

Kemudian lanjutkan dengan klik tombol Yes untuk melakukan proses kompilasi script untuk pembuatan file installer seperti pada gambar 5.36 berikut ini



Gambar 5.36 Dialog kompilasi script

Jika proses berhasil maka pada folder output yang telah diisikan akan terbuat file secara otomatis file setup.exe. Double klik file tersebut untuk melakukan instalasi program.

B. Keterampilan yang diperlukan dalam Membuat Program Object Oriented Dengan Interface Dan Paket

1. Mengidentifikasi penggunaan Inheritance
2. Menulis class 2 (dua) atau lebih pada file yang berbeda.
3. Mengorganisasikan Super class dan Sub class.
4. Mengimplementasikan polymorphism
5. Mengimplementasikan overrding method dan overloading method.
6. Menggunakan keyword super
7. Mengoperasikan software IDE (Integrated Development Environment) untuk bahasa pemrograman JAVA seperti Netbeans, GEL, Eclipse dan lain-lain
8. Melakukan instalasi dan menjalankan software EXE4J.
9. Melakukan instalasi dan menjalankan software Inno Setup.

C. Sikap kerja yang diperlukan dalam Membuat Program Object Oriented Dengan Interface Dan Paket

Harus bersikap secara:

1. Cermat dan teliti dalam mengorganisasikan super class dan sub class
2. Sering melakukan uji coba kode program.
3. Sesuai dengan kaidah penulisan sintaks bahasa pemrograman JAVA.
4. Berpikir analitis serta evaluatif waktu melakukan uji coba kode program.

BAB VI
MENGKOMPILASI PROGRAM

A. Pengetahuan yang Diperlukan dalam Mengkompilasi Program

1. Perbaikan Kesalahan

Sebuah program yang muncul pesan kesalahan atau error adalah hal yang wajar pada dunia pemrograman. Semakin kompleks dan besar skala program maka akan semakin banyakj kemungkinan terjadinya error. Hampir tidak ada satupun program yang tidak memiliki error, semua dikarenakan keterbatasan manusia sebagai pembuatnya. Melihat kenyataan tersebut yang dapat dilakukan oleh seorang programmer adalah berusaha untuk mencari kesalahan-kesalahan tersebut baik kesalahan besar ataupun kecil untuk diperbaiki terus menerus, melakukan update program dengan tujuan untuk meminimalisir sebuah error pada program. Bayangkan jika kesalahan menuliskan hasil output 1000 menjadi 100, sudah pasti hal tersebut akan mengganggu kinerja dari penggunanya. Untuk mengenali kesalahan-kesalahan tersebut maka setidaknya seorang programmer mengetahui jenis-jenis kesalahan program. Secara garis besar error pada program dibagi menjadi 3 (tiga) yaitu

a. Syntax Error

Kesalahan yang paling sering ditemukan pada saat membuat program adalah kesalahan sintaks atau Syntax Error, dimana perintah atau statement yang diketikkan menyalahi aturan pengkodean yang dimiliki oleh bahasa pemrograman yang digunakan.

Sebuah bahasa pemrograman memiliki aturan pengkodean tersendiri yang harus dipatuhi, sebagai contoh pada bahasa pemrograman JAVA, setiap statement diwajibkan diakhiri dengan tanda titik koma (;). Jika tidak dituliskan, maka program akan menampilkan pesan Syntax Error pada saat dijalankan. Setiap

bahasa pemrograman memiliki keyword, yaitu perintah-perintah baku yang digunakan. Sebagai contoh, keyword yang umum adalah kondisi if, perulangan for atau while, penulisan fungsi dan lambang aritmatika seperti modulus, pangkat, dan lain-lain. Kesalahan penulisan keyword juga merupakan Syntax Error.

Kesalahan penulisan parameter pada sebuah function atau procedure juga termasuk dalam kategori Syntax Error, misalnya jika function yang digunakan memerlukan parameter sementara programmer lupa menuliskan parameter tersebut. Syntax Error merupakan jenis kesalahan yang paling sering ditemui, tetapi juga pada umumnya paling mudah untuk ditanggulangi. Syntax Error cukup mudah diketahui karena setiap bahasa pemrograman yang digunakan akan menampilkan pesan kesalahan dan menunjukkan baris kesalahan dengan tepat. Pada beberapa bahasa pemrograman, disediakan fasilitas Auto Syntax Check, dimana akan muncul sebuah pesan peringatan ketika mengetikkan sintaks yang salah.

b. Run-time Error

Jenis kesalahan Run-time Error terjadi ketika kode program melakukan sesuatu yang tidak dimungkinkan. Contohnya pada saat sebuah aplikasi mencoba mengakses file yang tidak terdapat dalam media penyimpanan, atau terjadi kesalahan alokasi ruang memori. Terkadang Run-time Error terjadi karena berbagai aspek dan tidak selalu karena kesalahan pemrograman, sebagai contoh jika programmer sengaja menghapus beberapa file penting yang digunakan oleh suatu aplikasi, maka terdapat kemungkinan akan terjadi Run-time Error pada saat aplikasi tersebut dijalankan. Run-time Error menyebabkan sebuah pogram akan langsung berhenti. Walaupun demikian, pencegahan semaksimal mungkin dengan memberikan validasi dan pesan yang user-friendly saat terjadi kesalahan pada aplikasi, akan sangat membantu untuk

mengetahui mengapa aplikasi tidak berjalan dengan semestinya. Salah satu cara mengatasi Run-time error dalam bahasa pemrograman JAVA yaitu menggunakan blok kode try catch dan finally, jadi jika terdapat kesalahan maka akan muncul dialog box sesuai dengan apa yang akan diinformasikan mengenai kesalahan tersebut. Contoh penggunaanya seperti pada listing program 6.1 berikut ini

```
import javax.swing.*;
public class KoreksiKesalahan {

    public static void main(String []args){

        String bilangan1 = JOptionPane.showInputDialog("Isikan Nilai Bilangan 1 ");
        String bilangan2 = JOptionPane.showInputDialog("Isikan Nilai Bilangan 2 ");

        int hasil = Integer.parseInt(bilangan1) / Integer.parseInt(bilangan2);
        JOptionPane.showMessageDialog(
                null,"Hasil " + bilangan1 + " dibagi dengan " + bilangan2 + "=" + hasil);

    }
}
```

Listing 6.1 Contoh program sederhana

Contoh program sederhana pada listing 6.1 adalah berfungsi untuk menghitung pembagian 2(dua) bilangan. Program tersebut tidak mengalami kesalahan sintaks, karena terbukti bahwa program tersebut dapat dikompilasi dan dijalankan, selain itu juga tidak terdapat kesalahan logika (logical error), karena ketika program diisikan bilangan1 dengan nilai 60 dan bilangan2 dengan nilai 5 maka hasil output yang tampil adalah 12 seperti pada gambar 6.1 berikut ini

Gambar 6.1 Hasil output program pembagian

Akan tetapi program tersebut mengalami Run-time Error ketika nilai pada bilangan2 diisikan dengan angka 0(nol), seperti pada gambar 6.2, karena dalam operasi aritmatika tidak diijinkan melakukan pembagian dengan 0(nol)

```
exception in thread "main" java.lang.ArithmeticException: / by zero
        at KoreksiKesalahan.main(KoreksiKesalahan.java:16)
C:\Users\USERR\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 2 seconds)
```

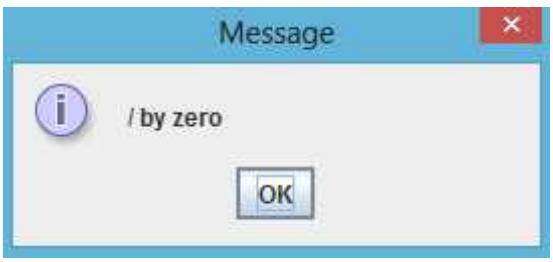
Gambar 6.2 Output hasil pembagian dengan bilangan 0

Untuk mengatasi hal tersebut seorang programmer perlu memperbaiki Runtime Error dengan menggunakan blok try catch dan finally seperti pada listing 6.2 berikut ini

```
import javax.swing.*;
public class KoreksiKesalahan {

    public static void main(String []args){
        int hasil=0;
        String bilangan1 = JOptionPane.showInputDialog("Isikan Nilai bilangan 1 ");
        String bilangan2 = JOptionPane.showInputDialog("Isikan Nilai bilangan 2 ");

        try{
            hasil = Integer.parseInt(bilangan1) / Integer.parseInt(bilangan2);

        }
        catch(ArithmeticException e){
            JOptionPane.showMessageDialog(null, e.getMessage());
        }
        finally{
            JOptionPane.showMessageDialog(
                null,"Hasil " + bilangan1 + " dibagi dengan " + bilangan2 + " " + hasil);
        }
    }
}
```

Listing 6.2 Perbaikan kesalahan program

Program tersebut ketika dijalankan dan bilangan2 diisikan dengan nilai 0(nol) maka akan muncul pesan kesalahan "/ by zero" seperti pada gambar 6.3 berikut ini



Gambar 6.3 Pesan kesalahan dari sistem

Ataupun dapat digantikan pesan kesalahan sesuai apa yang diinginkan dengan menuliskan program pada bagian catch.

c. Logical Error

Logical Error merupakan jenis kesalahan yang cukup sulit untuk ditemukan penyebabnya. Karena aplikasi yang mengandung Logical Error berjalan tanpa pesan kesalahan, tetapi mengeluarkan hasil yang tidak diharapkan, misalnya jika aplikasi program menghasilkan perhitungan yang salah. Logical Error baru dapat diketahui setelah dilakukan testing dan meninjau hasilnya. Logical Error dapat diperbaiki dengan memeriksa alur program dan nilai variabel yang dihasilkan.

2. Program Debugging

Sebuah error pada aplikasi disebut dengan istilah bug, atau dalam Bahasa Inggris berarti kutu atau binatang kecil. Konon istilah bug muncul karena ditemukannya binatang kecil yang menyebabkan kerusakan pada sebuah komputer tabung pada tahun 1945. Bug aplikasi terdapat pada kode program, yang dapat mengganggu kenyamanan pengguna aplikasi. Pada tingkat tertentu, keberadaan bug cukup memberikan efek yang besar.

Mungkin belum melupakan saat dimana orang ramai membicarakan "Y2K Bug" atau bug tahun 2000, atau munculnya istilah "Blue screen of Death" pada sistem operasi Windows, atau "Kernel Panic" pada sistem operasi Linux, semua contoh tersebut menunjukkan sebuah bug serius dapat mengakibatkan dampak negatif yang cukup besar.

Proses mencari penyebab bug disebut dengan debugging, yang juga merupakan tugas programer setelah menerima laporan bug. Walaupun demikian, jangan menjadikan pekerjaan sebagai pencari bug, untuk itu hanya ada satu cara, minimalkan bug pada aplikasi yang dibuat. Apa yang harus dilakukan untuk menghindari jenis-jenis kesalahan yang

telah disebutkan di atas. Bisa jadi tidak ada program yang sempurna, tetapi selalu ada cara untuk mengatasi atau menghindari kesalahan semaksimal mungkin.

a. Selalu Deklarasikan Variabel

Syntax Error, bahkan Logical Error, mungkin terjadi jika terdapat penulisan variabel yang salah. Sebaiknya mendeklarasikan variabel yang digunakan walaupun bisa jadi bahasa pemrograman yang digunakan mengijinkan untuk tidak melakukan deklarasi variabel. Visual Basic merupakan salah satu bahasa pemrograman yang mengijinkan penggunaan variabel tanpa deklarasi, walaupun demikian disarankan tetap menggunakan deklarasi variabel. Hal tersebut akan memperkecil kesalahan penulisan variabel. Masih dengan contoh Visual Basic yang dapat menambahkan perintah Option Explicit pada program untuk mencegah kesalahan tulis pada variabel, jika terdapat variabel yang belum dideklarasikan, maka Visual Basic akan menampilkan pesan kesalahan. Sebaiknya memiliki suatu skema standard untuk pemberian nama variabel dan konsisten dengan penggunaannya. Contohnya berikan nama variabel diawali dengan huruf s jika bertipe data string, misalnya sResult, sTemp, dan lain-lain. Pada Visual Basic maupun beberapa bahasa pemrograman lain yang berorientasi object, kita dapat mendeklarasikan variabel dengan tipe data object. Terdapat berbagai jenis macam object yang dikenal, dan sebaiknya menuliskannya dengan lengkap object yang dimaksud. Misalnya object ListBox, Label, dan lain-lain.

b. Gunakan Variabel Lokal

Sangat disarankan agar selalu menggunakan variabel lokal. Salah satu manfaatnya adalah jika terjadi kesalahan program (terutama Logical Error), maka penyebab kesalahan dan solusinya akan lebih mudah ditemukan. Hal ini dikarenakan variabel lokal memiliki ruang lingkup penggunaan yang lebih kecil dibandingkan variabel global, yang dapat diakses oleh procedure yang mana

saja. Penggunaan variabel global, sering menimbulkan kerancuan dan memperbesar kemungkinan terjadinya kesalahan tanpa disadari.

c. Kenali Jenis Bug

Bug yang timbul pada sebuah aplikasi memiliki karateristik. Karena itu selalu baca dan perhatikan baik-baik pesan kesalahan yang timbul. Beberapa jenis bug berdasarkan karakteristiknya adalah sebagai berikut:

i. Divide By Zero.

Jika pada sebuah pembagian, pembagi bernilai 0, maka program akan terhenti dan mengalami error.

ii. Infinite Loop.

Pengertian loop adalah perulangan, yang sering digunakan dalam teknik pemrograman. Penggunaan loop yang salah dapat mengakibatkan program menjalankan sebuah procedure tanpa akhir.

iii. Arithmatic overflow or Underflow.

Overflow terjadi saat sebuah perhitungan menghasilkan nilai yang lebih besar daripada nilai yang dapat ditampung oleh media/variabel penyimpan. Sementara underflow merupakan kebalikannya. Pada perhitungan aritmatik, hal ini sering ditemukan dan menjadi masalah.

iv. Exceeding Array Bounds.

Array merupakan variabel berdimensi yang memiliki indeks. Saat program mengakses indeks diluar array yang ditentukan, maka akan mengakibatkan error.

v. Access Violation.

Hal yang terjadi saat sebuah proses mencoba melewati batas yang diijinkan oleh sistem. Misalnya menulis sebuah nilai pada alamat memory, segmen, atau media yang diproteksi.

vi. Memory Leak.

Penggunaan memory yang tidak diinginkan, dapat terjadi karena program gagal melepaskan memory yang sudah tidak digunakan.

vii. Stack Overflow or Underflow.

Stack merupakan struktur data dengan prinsip LIFO (Last In First Out), pada program dapat mengimplementasikan logika stack untuk suatu tujuan. Tetapi jika stack melebihi atau dibawah nilai yang diijinkan oleh program, maka akan timbul kesalahan Stack Overflow/Underflow.

viii. Buffer Overflow.

Buffer merupakan tempat penyimpanan sementara dalam teknik pemrograman. Buffer Overflow terjadi jika menyimpan terlalu banyak data yang tidak dapat ditampung oleh buffer yang disediakan.

ix. Deadlock.

Merupakan suatu kondisi dimana dua atau lebih proses saling menunggu satu sama lain untuk menyelesaikan prosesnya, dan tidak satupun dari proses tersebut yang selesai. Problem deadlock sering ditemukan pada multiprocessing.

x. Off By One Error.

Merupakan istilah untuk menggambarkan perulangan yang terlalu banyak atau terlalu sedikit. Misalnya perulangan yang dikehendaki adalah lima kali, tetapi kenyataan yang terjadi aplikasi mengulang proses tersebut sebanyak empat kali atau enam kali. Kesalahan ini pada umumnya terjadi karena kesalahan logika penulisan kode pada proses perulangan.

xi. Berikan Komentar

Jangan kuatir kode program dipenuhi oleh komentar. Karena akan lebih mudah bagi programmer untuk mempelajari lagi kode-kode program yang pernah dibuat dengan membaca komentar. Dengan mengerti kode program dengan baik, maka akan menjadi lebih mudah jika pada suatu saat

terdapat Logical Error yang membutuhkan analisa ulang kode program.

xii. Gunakan Log File

Informasi proses yang dijalankan aplikasi dan tersimpan pada sebuah log file (dapat berupa file text atau table) dapat menjadi informasi yang sangat berguna untuk menganalisa bug yang mungkin terjadi. Terutama informasi yang menjelaskan apa yang terjadi sebelum, selama, dan sesudah bug terjadi. Untuk menghindari log file yang terlalu besar, yaitu dapat memisahkan log file terbagi menjadi log untuk komponen-komponen utama pada aplikasi. Jangan lupa untuk selalu mencatat waktu (timestamp) pada setiap record, dan pastikan dapat menghapus atau melakukan backup pada log file secara periodik.

xiii. Validasi

Tidak semua orang mematuhi aturan yang diterapkan pada aplikasi, karena itu programmer harus melakukan validasi untuk data yang dimasukkan oleh pengguna. Misalnya pada suatu form pendaftaran, sebaiknya melakukan validasi untuk input yang tidak boleh kosong (mandatory/required fields), melakukan pembatasan karakter, dan validasi huruf/angka yang diperlukan.

xiv. Mengenal Environment

Saat mengetikkan kode program, menjalankannya, atau melakukan debug pada program, maka terdapat environment yang berbeda-beda. Terdapat 3 environment yang umum dikenal, yaitu:

a. Design Time.

   Aplikasi yang dikerjakan dilakukan pada saat design time.

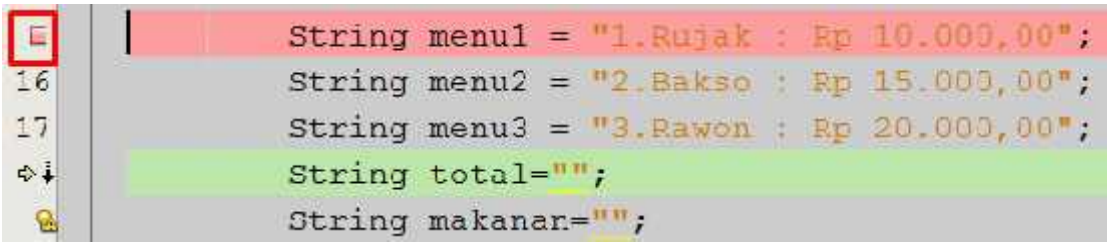
b. Run Time.

   Saat menjalankan aplikasi.

c. Break Mode.

Environment saat melakukan proses debug atau melihat kode program saat program tersebut dijalankan, dapat melihat alur program dan perubahan nilai pada variabel, sehingga dapat menelusuri kesalahan yang terjadi. Break Mode terletak diantara Design Time dan Run Time.

Untuk melakukan debugging pemrograman JAVA menggunakan editor Netbeans yaitu dengan melakukan double klik pada baris kode yang dicurigai terdapat kesalahan, sehingga akan muncul tanda kotak berwarna merah, seperti pada gambar 6.4 berikut ini

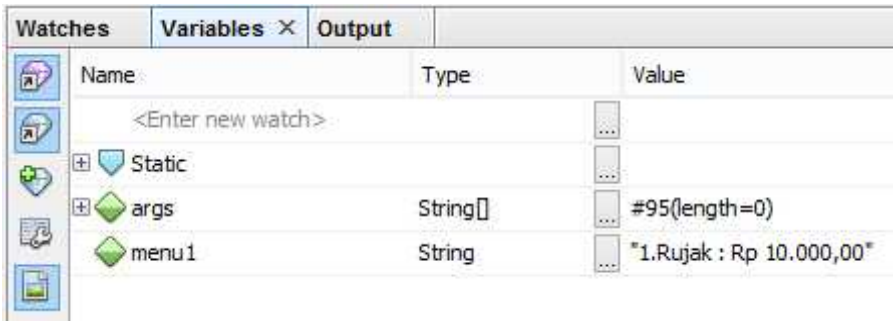


Gambar 6.4 Proses debugging

Kemudian pilih menu Debug → Step Into/Step Over atau dengan menggunakan shortcut F7 atau F8 seperti pada gambar 6.5 berikut ini

Gambar 6.5 Menu Debugging

Kemudian perhatikan tampilan output program dibagian bawah editor, dari tampilan tersebut dapat mengetahui perubahan nilai variabel, method, stack, watch dan lain-lain untuk membantu proses debugging seperti pada gambar 6.6 berikut ini



Gambar 6.6 Tampilan debugger

B. Keterampilan yang diperlukan dalam Mengkompilasi Program

1. Identifikasi jenis-jenis kesalahan pada program.
2. Mengoperasikan software IDE (Integrated Development Environment) untuk bahasa pemrograman JAVA seperti Netbeans, GEL, Eclipse dan lain-lain
3. Melakukan instalasi dan menjalankan software EXE4J.
4. Melakukan instalasi dan menjalankan software Inno Setup.

C. Sikap kerja yang diperlukan dalam Mengkompilasi Program

Harus bersikap secara:

1. Cermat dan teliti dalam menganalisis kode program.
2. Tekun dalam proses pemahaman sintaks.
3. Sesuai dengan kaidah-kaidah bahasa pemrograman.
4. Berpikir analitis serta evaluatif ketika melakukan trial and error.
5. Giat dalam mencari bugs dan memperbaikinya.

DAFTAR PUSTAKA

A. Buku Referensi

a. Eck, David dan Smith, Introduction to Programming Using Java, Geneva New York, Seventh Edition Version 7.0 August 2014
b. Downey and Mayfield, Think Java ,Green Tea Press, Needham USA, 2016
c. Wicaksono, Seri Penuntun praktis Dasar-dasar Pemrograman Java 2, Penerbit PT Elex Media Komputindo Kelompok Gramedia JAKARTA, 2002

B. Referensi Lainnya

a. Netbeans Documentaion, alamat url :https://netbeans.org/kb/, diakses pada tanggal 18 Februari 2018

DAFTAR ALAT DAN BAHAN

A. Daftar Peralatan/Mesin

No.	Nama Peralatan/Mesin	Keterangan
1.	Laptop, infocus, laserpointer	Untuk di ruang teori
2.	Laptop	Untuk setiap peserta
3.	Software JDK (Java Development Kit) 1.8	Untuk setiap peserta
4.	IDE Netbeans 7.0	Untuk setiap peserta
5.	Software EXE4J	Untuk setiap peserta
6.	Software Inno Setup	Untuk setiap peserta

B. Daftar Bahan

No.	Nama Bahan	Keterangan
1.	Kertas A4	Setiap peserta
2.	Alat Tulis	Setiap peserta

DAFTAR PENYUSUN

No.	Nama	Profesi
1.	Alun Sujjada, S.Kom.,M.T	1. Dosen STT Atlas Nusantara Malang Prodi Teknik Informatika 2. Asesor LSP STT Atlas Nusantara Malang 3. Anggota APTIKOM (Asosiasi Perguruan Tinggi Komputer)

kptk.or.id

instagram.com/lp3tk

fecebook.com/lp3tk

twitter.com/lp3tk

youtube.com/lp3tk