



Modul #5

Package

- PAKET -

DAFTAR ISI

A. PENDAHULUAN	1
1. Identitas Modul	1
2. Tujuan Pembelajaran	1
3. Deskripsi Materi	1
B. DASAR TEORI	2
1. Konsep Package	2
2. Hak Akses <i>Package</i>	2
3. <i>Built-in Package</i>	4
4. <i>User Defined Package</i>	5
5. <i>Sub Package</i>	5
6. <i>Import Package</i>	6
7. Keuntungan Menggunakan <i>Package</i>	7
C. TUGAS PRAKTIKUM	8
1. Program Cetak Kelas	8
2. Program Cetak Nama	10
D. EVALUASI	12
1. Program Toko Barang	12
2. Program Penghitung Luas dan Keliling Bangun Datar	12
3. Program Menampilkan Data Peserta Didik	12

A. PENDAHULUAN

Kompetensi Dasar:

Memahami konsep package pada Java

1. Identitas Modul

Mata Pelajaran : Pemrograman Berorientasi Objek

Kelas : XI

Judul Modul : Package (Paket)

2. Tujuan Pembelajaran

Setelah mempelajari bab ini diharapkan siswa akan mampu :

- a. Memahami konsep *package*.
- b. Memahami jenis-jenis *package*
- c. Menerapkan konsep *package* dalam membuat program

3. Deskripsi Materi

Modul 4 yang berjudul *Package* ini memaparkan tentang konsep *package* (paket) pada pemrograman Java untuk mengelompokkan berbagai hal (*class*, *interface*, *enum*, dsb) yang memiliki fungsi berkaitan serta memiliki akses dan pengelolaan yang berguna untuk mengorganisir beberapa buah *class/object*. Dengan *package* program dapat dibuat lebih rapi dan terhindar dari konflik karena penamaan yang sama.

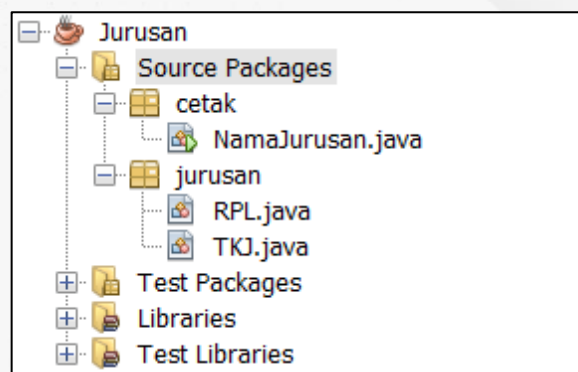
B. DASAR TEORI

1. Konsep Package

Package adalah fitur pada Java untuk mengelompokkan berbagai hal (*class*, *interface*, *enum*, *dsb*) yang memiliki fungsi berkaitan dan menyediakan proteksi akses dan pengelolaan *namespace*. *Package* dapat diibaratkan sebagai folder yang mengelompokkan berbagai jenis file, artinya 1 *package* adalah 1 folder di file system. *Package* juga dapat diibaratkan seperti hubungan antara folder dan subfolder, menu dan submenu, bab dan subbab, program, dan subprogram, menu, bab dan subbab, program, dan subprogram, *system* dan *subsystem*, dan lain sebagainya.

Dengan *package* program dapat dibuat lebih rapi dan terhindar dari konflik karena penamaan yang sama dan berguna untuk mengorganisir file dalam suatu *project* atau *library*. Nama *package* menggunakan huruf kecil semua (*lowercase*) dan mengikuti nama *domain* dengan susunan terbalik. Contoh : `com.facebook.katana`

Package mempengaruhi mekanisme hak akses ke kelas di dalamnya. Hal terpenting yang diperhatikan pada saat mendeklarasikan *package*, bahwa class tersebut harus disimpan pada suatu *directory* yang sama dengan nama *package*-nya. Alasan menggunakan *package* pada Java ialah untuk menghindari tabrakan nama kelas yang akan dibuat dengan nama kelas yang sudah ada. Selain itu, salah satu yang menjadi keuntungan menggunakan *package* adalah untuk mudahnya developer dalam hal mencari dan *manage* akses yang diberikan. Mengerti akan konsep dari *package* akan membantu mengelola dan menggunakan file yang disimpan didalam JAR (Java Archive).



Contoh *package* pada *project* Jurusan

2. Hak Akses Package

Package juga mempengaruhi mekanisme hak akses ke kelas-kelas di dalamnya.

a. Pengaruh *package* terhadap method `main()`

Kelas yang mengandung method `main()` memiliki syarat tidak berada dalam suatu *package*, dan *hierarki* posisi folder-nya di atas *package* yang di-*import*.

b. Membuat *Package*

Ada tiga langkah untuk membuat package.

- 1) Mendeklarasikan dan memberi nama *package*.
- 2) Membuat struktur dan nama direktori yang sesuai dengan struktur dan nama *package*.
- 3) Mengkompilasi kelas-kelas sesuai dengan *package*-nya masing-masing.

c. Mendeklarasikan dan memberi nama *package*

Deklarasi package harus diletakkan pada bagian paling awal (sebelum deklarasi *import*) dari *source code* setiap kelas yang dibungkus *package* tersebut. Bentuk umum deklarasi package:

```
package namaPackage;
```

Deklarasi tersebut akan memberitahukan kompilator, ke *library* manakah suatu kelas dikompilasi dan dirujuk. Berikut syarat nama package.

- 1) Diawali huruf kecil.
- 2) Menggambarkan kelas-kelas yang dibungkusnya.
- 3) Harus unik (berbeda dengan nama *package standard*).
- 4) Merepresentasikan *path* dari *package* tersebut.
- 5) Harus sama dengan nama direktorinya.

Contoh *package standard*.

- 1) java.lang (berisi kelas-kelas fundamental yang sering digunakan).
- 2) java.awt dan javax.swing (berisi kelas-kelas untuk membangun aplikasi GUI)
- 3) java.io (berisi kelas-kelas untuk proses input output).

d. Membuat struktur direktori

Pada langkah ini, buatlah direktori menggunakan *file manager* (di Windows menggunakan *explorer*) sesuai struktur *package* dari langkah sebelumnya. Kemudian tempatkan kelas-kelas tersebut ke direktori yang bersesuaian (mirip seperti menyimpan *file-file* ke dalam *folder*). *Package* dapat bersarang di *package* lain, sehingga dapat dibuat hierarki *package*.

Bentuk umum pernyataan

```
package namaPackage1[.nama Package2[.nama Package3]];
```

Contoh hirarki package di JDK:

```
package java.awt.image;
```

e. *Compile* dan *run* kelas dari suatu *package*

Selanjutnya masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi *byte code* (".class"). Artinya *package* tersebut siap digunakan.

f. Menggunakan package

Ada dua cara menggunakan suatu *package* yaitu sebagai berikut:

- 1) Kelas yang menggunakan berada dalam direktori (*package*) yang sama dengan kelas-kelas yang digunakan. Maka tidak diperlukan import.
- 2) Kelas yang menggunakan berada dalam direktori (*package*) yang berbeda dengan kelas-kelas yang digunakan. Maka pada awal *source code* di kelas pengguna harus mencantumkan:

```
import namaPackage>NamaKelas; atau  
import nama Package.*;
```

Contoh

```
import java.text.DecimalFormat;  
import javax.swing.*;
```

g. Setting Classpath

Path hierarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama *Classpath*. *Classpath* diset dengan aturan: berawal dari drive (C:\ atau D:\) sampai dengan satu tingkat sebelum kita mendeklarasikan *package*.

3. **Built-in Package**

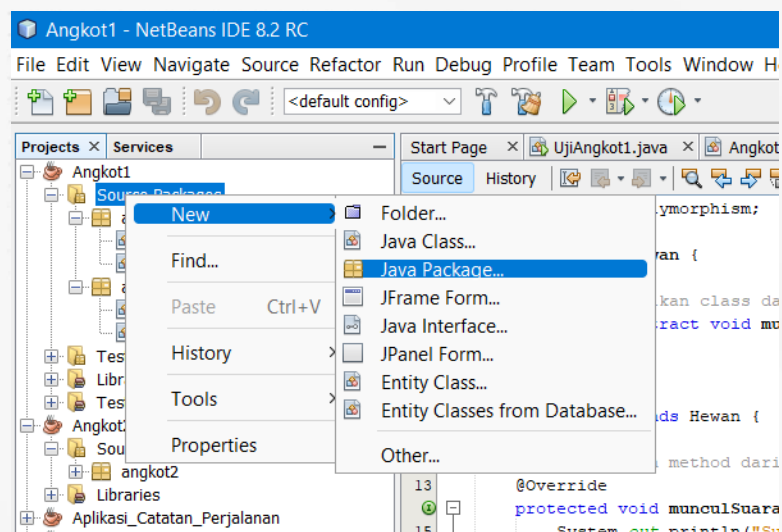
Built-in package adalah *package* bawaan yang telah disediakan oleh Java. Beberapa contoh package built-in yang sering digunakan

Paket	Keterangan
java.util	Menyediakan berbagai kelas untuk menangani berbagai kebutuhan umum (utility) seperti Scanner, Date, ArrayList, HashMap, Observer, Observable, dsb
java.io	Menyediakan berbagai kelas untuk menangani input / output seperti pembacaan file sampai penulisan kembali kedalam disk
java.sql	Menyediakan berbagai kelas untuk menangani koneksi database
javax.swing	Menyediakan berbagai kelas untuk implementasi user interface (UI). Di dalam javax.swing terdapat berbagai komponen GUI swing Java seperti button, input, label, dll
javax.swing.event	Paket ini berisi kelas-kelas dan interface yang memperbolehkan penanganan event untuk komponen grafis yang terletak di paket javax.swing.

java.lang	Paket ini berisi kelas-kelas dan interface yang diperlukan oleh banyak program Java. Paket ini diimpor oleh kompiler ke semua program Java secara otomatis.
java.applet	Paket ini berisi kelas-kelas Applet dan juga beberapa interface yang memperbolehkan interaksi applet dan browser serta untuk memainkan klip audio. Kelas javax.swing, JApplet digunakan untuk menetapkan applet yang menggunakan komponen GUI swing.
java.text	Paket ini berisi kelas-kelas dan interface yang memperbolehkan program Java untuk memanipulasi angka, tanggal, karakter dan juga string.
java.net	Paket ini berisi kelas yang memperbolehkan program untuk berkomunikasi melalui jaringan

4. **User Defined Package**

User defined package adalah package yang dibuat dan didefinisikan oleh user. Cara termudah membuat package pada IDE netbeans adalah dengan klik kanan source package > New > Java Package.



Contoh *user defined package* pada kelas Angkot1

Ketika kelas dibuat di dalam package maka IDE otomatis mendeklarasikan *package* dengan *keyword package*.

5. **Sub Package**

Sub-package adalah *package* yang dibuat di dalam *package* lain. Pada *sub-package* digunakan tanda titik (dot) untuk memisahkan *parent* dan *child package*. Contoh:

- Package `animalsystem.pets` artinya `pets` adalah sub-package dari `animalsystem` digunakan untuk mengelompokkan kelas-kelas dari binatang peliharaan.
- Package `animalsystem.pets` artinya `wild` adalah sub-package dari `animalsystem` digunakan untuk mengelompokkan kelas-kelas dari binatang buas

6. **Import Package**

Untuk mengakses sesuatu (*class*, *interface*, *enum*, dsb) diluar *package* harus dilakukan *import package*. Tanpa melakukan *import* obyek diluar package tidak dapat dikenali. Untuk melakukan *import* menggunakan keyword "import". Contoh:

```
package packagejava;

import animalsystem.pets.Cat;

public class Pet {

    public static void main(String[] args) {
        Cat cat = new Cat();
    }
}
```

Penjelasan program: Pada contoh potongan program diatas kelas `Pet` berada pada package `packagejava` ditunjukkan pada pernyataan `package packagejava`. Sedangkan kelas `Cat` berada pada kelas `animalsystem.pets` sehingga untuk menggunakan kelas `Cat` diperlukan `import package animalsystem.pets` dengan pernyataan `import animalsystem.pets.Cat`.

Pada program sebelumnya ditunjukkan cara mengimport kelas `Cat` pada package `animalsystem.pets`. Jika diperlukan untuk mengimport semua (kelas `Cat`, kelas-kelas lain, *interface*, dll) yang ada pada package `animalsystem.pets`, dapat digunakan tanda asterisk (`*`) yang artinya semua.

```
package packagejava;
import animalsystem.pets.*;

public class Pet {
    public static void main(String[] args) {
        Cat cat = new Cat();
        Dog dog = new Dog();
    }
}
```


Penjelasan program: Pernyataan `import animalssystem.pets.*` artinya mengimport semua (*class*, *interface*, dll) yang ada pada package `animalssystem.pets`

7. Keuntungan Menggunakan *Package*

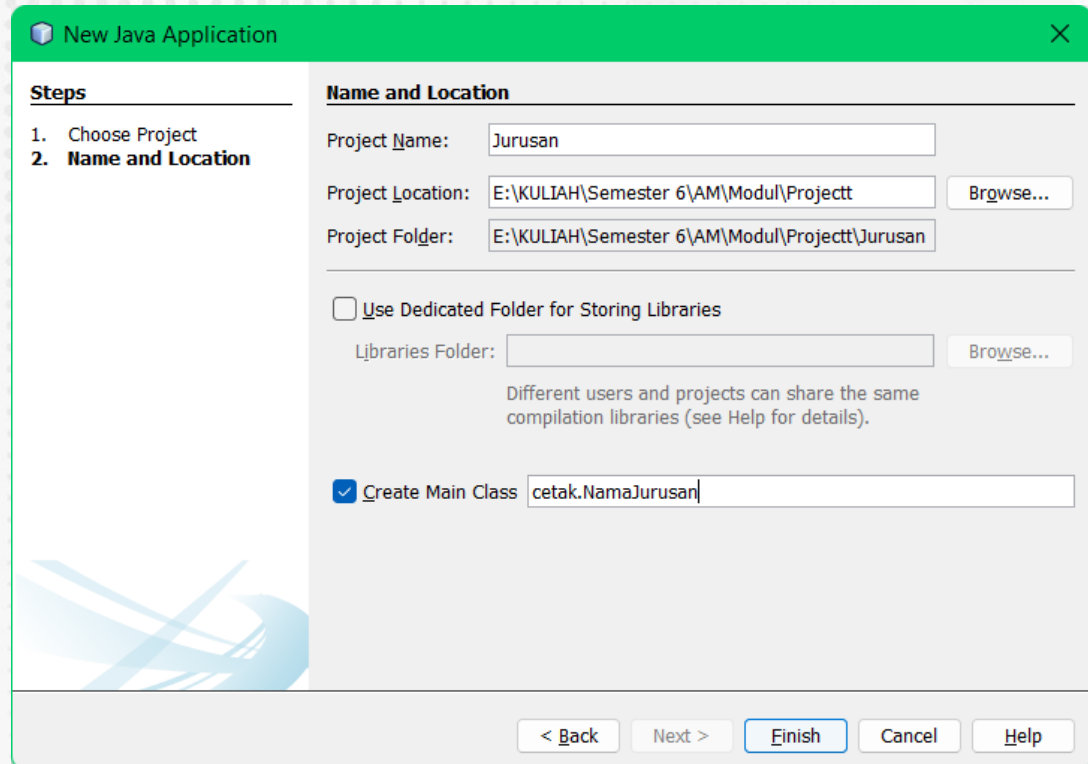
Programmer Java terkadang membuat kelas sendiri yang mempunyai nama sama dengan kelas yang sudah ada di Java API tanpa sengaja. Hal ini menyebabkan konflik penamaan kelas. Untuk mengatasi masalah terkait penamaan kelas, letakkan kelas yang mempunyai nama sama ke dalam paket berbeda dan mengakses kelas tersebut beserta dengan nama paketnya. Kelas-kelas Java yang mempunyai kemiripan fungsi seharusnya diletakkan pada paket yang sama sehingga akan mempermudah penempatan dan pendistribusian.

Paket juga berfungsi memberi proteksi pada kelas dan interface yang ada di dalamnya. Sebagai contoh, kelas yang dideklarasikan dengan kata kunci *private*, hanya dapat diakses oleh kelas-kelas lain yang berada dalam paket yang sama.

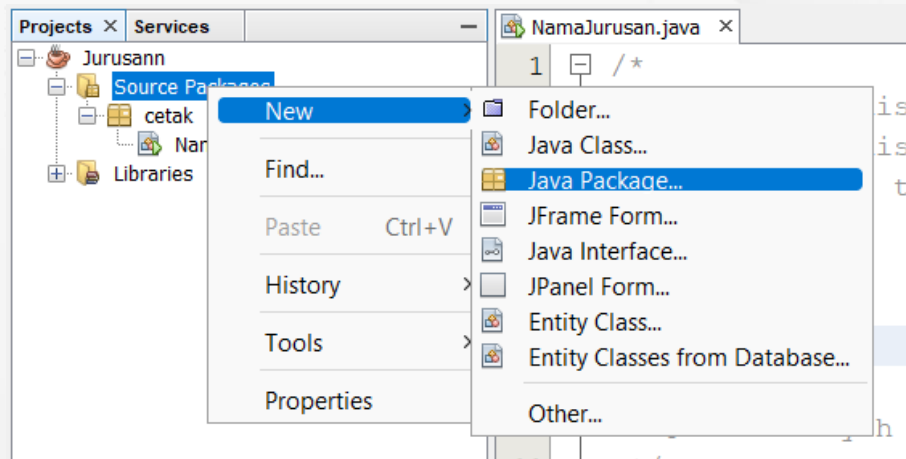
C. TUGAS PRAKTIKUM

1. Program Cetak Kelas

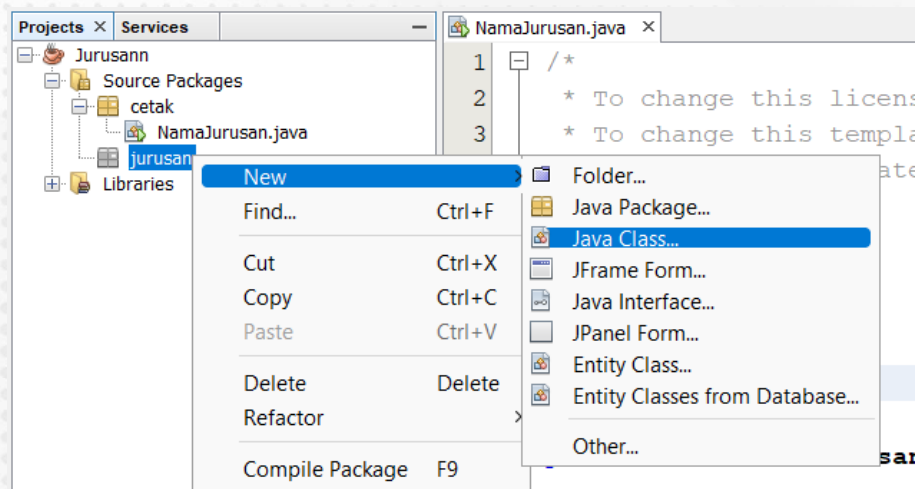
Buatlah Project Baru di Netbeans dengan nama Jurusan, kemudian untuk Main Class bernama cetak>NamaJurusan



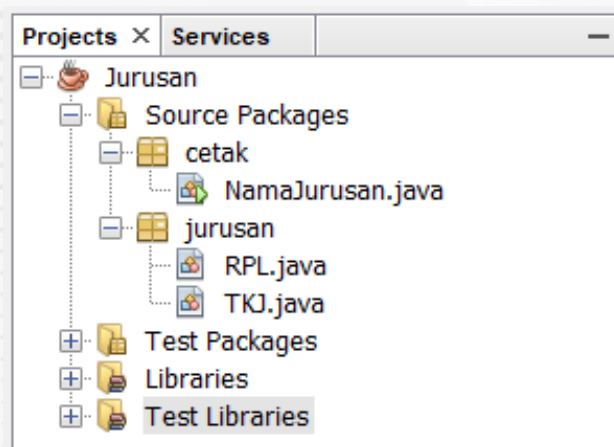
Kemudian buatlah sebuah *Java Package* baru dengan nama jurusan



Setelah itu buatlah dua buah *Java Class* baru dengan nama TKJ dan RPL

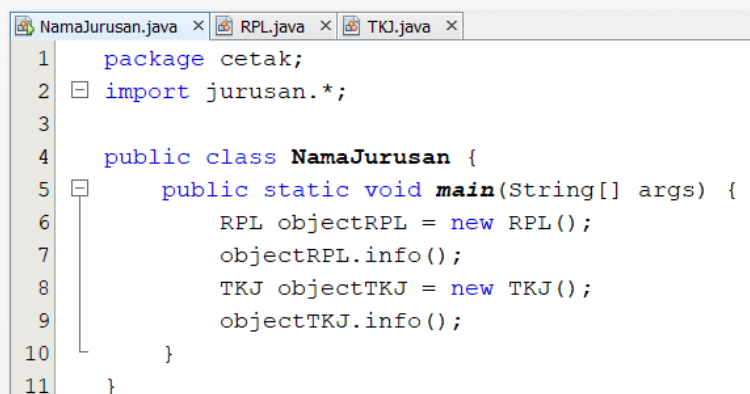


Sehingga struktur direktori seperti berikut:

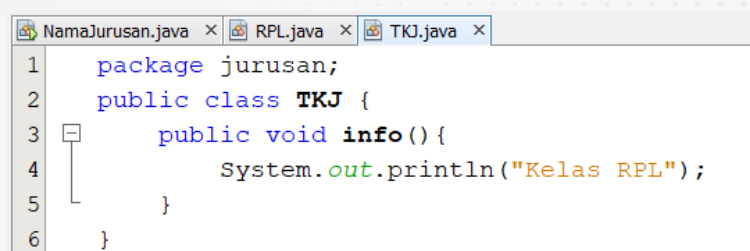


Tulislah dan simpan kode program di bawah ini:

Kode program *class* NamaJurusan . java



Kode program *class* RPL . java



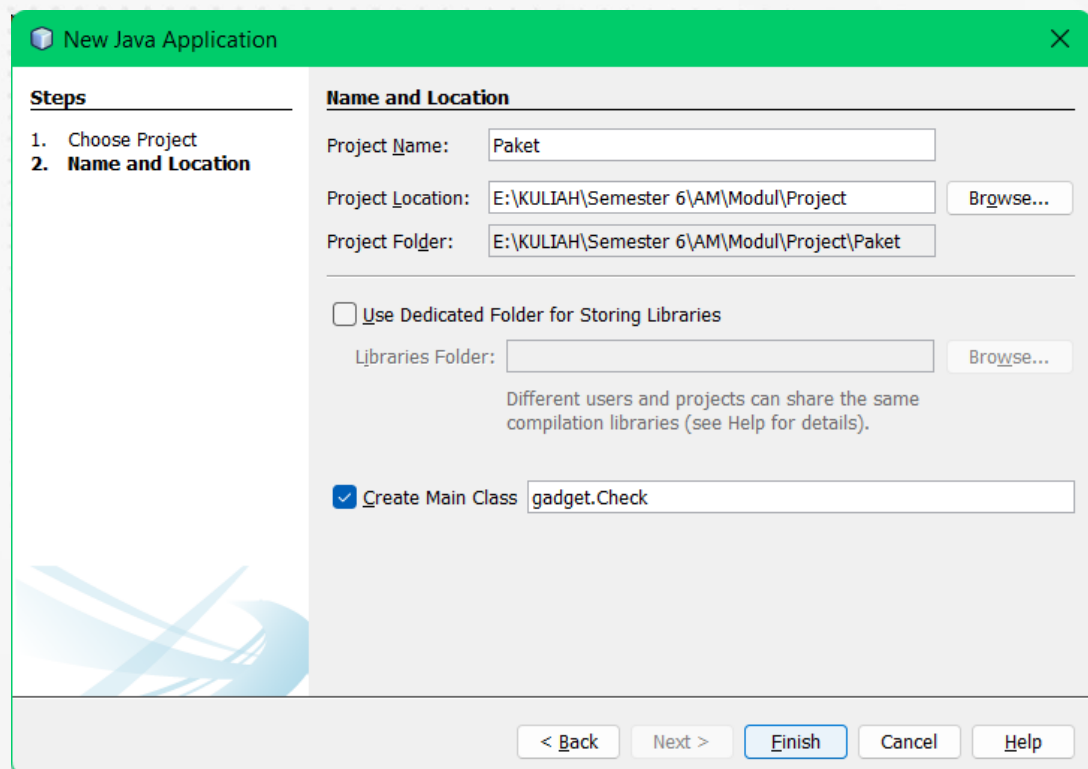
Kode program class TKJ.java

```
NamaJurusan.java x RPL.java x TKJ.java x
1 package jurusan;
2 public class RPL {
3     public void info() {
4         System.out.println("Kelas TKJ");
5     }
6 }
```

- a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- b. Berikan penjelasan terkait jalannya program ini!

2. Program Cetak Nama

Buatlah Project Baru di Netbeans dengan nama Paket, kemudian untuk Main Class bernama gadget.Check



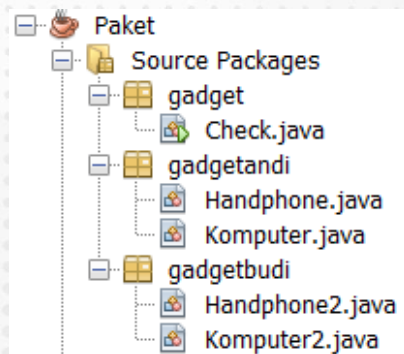
Kemudian buatlah sebuah *Java Package* baru dengan nama gadgetandi

Setelah itu buatlah dua buah *Java Class* baru dengan nama Komputer dan Handphone

Kemudian buatlah sebuah *Java Package* baru dengan nama gadgetbudi

Setelah itu buatlah dua buah *Java Class* baru dengan nama Komputer2 dan Handphone2

Sehingga struktur direktori seperti berikut:



Tuliskan dan simpan kode program di bawah ini:

Kode program class `Handphone.java`

```

1 package gadgetandi;
2 public class Handphone {
3     public String cekInfo() {
4         return "File Video ini berasal dari Handphone Andi";
5     }
6 }

```

Kode program class `Komputer.java`

```

1 package gadgetandi;
2 public class Komputer {
3     public String cekInfo() {
4         return "File Corel Draw ini berasal dari Komputer Andi";
5     }
6 }

```

Kode program class `Handphone2.java`

```

1 package gadgetbudi;
2 import gadgetandi.*;
3 public class Handphone2 {
4     public String cekInfo() {
5         return "File Audio ini berasal dari Handphone Budi";
6     }
7 }

```

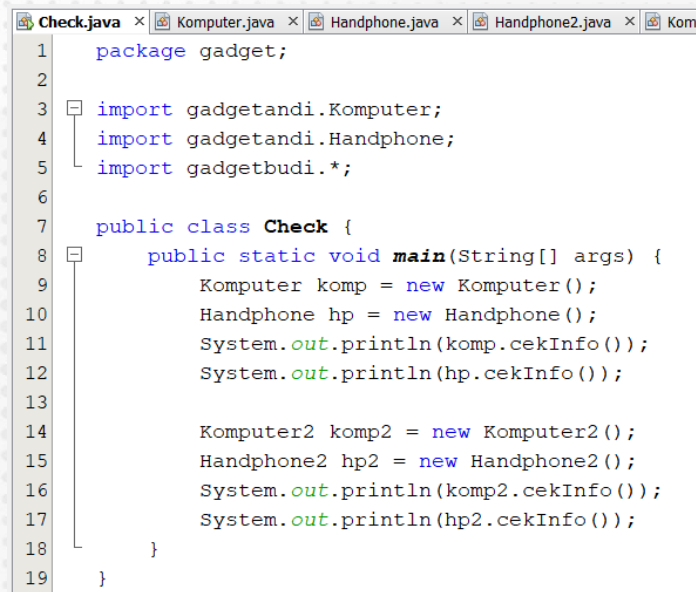
Kode program class `Komputer2.java`

```

1 package gadgetbudi;
2 import gadgetandi.*;
3 public class Komputer2 {
4     public String cekInfo() {
5         return "File Photoshop ini berasal dari Komputer Budi";
6     }
7 }

```

Kode program class `Check.java`



```

1  package gadget;
2
3  import gadgetandi.Komputer;
4  import gadgetandi.Handphone;
5  import gadgetbudi.*;
6
7  public class Check {
8      public static void main(String[] args) {
9          Komputer komp = new Komputer();
10         Handphone hp = new Handphone();
11         System.out.println(komp.cekInfo());
12         System.out.println(hp.cekInfo());
13
14         Komputer2 komp2 = new Komputer2();
15         Handphone2 hp2 = new Handphone2();
16         System.out.println(komp2.cekInfo());
17         System.out.println(hp2.cekInfo());
18     }
19 }

```

- Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!

D. EVALUASI

1. Program Toko Barang

- Buatlah sebuah program untuk toko penjualan barang yang terdiri minimal 3 buah *Java Package* berisi *package* untuk data transaksi, data harga barang, dan data stok dengan mengaplikasikan fitur *Java Package*!
- Berikan penjelasan terkait jalannya program yang Anda buat!

2. Program Penghitung Luas dan Keliling Bangun Datar

- Buatlah sebuah program untuk mencari luas bangun datar yang terdiri minimal 2 buah *Java Package* berisi *package* untuk rumus bangun datar (minimal 3 buah bangun datar seperti segitiga, persegi, lingkaran, dsb) dan *package* untuk menghitung luas dan keliling bangun datar dari inputan user dengan mengaplikasikan fitur *Java Package*!
- Berikan penjelasan terkait jalannya program yang Anda buat!

3. Program Menampilkan Data Peserta Didik

- Buatlah sebuah program untuk menampilkan data peserta didik yang terdiri minimal 3 buah *Java Package* berisi *package* untuk data pribadi peserta didik, data orang tua, dan data asal sekolah dengan mengaplikasikan fitur *Java Package*!
- Berikan penjelasan terkait jalannya program yang Anda buat!