

IT-652 – Pemrograman Berorientasi Aspek

Pengantar Aspect Oriented Programming

Ramos Somya

Pengantar AOP

- OOP → mendukung modularity program.
- Java memecah komponen-komponennya menjadi objek-objek terpisah yang saling berinteraksi.
- Class yang telah dibuat oleh suatu program OOP, dapat digunakan oleh program OOP lain secara langsung.
- Kelemahan: Karena berorientasi obyek, maka akan banyak sekali terjadi perubahan secara dinamis.
- Sebuah aplikasi harus bisa menangani perubahan ke depannya tanpa harus merubah keseluruhan program.

Contoh Kasus

- Misal terdapat sebuah class PembuatLaporan:

```
PembuatLaporan
-----
buatLaporanBulanan()
buatLaporanTahunan()
```

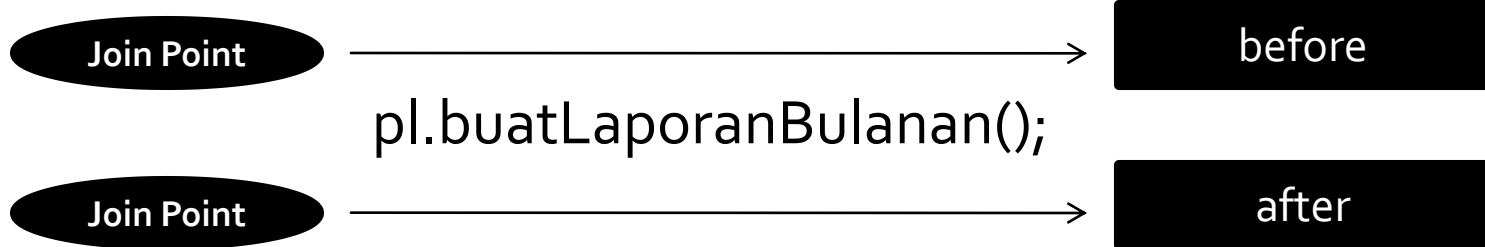
- Misal di suatu waktu ada perubahan pada proses bisnis.
- Fitur PembuatLaporan diakses oleh pihak yg berbeda.
- Misal LaporanBulanan oleh Admin, Laporan Tahunan oleh Bagian keuangan.



- Dengan AOP, kita bisa menyisipkan kode program tambahan tanpa merubah class awal yang sudah dibuat.
- Proses ini dinamakan CROSSCUTTING.
- Dalam OOP sulit untuk melakukan crosscutting, karena satu sama lain saling terhubung, jika 1 diubah, akan berpengaruh pada yang lainnya.



- **PembuatLaporan pl = new PembuatLaporan();**





point cut x(); // mengacu pada joint point tertentu
Bisa menambahkan advice (statement yang akan dilakukan sebelum/setelah joint point tertentu tercapai).

before(): x() ←

.....

.....

pl.buatLaporanBulanan(); // saat dipanggil lakukan call

Implementasi AOP pada Java

- Untuk implementasi AOP pada Java, kita menggunakan AspectJ.
- AspectJ menambahkan kepada Java dengan suatu konsep, yang disebut join point (dari crosscutting concerns) dan menambahkan beberapa konstruktor baru seperti pointcut, advice, inter-type declaration, dan aspect.
- Pointcut dan advice secara dinamic mempengaruhi program flow.

Instalasi

Yang dibutuhkan:

- IDE → pakai Eclipse
- `ajdt_2.1.0_for_eclipse_3.6.zip`.

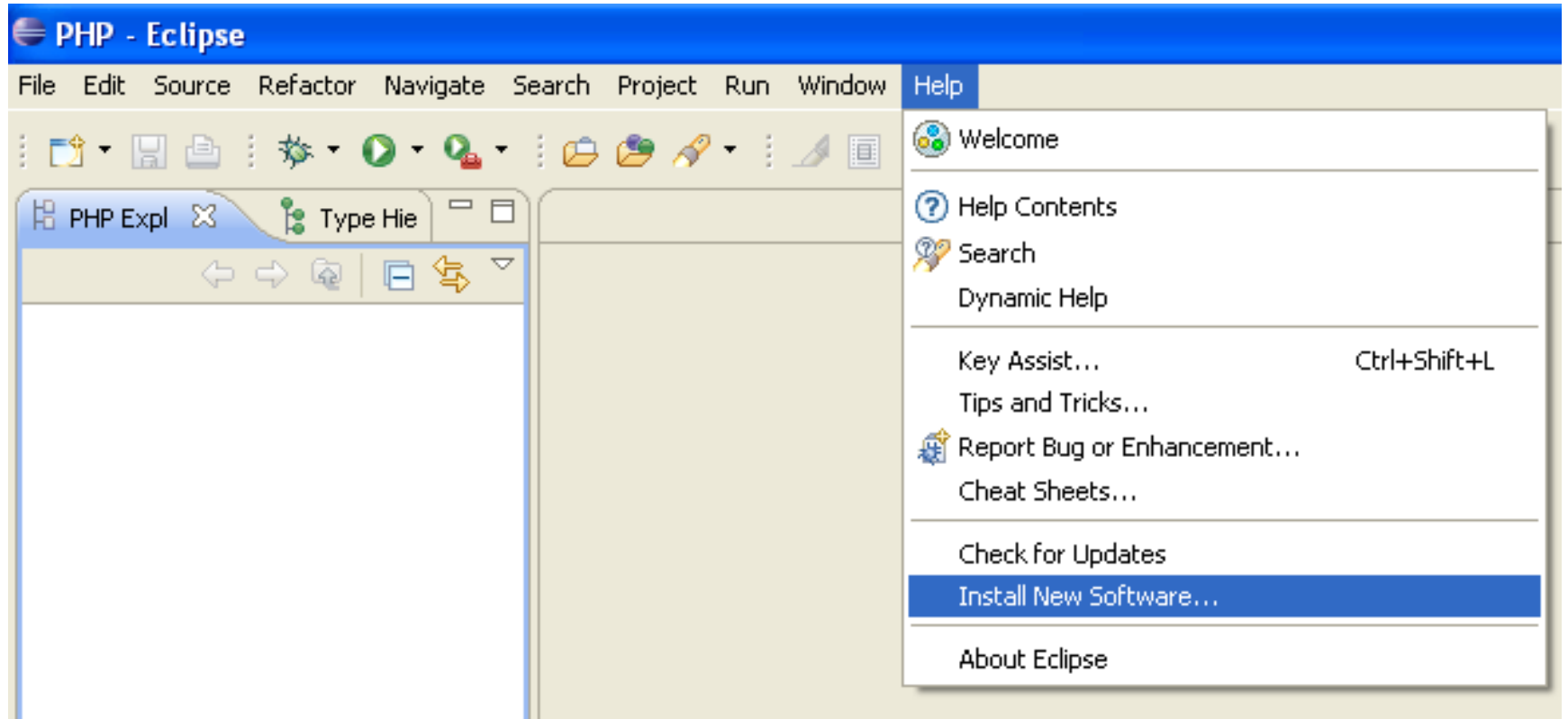
Langkah-langkah

- Instal dulu Eclipse.



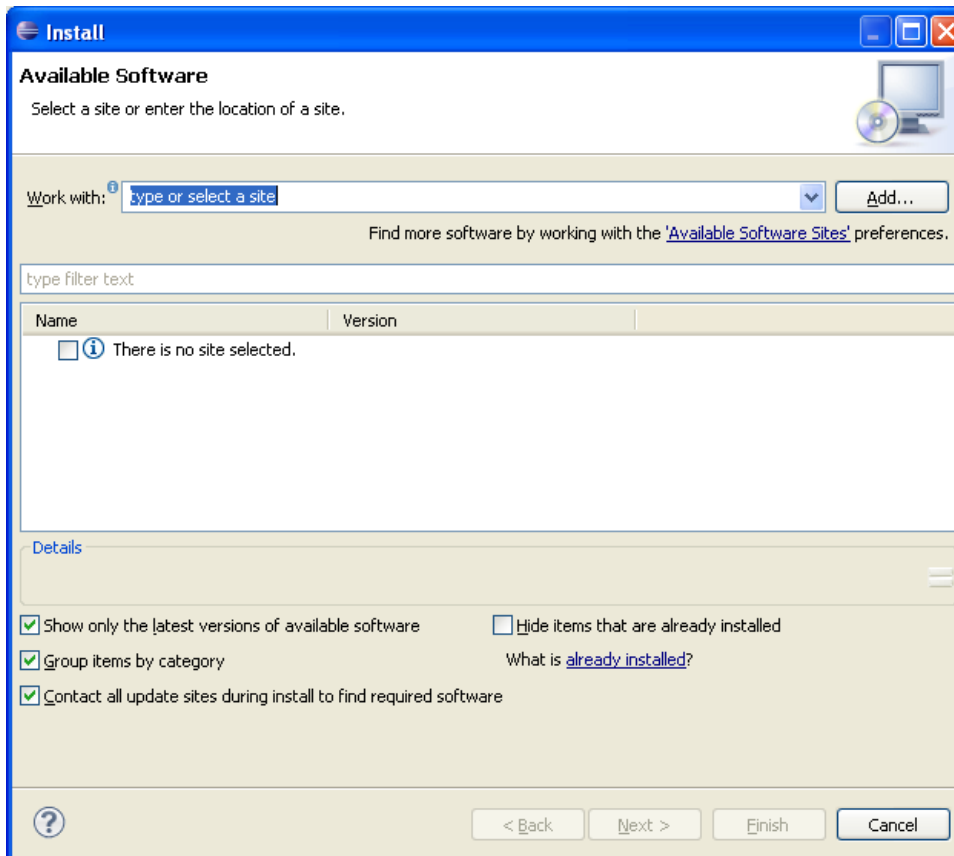
eclipse.exe

- Tambahkan plugin `ajdt_2.1.0_for_eclipse_3.6.zip`



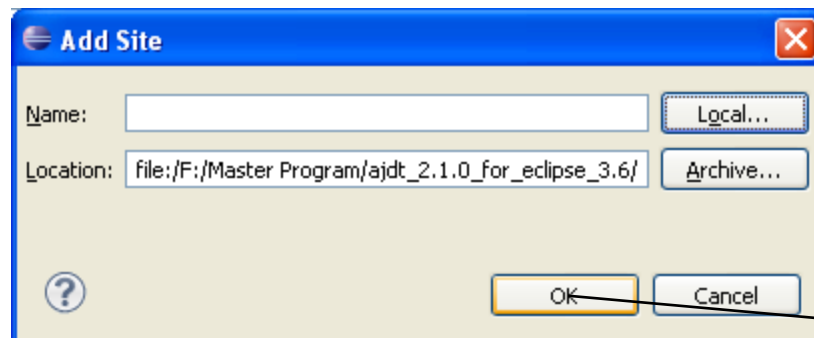
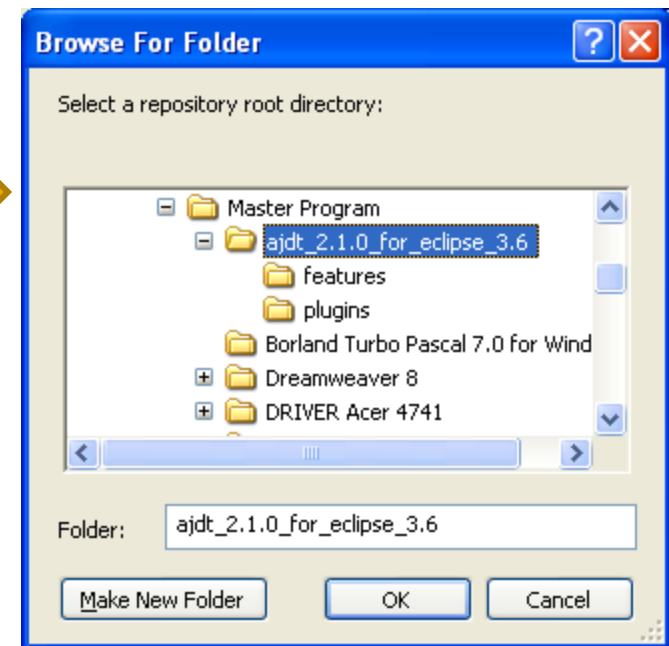
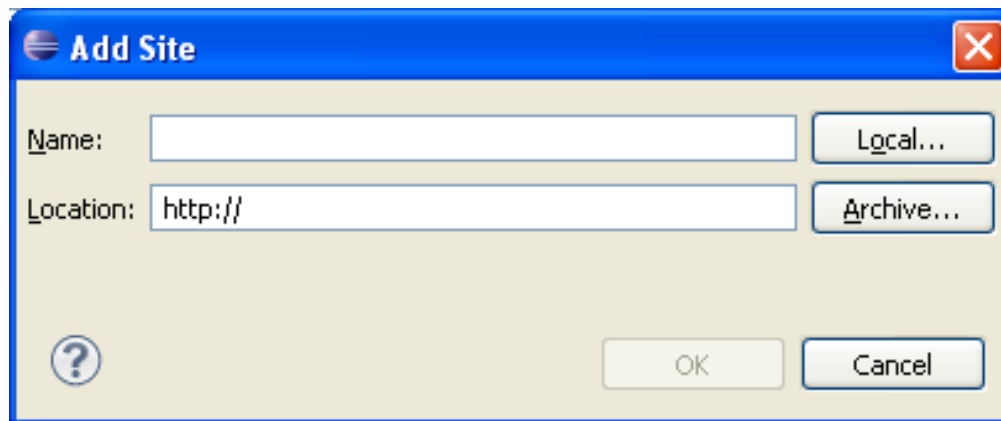


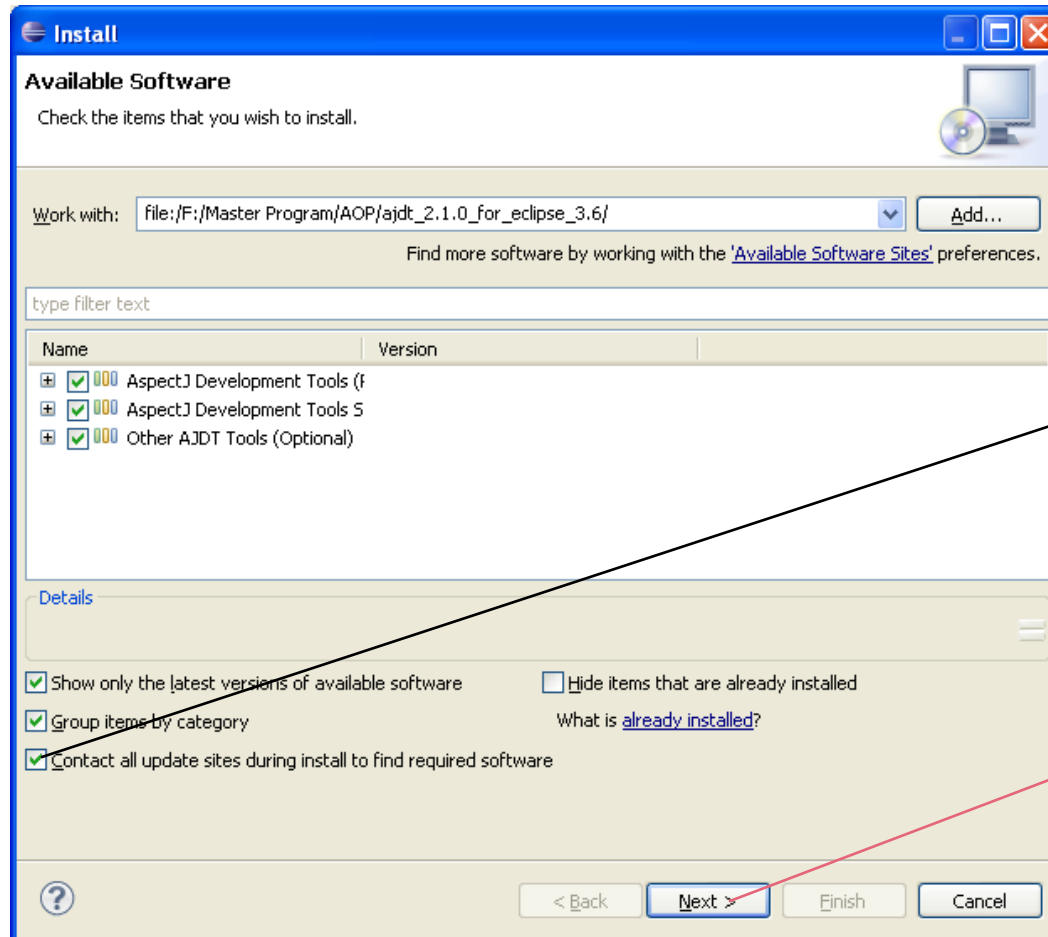
■ Klik Add





■ Pilih Local



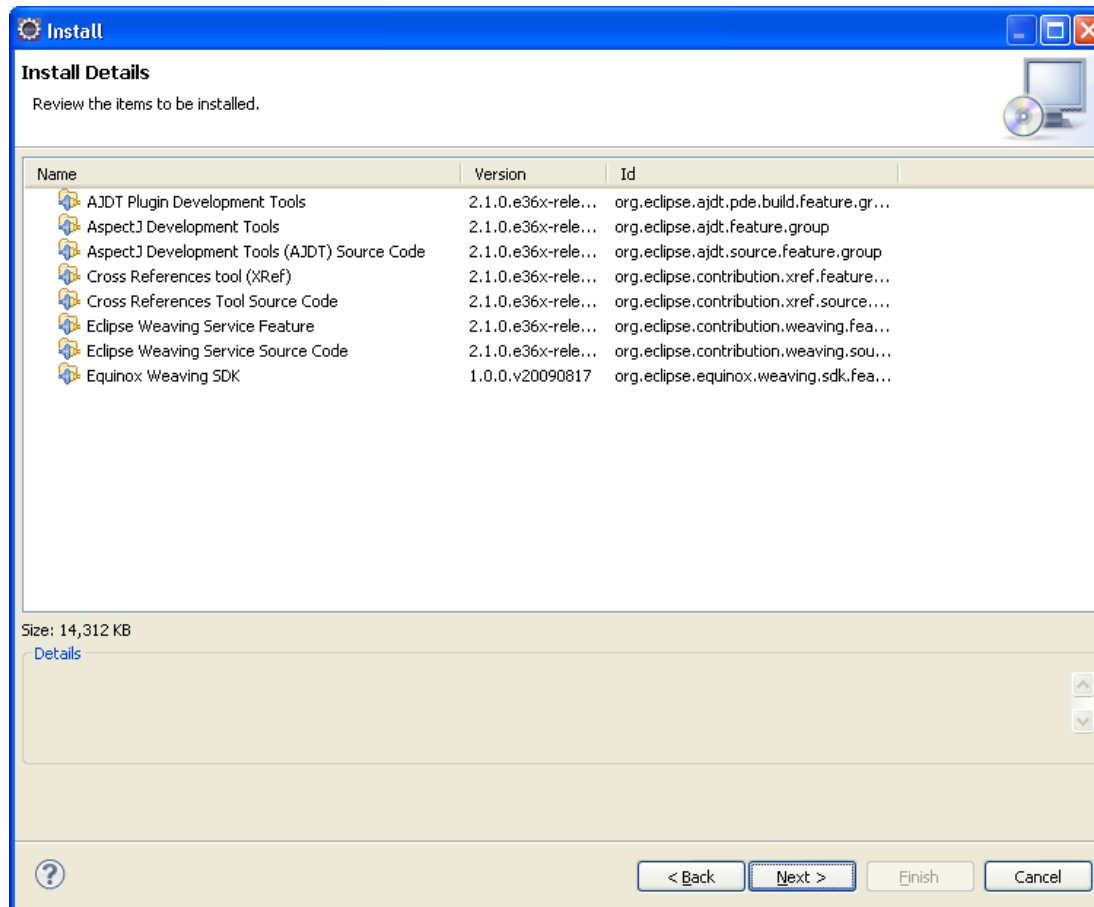


uncheck

Next

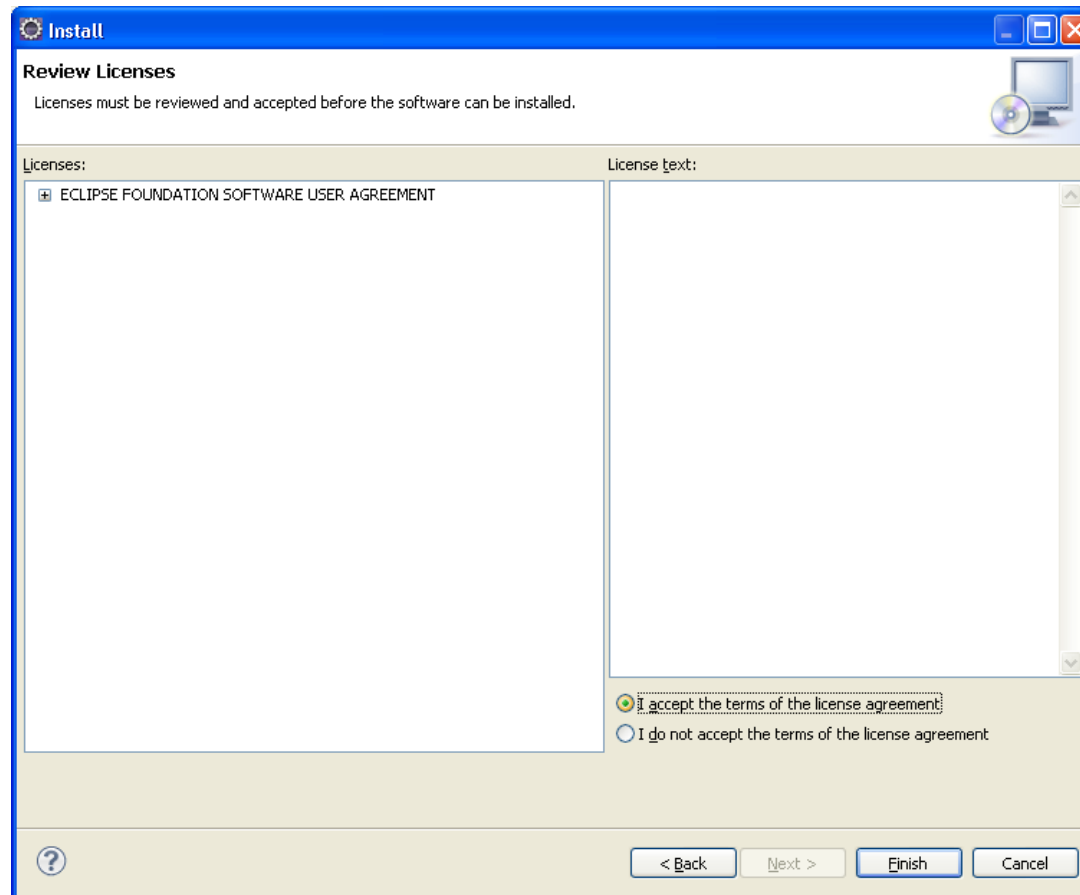


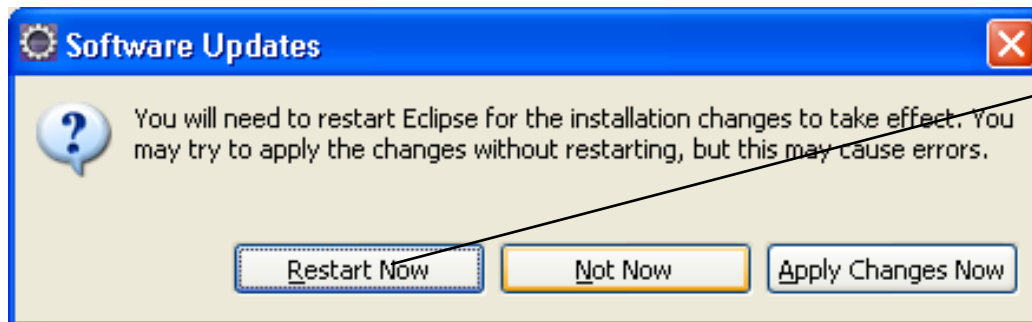
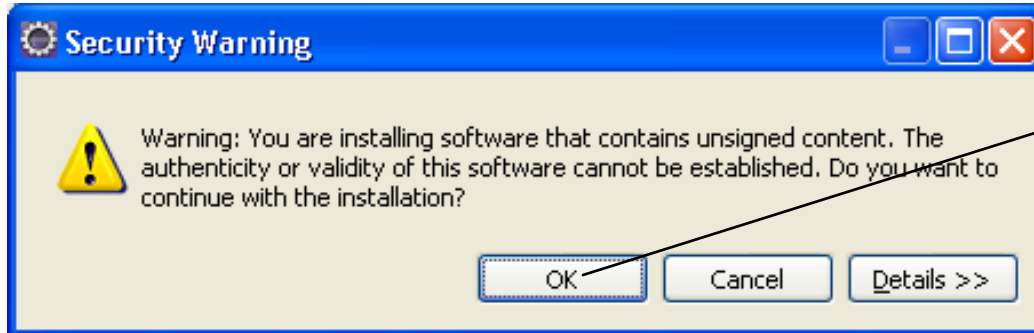
■ Next





■ I accept - Finish





Contoh Sederhana

- Class TestHello

```
public class TestHello {  
  
    public void cetak(){  
        System.out.println("Dicetak oleh TestHello di metode cetak");  
    }  
  
    public void print(){  
        System.out.println("Dicetak oleh TestHello di metode print");  
    }  
  
}
```



■ Class MainHello

```
public class MainHello {  
    public static void main(String[] args) {  
        TestHello th = new TestHello();  
        th.cetak();  
        th.print();  
    }  
}
```



- **Aspect: SetelahCetak**

```
public aspect SetelahCetak {  
  
    pointcut namaPointCut() : call(void TestHello.*());  
  
    after() : namaPointCut(){  
        System.out.println("kode ini dijalankan setelah metode cetak jalan");  
    }  
  
    before() : call(void TestHello.print()){  
        System.out.println("Dijalankan sebelum metode print dipanggil");  
    }  
}
```

Coba Lagi

```
public class TestHello {  
  
    public void cetak(String str, int nilai)  
    {  
        System.out.println("Metode cetak di class TestHello diberi  
parameter String : " + "" + str + " dan int : " + nilai);  
    }  
}
```



```
public aspect SetelahCetak {
```

```
    pointcut namaPointcut(String s, int i) :  
        call(void TestHello.cetak(String, int)) && args(s, i);
```

```
    after(String s, int i) : namaPointcut(s, i)
```

```
{
```

```
    System.out.println("Kode ini dijalankan setelah metode cetak jalan");
```

```
    System.out.println("String param = " + s);
```

```
    System.out.println("int param = " + i);
```

```
}
```

```
}
```



```
public class mainHello {  
    public static void main(String[] args)  
    {  
        TestHello th = new TestHello();  
        th.cetak("tes string", 100);  
    }  
}
```

Tugas Take Home

- **Buatlah class Shape dengan atribut nama, panjang dan lebar.**
- **Buat setter dan getter-nya dan buat method info untuk mencetak nama, panjang dan lebar.**
- **Buat class Square yang merupakan turunan dari class Shape.**
- **Buat method hitungLuas pada class Square untuk menghitung luasnya (panjang*lebar).**
- **Tambahkan Aspect untuk membuat proteksi supaya user tidak bisa mengeset nilai panjang dan lebar jika nilainya > 100.**
- **Buat class Hitung, buat obyek dari class Square, set panjang dan lebarnya lalu tampilkan info dan hitung luasnya.**

Ketentuan

- Tugas individu.
- Kumpulkan pada hari Selasa, 4 Oktober 2011 pukul 14.00 - 15.00 di kantor dalam bentuk hardcopy.
- Sertakan kode program dan output-nya dan beri penjelasan untuk kode programnya.
- **Copy paste nilai = 0**