Riza Satria Perdana, S.T., M.T.

Teknik Informatika - STEI ITB

Inheritance Feature

Pemrograman Berorientasi Objek

Mengakses Super Class

```
public class Superclass {

    public void printMethod() {
        System.out.println("Printed in Superclass.");
    }
}

public class Subclass extends Superclass {

    // overrides printMethod in Superclass
    public void printMethod() {
        super.printMethod();
        System.out.println("Printed in Subclass");
    }
    public static void main(String[] args) {
        Subclass s = new Subclass();
        s.printMethod();
    }
}
```

Constructor pada Inheritance

- Konstruktor kelas basis akan dieksekusi terlebih dahulu dari kelas turunannya
- Konstruktor kelas turunan bisa memilih konstruktor kelas basis yang akan dieksekusi

Overriding dan Overloading

- Overriding (redefine): kelas turunan membuat ulang (mendefinisikan ulang) method dari kelas basis
- Overloading: kelas turunan membuat lagi method dengan nama yang sama dari kelas basis tapi dengan parameter berbeda

Composition vs Inheritance

- “has a” vs “is a”
- Pada komposisi: suatu kelas disusun atas objek-objek dari kelas-kelas lain
- Pada turunan: suatu kelas dibuat berdasarkan kelas yang lain

Istilah Seputar Inheritance

- Multiple Inheritance: kelas yang diturunkan dari 2 atau lebih kelas basis
- Repeated Inheritance: multiple inheritance yang kelas basisnya merupakan turunan dari kelas yang sama (repeated base class)

Final Class dan Final Method

- **Final Method:** method yang tidak bisa di-redefine (override) di kelas turunannya
- **Final Class:** kelas yang tidak bisa diturunkan lagi

Abstract Class dan Method

- Kelas Abstrak: kelas yang tidak bisa diinstansiasi
- Method Abstrak: method yang dideklarasikan tapi tidak ada implementasinya dan akan diimplementasikan oleh kelas turunan
- Kelas yang mempunyai method abstrak harus manjadi kelas abstrak

Contoh

```
public class ChessAlgorithm {
    enum ChessPlayer { WHITE, BLACK }
    ...
    final ChessPlayer getFirstPlayer() {
        return ChessPlayer.WHITE;
    }
    ...
}

public abstract class GraphicObject {
    // declare fields
    // declare non-abstract methods
    public abstract void draw();
}
```

Contoh ...

```
public abstract class GraphicObject {
    private int x, y;
    ...
    public void moveTo(int newX, int newY) {
        ...
    }
    public abstract void draw();
    public abstract void resize();
}
```

```
public class Circle extends GraphicObject {
    public void draw() {
        ...
    }
    public void resize() {
        ...
    }
}

public class Rectangle extends GraphicObject {
    public void draw() {
        ...
    }
    public void resize() {
        ...
    }
}
```

Terima Kasih