

Inheritance & Polymorphism

Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009–10 Semester

INHERITANCE

True Or False?

- ▶ Mobile Phone is a kind of a Telephone
- ▶ Mobile Phone inherits Telephone
- ▶ Mobile Phone is derived from Telephone
- ▶ Mobile Phone is a subclass of Telephone
- ▶ Mobile Phone extends Telephone
- ▶ Mobile Phone is a child class of Telephone
- ▶ Telephone is the super class of Mobile Phone
- ▶ Telephone is the base class of Mobile Phone

ALL TRUE!

Syntax

- ▶ Assume class MySuper:

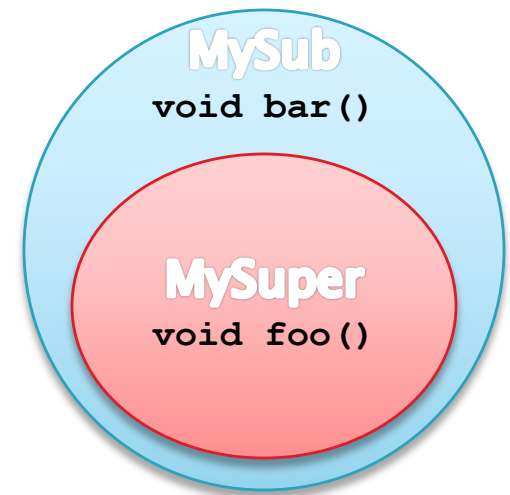
```
class MySuper {  
    public void foo() {}  
}
```

- ▶ Define MySub as a derived class of MySuper:

```
class MySub extends MySuper {  
    public void bar() {  
        // call foo() method of MySuper  
    }  
}
```

- ▶ Addressing a member implemented in Super:

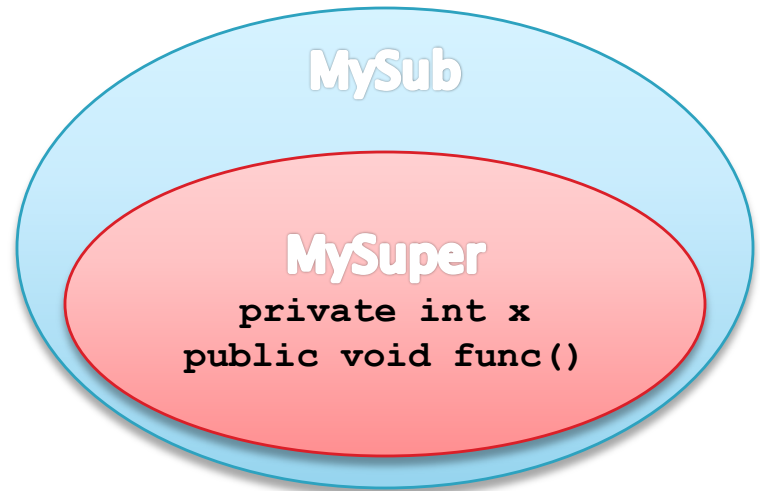
```
class MySub extends MySuper {  
    public void bar() {  
        super.foo();  
    }  
}
```



Rule #1: Use of Super

- ▶ The derived class includes all members of the super class as they are implemented
- ▶ The non-private members can be used directly

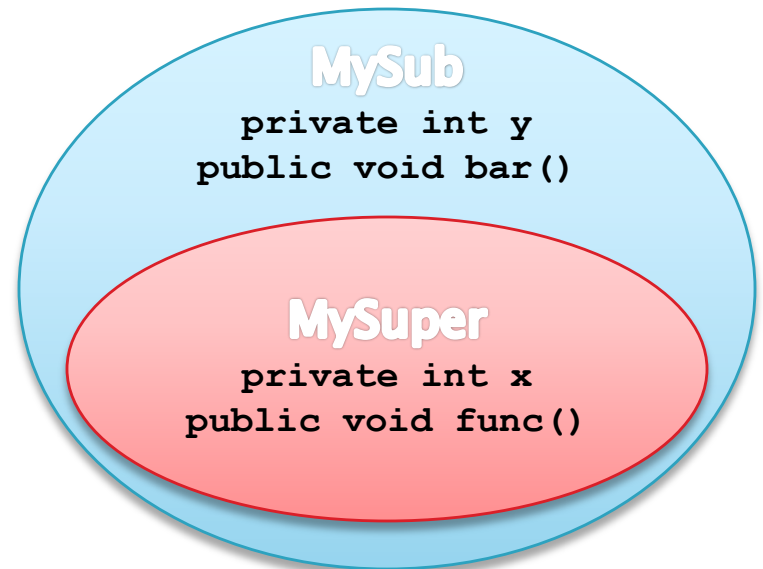
```
class MySuper {  
    private int x;  
    public void func() {}  
}  
class MySub extends MySuper {  
}  
class Irrelevant {  
    public void foo() {  
        MySuper supr = new MySuper();  
        supr.func();  
        MySub sub = new MySub();  
        sub.func();  
    }  
}
```



Rule #2: Extending Super

- ▶ The derived class can extend the functionality of the parent class

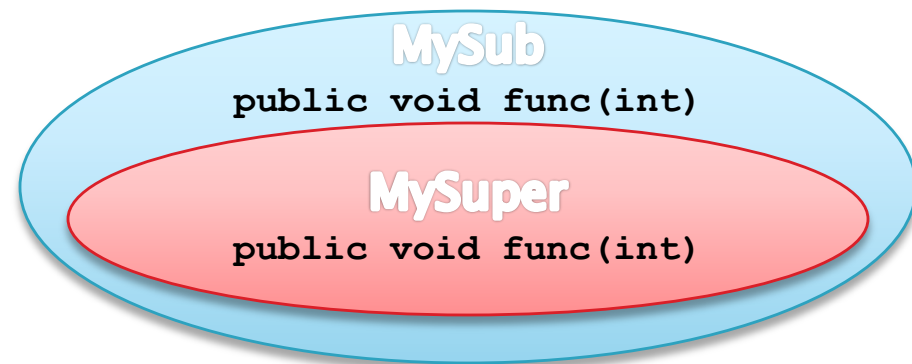
```
class MySuper {  
    private int x;  
    public void func() {}  
}  
class MySub extends MySuper {  
    private int y;  
    public void bar() {}  
}  
class Irrelevant {  
    public void foo() {  
        MySuper supr = new MySuper();  
        MySub sub = new MySub();  
        supr.bar();  
        sub.bar();  
    }  
}
```



Rule #3: Overriding Super

- ▶ The derived class can **override** the functionality of the parent class
- ▶ Overridden has precedence over super
- ▶ Overriding fields or static methods is called “hiding” and is **not recommended**

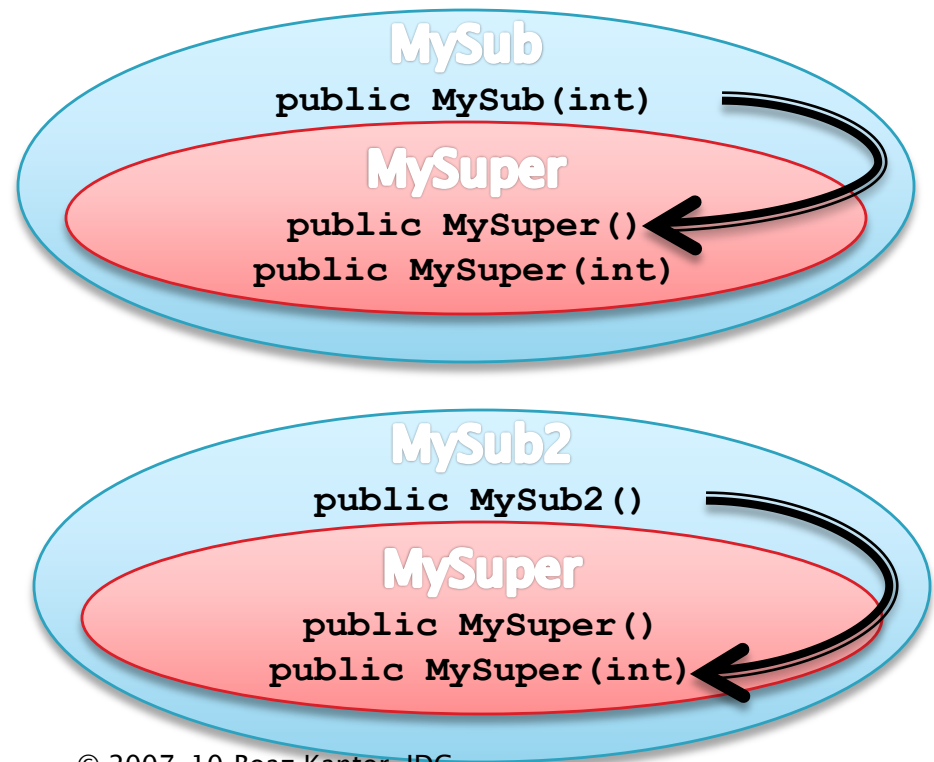
```
class MySuper {
    public void func(int x) {
        System.out.println("x = " + x);
    }
}
class MySub extends MySuper {
    public void func(int x) {
        System.out.println("x = " + x*x);
    }
}
class Irrelevant {
    MySuper supr = new MySuper();
    MySub sub = new MySub();
    supr.func(5);        // prints "x = 5"
    sub.func(5);        // prints "x = 25"
}
```



Rule #4: Constructors

- ▶ Constructors are not derived
- ▶ Any constructor must first call any super constructor using `super(parameters)`
- ▶ If no such call exists, `super()` is implicitly called

```
class MySuper {
    public MySuper() {}
    public MySuper(int x) {}
}
class MySub extends MySuper {
    public MySub(int x) {
    }
}
class MySub2 extends MySuper {
    public MySub2() {
        super(10);
    }
}
```



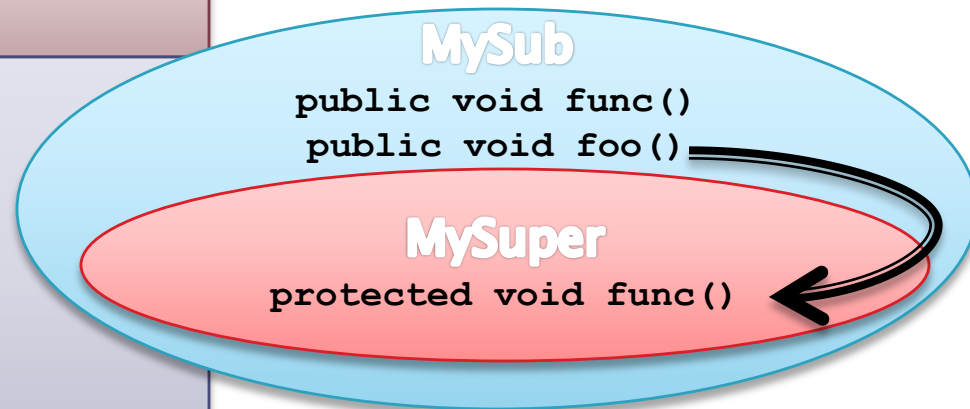
Rule #5: Protected

- ▶ Anything “**protected**” is visible to other classes of the same package and **derived classes**, no matter in which package
- ▶ You can increase visibility, but can't reduce it

```
package pack1;
public class MySuper {
    protected void func() { }
}

package pack2;
public class MySub extends pack1.MySuper {
    public void func() { }
    public void foo() {
        super.func();
    }
}

public class Relevant {
    public void f() {
        pack1.MySuper sup = new pack1.MySuper();
        sup.func();           // COMPILATION ERROR
        MySub sub = new MySub();
        sub.foo();           // SABABA
        sub.func();         // ALSO SABABA
    }
}
```



Rule #6: Polymorphism

▶ Assume :

- `MySuper var = new MySub();`

▶ Then:

- MySub must inherit MySuper
- var is of type MySub:
 - `var instanceof MySub` is true
 - `var instanceof MySuper` is true
- Compiler thinks it's of type MySuper
- To allow MySub view of var, cast:
 - `((MySub) var).mysubSpecificMethod();`

Rule #7: java.lang.Object

- ▶ If a class does not explicitly inherit any other class, it **automatically inherits** `java.lang.Object`
- ▶ Important Object methods to override:
 - `protected Object clone()`
 - `public boolean equals(Object obj)`
 - `public String toString()`
 - ...more (`getClass()`, `finalize()`, etc...)

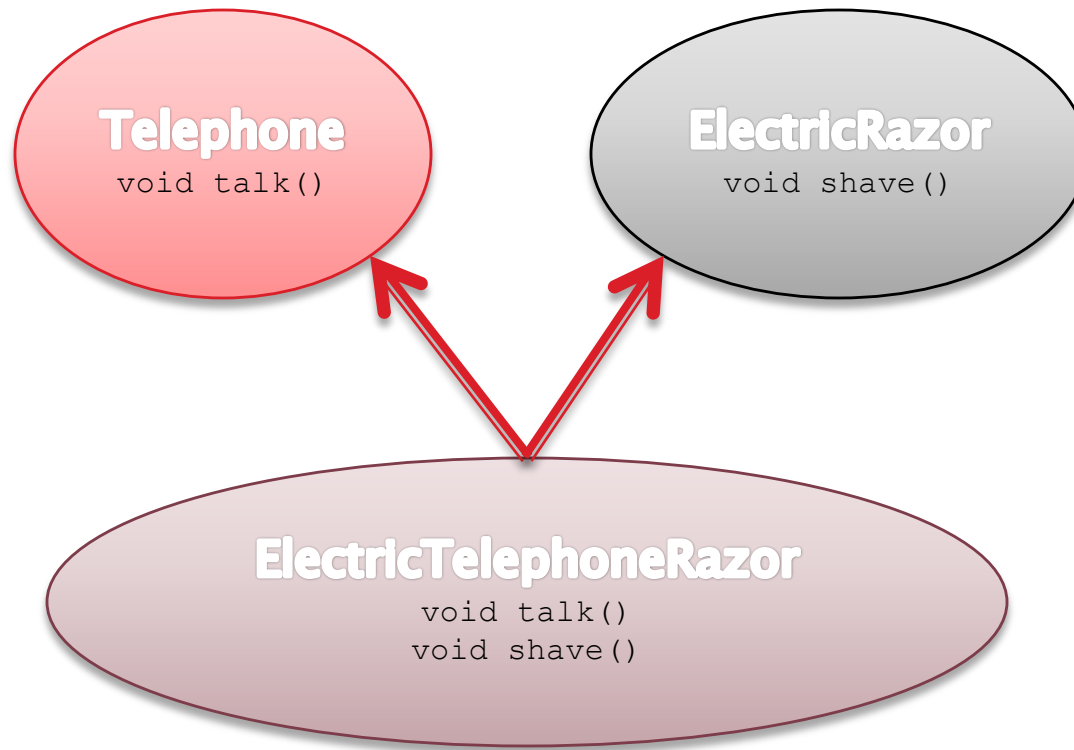
Rule #8: final

- ▶ Classes, fields and methods can be “**final**”
- ▶ The above cannot be overridden

```
class MySuper {  
    public final void func(int) {}  
}  
class MySub extends MySuper {  
    public void func(int) {}           // COMPILATION ERROR  
}
```

Rule #9: Multiple Inheritance

- ▶ There is no multiple inheritance in Java!



Rule #10 (good practices)

- ▶ Avoid protected instance variables
 - Always prefer private variables, with public or protected getters/setters
- ▶ Any method called directly or indirectly by a constructor should be final
- ▶ Always use `instanceof` before casting a reference

Telephone



Dial

1. Pick up the handset
2. Wait for a line
3. For each digit:
 1. Rotate the dial

Correspond

1. Talk to the microphone
2. Listen to the ear piece

Hang up

1. Place handset on base

Mobile Phone



Dial

1. Click the green button
2. Wait for a line
3. For each digit:
 1. Click its button

Correspond

1. Talk to the microphone
2. Listen to the ear piece

Hang up

1. Click the red button

Recharge

1. Hang up
2. Place phone in base unit
3. Wait until green light is lit

Inheritance & Polymorphism

Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009–10 Semester