Object Oriented Programming

Kookmin University
Department of Computer Science

Announcements

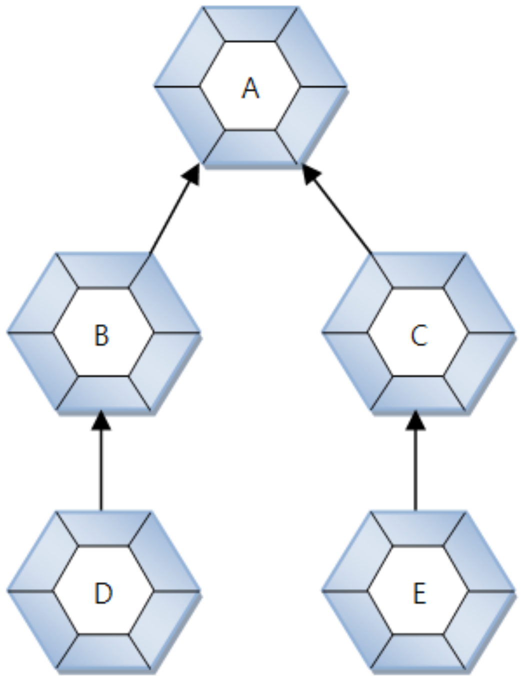
- Final Exam
  - Dec. 4th 17:00 ~
  - In English with a helper for English interpretation
- The second live coding test
  - Nov. 29th 17:00 ~

Polymorphism and Type Casting

- Polymorphism (다형성) allows allocation of multiple types in another type
    - It is tightly connected with inheritance and parent-child relationship
- Type casting for a class
    - A child class object can be automatically converted (type casting) to a parent object type
    - A parent class object cannot be converted to a child class type
    - Remember CellPhone class and SmartPhone class.

Automatic Type Conversion

- After a class type conversion to a parent class type, only the members from the parent class can be accessed



```
B b = new B();
C c = new C();
D d = new D();
E e = new E();
```

```
A a1 = b;  (가능)
A a2 = c;  (가능)
A a3 = d;  (가능)
A a4 = e;  (가능)

B b1 = d;  (가능)
C c1 = e;  (가능)

B b3 = e;  (불가능)
C c2 = d;  (불가능)
```

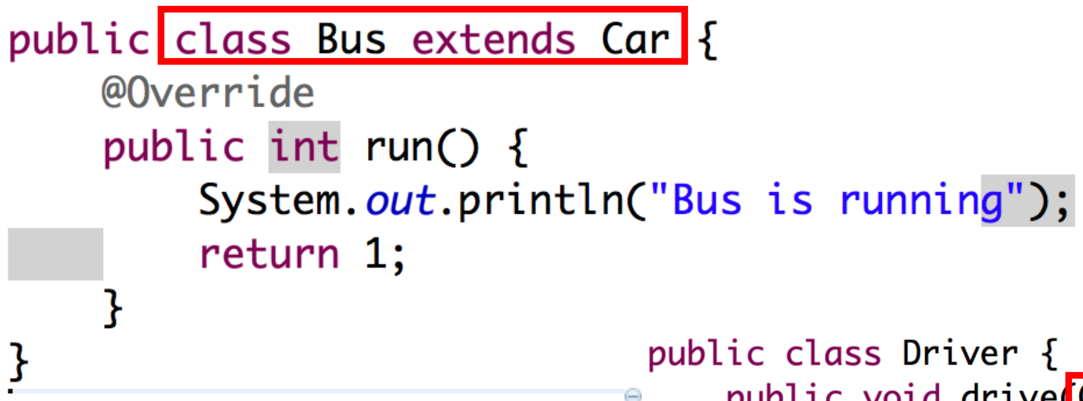
이것이 자바다 P. 308

Field Polymorphism

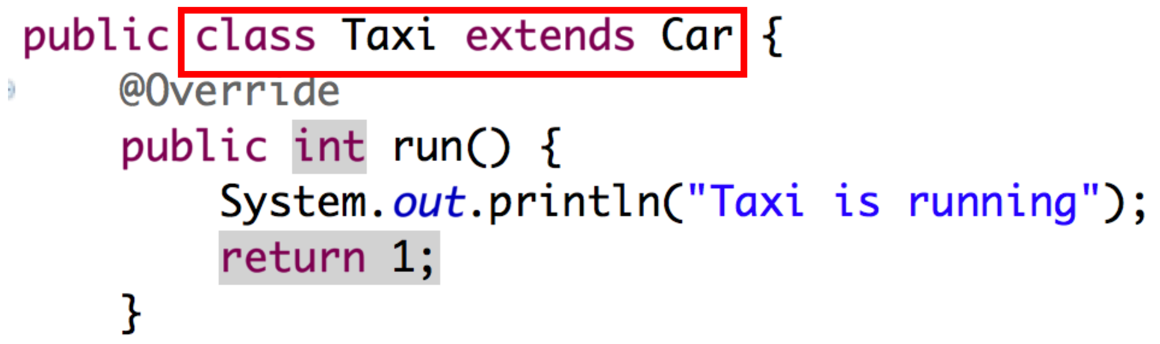
- Field object can have multiple forms if they share a same parent class type
- Imagine a car that has four tires. The tires do not have to manufactures from a same class (or company) assuming all tires meet standards
- Example : Car

Polymorphism for Method Arguments

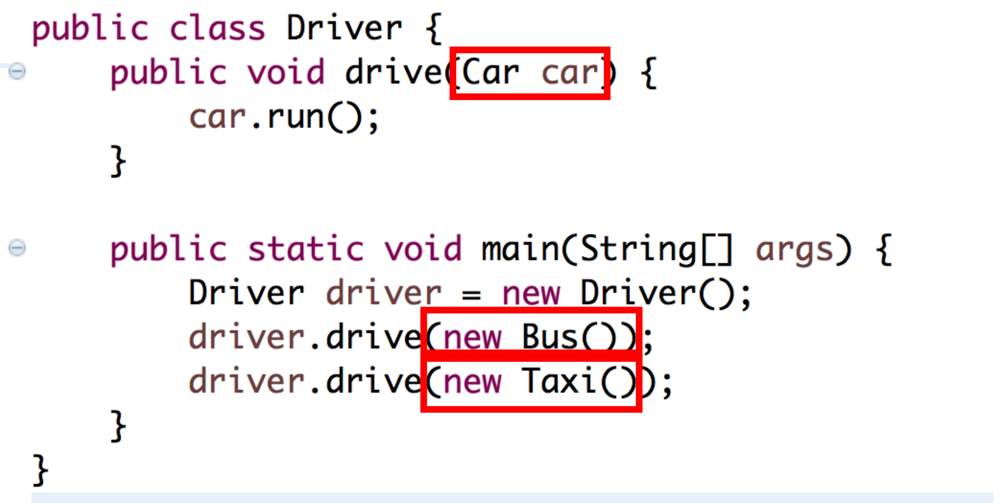
- Polymorphism applies well to class fields
- Polymorphism can also applies to method arguments



```java
public class Bus extends Car {
    @Override
    public int run() {
        System.out.println("Bus is running");
        return 1;
    }
}
```



```java
public class Taxi extends Car {
    @Override
    public int run() {
        System.out.println("Taxi is running");
        return 1;
    }
```



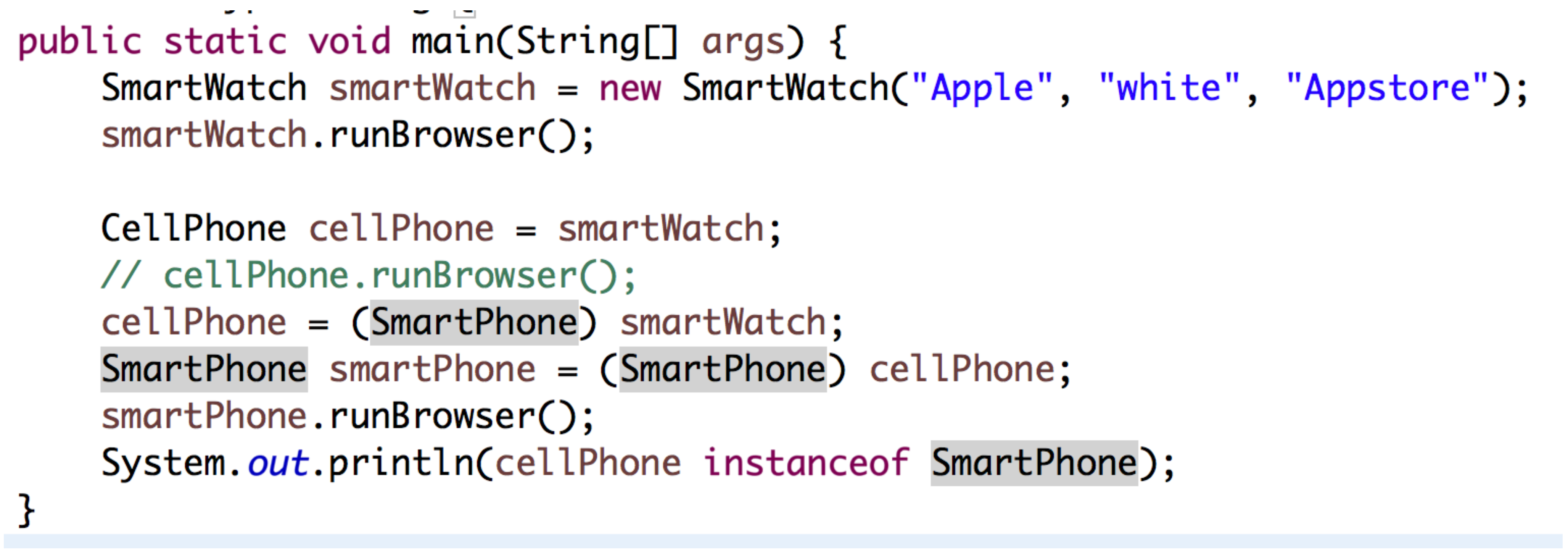
```java
public class Driver {
    public void drive(Car car) {
        car.run();
    }

    public static void main(String[] args) {
        Driver driver = new Driver();
        driver.drive(new Bus());
        driver.drive(new Taxi());
    }
}
```

Type Casting from Parent to Child Class

- In case an object created from a child class definition is declared with a parent class type, it can be converted to a child class type that generally has more fields/methods
  - EX: CellPhone cellPhone = new SmartPhone();
  - The cellPhone can be converted to SmartPhone() object
- To type casting to a child class, use parenthesis with a child class name
  - EX: ParentClass parentClass = new ChildClass();
  - ChildClass childClass = (ChildClass)parentClass ;
- isinstanceof allows to check if a given class is instance of an object

TypeCasting Example

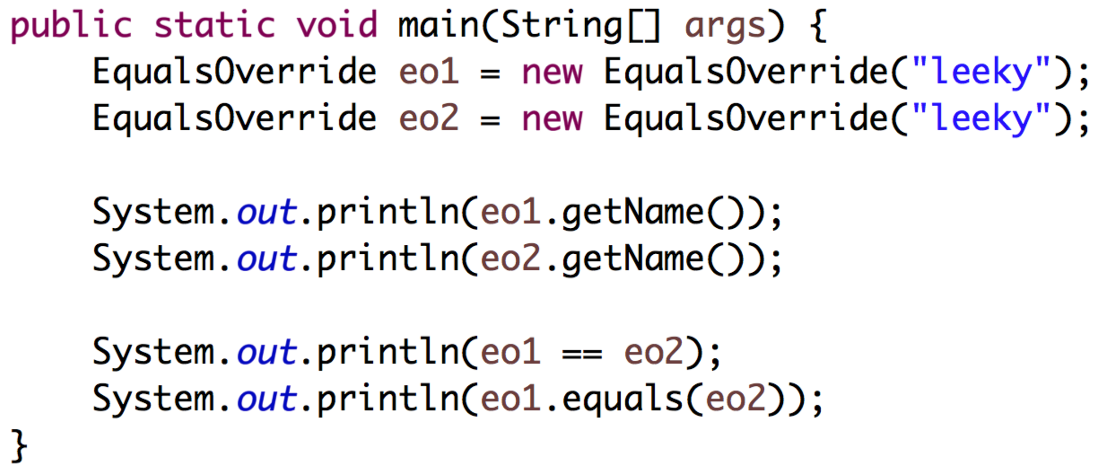
- TypeCasting.java



```java
public static void main(String[] args) {
    SmartWatch smartWatch = new SmartWatch("Apple", "white", "Appstore");
    smartWatch.runBrowser();

    CellPhone cellPhone = smartWatch;
    // cellPhone.runBrowser();
    cellPhone = (SmartPhone) smartWatch;
    SmartPhone smartPhone = (SmartPhone) cellPhone;
    smartPhone.runBrowser();
    System.out.println(cellPhone instanceof SmartPhone);
}
```

Overriding toString and equals

- By overriding toString, we can replace how an object is converted to a String.
- How about checking if two objects are the same
- Can you guess the outcome of the below example?



```java
public static void main(String[] args) {
    EqualsOverride eo1 = new EqualsOverride("leeky");
    EqualsOverride eo2 = new EqualsOverride("leeky");

    System.out.println(eo1.getName());
    System.out.println(eo2.getName());

    System.out.println(eo1 == eo2);
    System.out.println(eo1.equals(eo2));
}
```

Override equals method

- Method signature
  - public boolean equals(Object obj)
  - After getting an obj to compare from the argument, it needs to check if the type of obj is same
  - instanceof method checks the class hierarchy
  - child instanceof parent : true
  - parent instanceof child : false
- Question
  - How does String.equals works?

References

- 이것이자바다 – 한빛미디어 2015