

10.Inheritance

Sakib Abrar

CSE

Bangladesh University of Engineering & Technology

sakib.cgbs@gmail.com

September 10, 2020

Overview

Inheritance Basics

DRY principle

Java "extends" keyword

Java "super" keyword

Parent Class variable can hold child class data

Constructor Examples

Default Constructor

Access Modifiers in Java

Inheritance Basics

- ▶ Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.
- ▶ The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

DRY : Don't Repeat Yourself

In coding try not to repeat yourself.

If you're getting a lot of similarities in two class then try to use inheritance.

This is like a parent child relationship.

Childs inherit some traits from their parent.

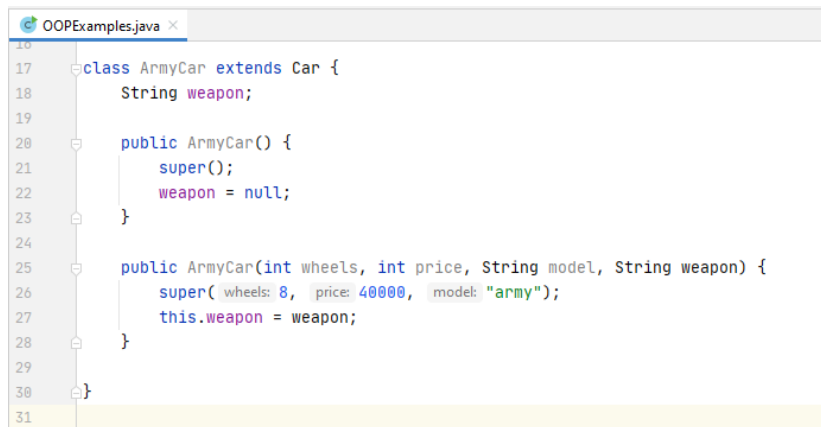
A Child class can inherit properties from parent class.

Java "extends" keyword

```
OOPExamples.java x
1 class Car {
2     int wheels;
3     int price;
4     String model;
5
6     public Car() {
7
8     }
9
10    public Car(int wheels, int price, String model) {
11        this.wheels = wheels;
12        this.price = price;
13        this.model = model;
14    }
15 }
16
17 class SuperCar extends Car {
18     String racingEngine;
19 }
20
21 class ArmyCar extends Car {
22     String weapon;
23 }
24
```

Java "super" keyword

You can use super to call parent class constructor in child class constructor.



```
16
17 class ArmyCar extends Car {
18     String weapon;
19
20     public ArmyCar() {
21         super();
22         weapon = null;
23     }
24
25     public ArmyCar(int wheels, int price, String model, String weapon) {
26         super( wheels: 8, price: 40000, model: "army");
27         this.weapon = weapon;
28     }
29
30 }
31
```

Parent Class variable can hold child class data

```
class Car {  
    // Car data and methods  
}  
  
class SuperCar extends Car {  
    // SuperCar data and methods  
}  
  
class ArmyCar extends Car {  
    // ArmyCar data and methods  
}  
  
public class InheritanceExamples {  
    public static void main(String[] args) {  
        Car amarGari = new ArmyCar();  
        Car tomarGari = new SuperCar();  
    }  
}
```

Constructor Examples

```
class Car {  
}  
  
class SuperCar extends Car {  
}  
  
class ArmyCar extends Car {  
}  
  
public class InheritanceExamples {  
    public static void main(String[] args) {  
        Car amarGari = new ArmyCar();  
        Car tomarGari = new SuperCar();  
    }  
}
```


Constructor Examples

You can also overload constructors:

```
class Car {  
    int wheels;  
    int price;  
    String model;  
  
    public Car(int wheels, int price, String model) {  
        this.wheels = wheels;  
        this.price = price;  
        this.model = model;  
    }  
  
    public Car(int price, String model) {  
        this.wheels = 4;  
        this.price = price;  
        this.model = model;  
    }  
}
```

Default Constructor

If you don't implement any constructor in your class, the Java compiler inserts default constructor into your code on your behalf. You will not see the default constructor in your source code (the .java file) as it is inserted during compilation and present in the bytecode (.class file).

The default constructor is inserted by compiler and has no code in it, on the other hand we can implement no-arg constructor in our class which looks like default constructor but we can provide any initialization code in it.

Access Modifiers

- ▶ **Private:** The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- ▶ **Default:** The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- ▶ **Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- ▶ **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Understanding Java Access Modifiers

Let's understand the access modifiers in Java by a simple table:

Most Restrictive ←————→ Least Restrictive				
Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

THE END