CSC 3215: Object Oriented Programming - 1 (JAVA)

Inhertance and Polymorphism

Dr. Kamruddin Nur

Associate Professor
Dept. of Computer Science, AIUB
kamruddin@aiub.edu

October, 2018

Contents

Inheritance

- Inheritance is a **mechanism** to **acquire** (inherit) all **properties** and **behaviours** of parent object
- We can **create new classes** built upon **existing classes**
- We can **reuse** **methods** and **fields** (member variables) of **parent** class and also can create new methods and fields
- Widely known as **parent-child** relationship also know as **IS-A relationship**

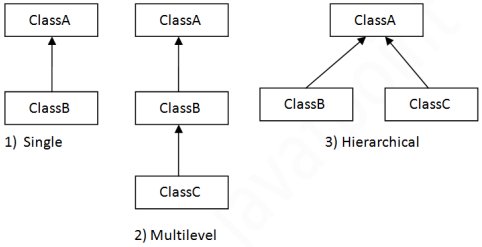
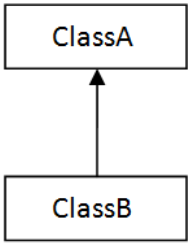
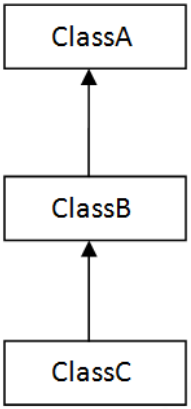


Figure 1: Types of Inheritance, figure source: javatpoint.com/inheritance-in-java

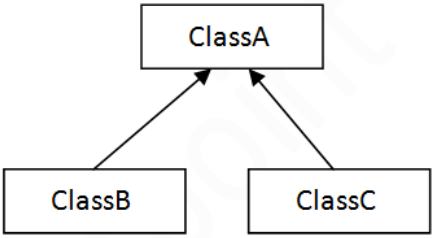
Inheritance *(Cont'd)*



1) Single



2) Multilevel



3) Hierarchical

Single inheritance

```java
package singleinheritance;

public class Father {
 //fields/properties/member variables
 private String name;
 private String property;

 //constructors
public Father(){}
public Father(String n, String p){
 this.name = n;
 this.property = p;
}

//methods/behaviours/functionalities
public void driveCar(){
 System.out.println("Driving our car
     ...");
}

public void showInfo(){
System.out.println("\nI am : " +
    this.name
  + "\nI have: "+ this.property);
}
```

```java
package singleinheritance;

public class Child extends Father
    {

private String otherthings = "";

public Child(String n, String p) {
super(n, p); //calling base
    class's constructor to access
    private properties
otherthings = "Game console"
}

public void rideBicycle() {
System.out.println("Riding my
    bicycle ...");
}

public void playGame() {
System.out.println("Playing my "
    + this.otherthings +" ...");
}
}
```

Single inheritance *Cont'd*

```
1  package singleinheritance;
2
3  public class Main {
4   public static void main(String[] args) {
5    Father f = new Father("Mr. Father",
         "House, Land and Car.");
6    f.showInfo();
7    f.driveCar();
8
9    Child c = new Child("Mr.
         Child","Bicycle");
10   c.showInfo();
11   c.driveCar();
12   c.rideBicycle();
13   c.playGame();
14  }
15 }
```

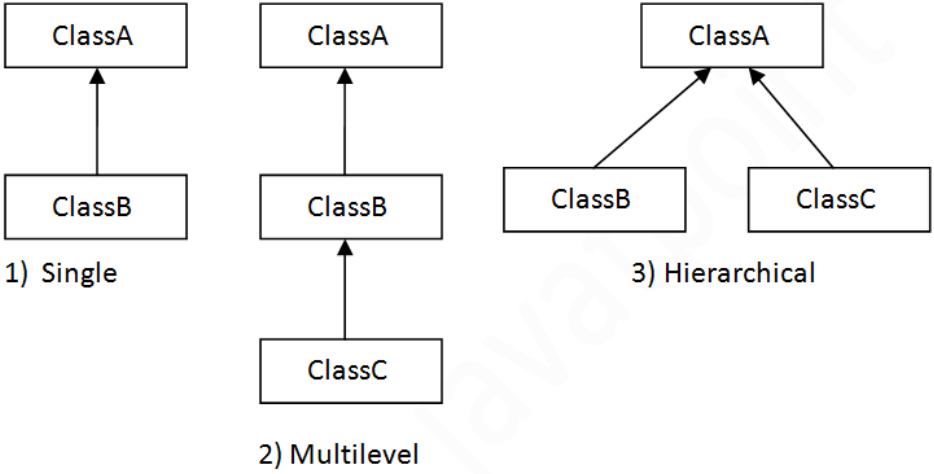
Single inheritance *Cont'd*

```
1  package singleinheritance;
2
3  public class Main {
4   public static void main(String[] args) {
5    Father f = new Father("Mr. Father",
         "House, Land and Car.");
6    f.showInfo();
7    f.driveCar();
8
9    Child c = new Child("Mr.
         Child","Bicycle");
10   c.showInfo();
11   c.driveCar();
12   c.rideBicycle();
13   c.playGame();
14  }
15 }
```

Program Output:

```
I am : Mr. Father
I have: House, Land and Car.
Driving our car ...

I am : Mr. Child
I have: Bicycle
Driving our car ...
Riding my bicycle ...
Playing my Game console ...
```

Types of Inheritance (recap)



Hierarchical Inheritance

```java
package hierarchicalinheritance;

public class Boy extends Father {

private String otherthings = "Game
    console";

public Boy(String n, String p) {
 super(n, p); //calling base class's
     constructor to access private
     properties
}

public void rideBicycle() {
 System.out.println("Riding my bicycle
     ...");
}

public void playGame() {
 System.out.println("Playing my " +
     this.otherthings +" ...");
}
}
```

```java
package hierarchicalinheritance;

public class Girl extends Father {

private String otherthings =
    "Skipping rope";

public Girl(String n, String p) {
super(n, p); //calling base
    class's constructor to access
    private properties
}

public void cook() {
System.out.println("Cooking
    Biriyani ...");
}

public void playGame() {
System.out.println("Playing with
    my " + this.otherthings +"
    ...");
}
}
```

Hierarchical inheritance *(Cont'd)*

```
1  package hierarchicalinheritance;
2
3  public class Main {
4  public static void main(String[] args) {
5  Father f = new Father("Mr. Father",
       "House, Land and Car.");
6  f.showInfo();
7  f.driveCar();
8
9  Boy b = new Boy("Mr. Boy","Bicycle");
10 b.showInfo();
11 b.driveCar();
12 b.playGame();
13
14 Girl g = new Girl("Ms.
       Girl","Tricycle");
15 g.showInfo();
16 g.driveCar();
17 g.playGame();
18
19 }
20 }
```

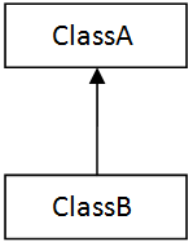
Hierarchical inheritance *(Cont'd)*

```
1  package hierarchicalinheritance;
2
3  public class Main {
4  public static void main(String[] args) {
5  Father f = new Father("Mr. Father",
       "House, Land and Car.");
6  f.showInfo();
7  f.driveCar();
8
9  Boy b = new Boy("Mr. Boy","Bicycle");
10 b.showInfo();
11 b.driveCar();
12 b.playGame();
13
14 Girl g = new Girl("Ms.
       Girl","Tricycle");
15 g.showInfo();
16 g.driveCar();
17 g.playGame();
18
19 }
20 }
```

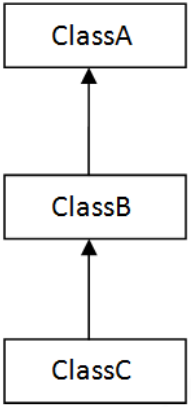
Program Output:

```
I am : Mr. Father
I have: House, Land and Car.
Driving our car ...

I am : Mr. Boy
I have: Bicycle
Driving our car ...
Playing my Game console ...

I am : Ms. Girl
I have: Tricycle
Driving our car ...
Playing with my Skipping rope
     ...
```

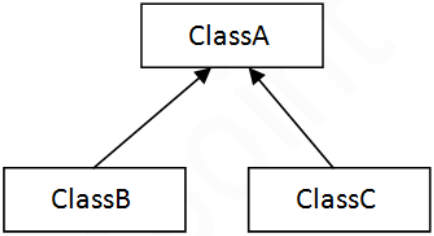
Types of Inheritance (recap)



1) Single



2) Multilevel



3) Hierarchical

Multi-level Inheritance

```
1  package multi.levelinheritance;
2
3  public class GrandFather {
4  //fields/properties/member variables
5  private String name;
6  private String property;
7
8  //constructors
9  public GrandFather() {
10 }
11
12 public GrandFather(String n, String p)
       {
13 this.name = n;
14 this.property = p;
15 }
16
17 public void showInfo() {
18 System.out.println("\nI am : " +
       this.name
19 + "\nI have: " + this.property);
20 }
21 }
```

```
1  package multi.levelinheritance;
2
3  public class Father extends
       GrandFather {
4  //fields/properties/member
       variables
5
6  //constructors
7  public Father() { }
8  public Father(String n, String p)
       {
9  super(n,p); //calling base
       class's constructor to access
       private properties
10 }
11
12 //methods/behaviours/functionalities
13 public void driveCar() {
14 System.out.println("Driving our
       car ...");
15 }
16
17 }
```

```
1  package multi.levelinheritance;
```

Multi-level Inheritance *(Cont'd)*

```
1  package multi.levelinheritance;
2
3  public class Boy extends Father {
4
5  private String otherthings = "Game console";
6
7  public Boy(String n, String p) {
8  super(n, p); //calling base class's constructor to access private
       properties
9  }
10
11 public void rideBicycle() {
12 System.out.println("Riding my bicycle ...");
13 }
14
15 public void playGame() {
16 System.out.println("Playing my " + this.otherthings +" ...");
17 }
18 }
```

Multi-level inheritance *(Cont'd)*

```
1  package multi.levelinheritance;
2
3  public class Main {
4  public static void main(String[] args) {
5
6  GrandFather gf = new GrandFather("Mr.
       Grand Father", "House, Land and
       Car.");
7  gf.showInfo();
8
9  Father f = new Father("Mr. Father",
       "Car");
10 f.showInfo();
11 f.driveCar();
12
13 Boy b = new Boy("Mr. Boy","Bicycle");
14 b.showInfo();
15 b.driveCar();
16 b.playGame();
17 }
18 }
```

Multi-level inheritance *(Cont'd)*

```
1  package multi.levelinheritance;
2
3  public class Main {
4  public static void main(String[] args) {
5
6  GrandFather gf = new GrandFather("Mr.
       Grand Father", "House, Land and
       Car.");
7  gf.showInfo();
8
9  Father f = new Father("Mr. Father",
       "Car");
10 f.showInfo();
11 f.driveCar();
12
13 Boy b = new Boy("Mr. Boy","Bicycle");
14 b.showInfo();
15 b.driveCar();
16 b.playGame();
17 }
18 }
```

Program Output:

```
I am : Mr. Grand Father
I have: House, Land and Car.

I am : Mr. Father
I have: Car
Driving our car ...

I am : Mr. Boy
I have: Bicycle
Driving our car ...
Playing my Game console ...
```

Multiple Inheritance??

- Multiple inheritance is not supported in java through class.
- However, it can be achieved through interface in different way

2 Method Overloading and Overriding

Method Overloading and Overriding

- **Method Overloading** - multiple methods having same name but different in parameters
- Two ways to overload -
  1. By changing number of arguments
  2. By changing the data type
  3. For example,

```
1 int sum (int a, int b);
2 int sum (int a, int b, int c);
3 float sum (float a, float b);
```

- **Method Overriding** - child class has the same method as declared in the parent class
- Purposes -
  1. to provide specific implementation of a method that is already provided by its super class
  2. Method overriding is used for runtime **polymorphism**

Method Overloading and Overriding *(Cont'd)*

Method overloading example,

```
1  //in GrandFather class
2  public void showInfo() {
3  System.out.println("\nI am : " + this.name
4  + "\nI have: " + this.property);
5  }
6
7  //we could override in parent class
8  public void showInfo() {
9   super.showInfo();
10  System.out.println("I have also : " + this.otherthings);
11 }
```

Method Overriding Conditions:

- method must have same name as in the parent class
- method must have same parameter as in the parent class.
- must be IS-A relationship (inheritance).

Thanks

Thanks for your time and attention!

kamruddin@aiub.edu