Inheritance: Overriding methods

1. Overriding methods
2. Final keyword

Overriding methods

1. Overriding methods

When a class extends another class:

All public methods declared in that superclass are automatically included in the subclass.

1. Overriding methods

```
public class Clothing {
    String cod;
    double price;
    String size;
    String color;
    char genre; //W==Woman, M==Man


    public String display() {
        return "cod='" + cod + '\'' +
                ", price=" + price +
                ", size='" + size + '\'' +
                ", color='" + color + '\'' +
                ", genre=" + genre;
    }
}
```

Clothing.java

1. Overriding methods

```
public class Clothing{
   String cod;
   double price;
   String size;
   String color;
   char genre;
   public String display() {…}
}
```

```
public class Jeans extends Clothing{
   String type;
}
```

1. Overriding methods

```
public class Clothing{
   String cod;
   double price;
   String size;
   String color;
   char genre;
   public String display() {…}
}
```



```
public class Jeans extends Clothing{
   String cod;
   double price;
   String size;
   String color;
   char genre;
   String type;
   public String display() {…}
}
```

Overriding a method:

Re-declaring the method in the subclass and then re-defining what it should do.

1. Overriding methods

```
public class Main {

    public static void main(String[] args) {

        Jeans jeans = new Jeans();
        jeans.cod = "9663/310";
        jeans.price = 39.95;
        jeans.size = "40";
        jeans.color = "blue";
        jeans.genre = 'M';
        jeans.type = "slim";
        System.out.println(jeans.display());
    }
}
```

Main.java

1. Overriding methods

```
public class Main {

    public static void main(String[] args) {

        Jeans jeans = new Jeans();
        jeans.cod = "9663/310";
        jeans.price = 39.95;
        jeans.size = "40";
        jeans.color = "blue";
        jeans.genre = 'M';
        jeans.type = "slim";
        System.out.println(jeans.display());
    }
}
```

Main.java

1. Overriding methods

```
public class Main {

    public static void main(String[] args) {

        Jeans jeans = new Jeans();
        jeans.cod = "9663/310";
        jeans.price = 39.95;
        jeans.size = "40";
        jeans.color = "blue";
        jeans.genre = 'M';
        jeans.type = "slim";
        System.out.println(jeans.display());
        //Prints: cod='9663/310', price=39.95, size='40', color='blue', genre=M
    }
}
```

Main.java

1. Overriding methods

```
public class Jeans extends Clothing{


    String type; //slim, fit, ..


    public String display() {
      return "Jeans{" +
            "cod='" + cod + '\'' +
            ", price=" + price +
            ", size='" + size + '\'' +
            ", color='" + color + '\'' +
            ", genre=" + genre +
            ", type='" + type + '\'' +
            '}';
    }
}
```

Jeans.java

1. Overriding methods

```
public class Jeans extends Clothing{


    String type; //slim, fit, ..


    public String display() {
      return "Jeans{" +
          "cod='" + cod + '\'' +
          ", price=" + price +
          ", size='" + size + '\'' +
          ", color='" + color + '\'' +
          ", genre=" + genre +
          ", type='" + type + '\'' +
          '}';
    }
}
```

Jeans.java

1. Overriding methods

```
public class Jeans extends Clothing{


    String type; //slim, fit, ..


    public String display() {
        return "Jeans{" +
                "cod='" + cod + '\'' +
                ", price=" + price +
                ", size='" + size + '\'' +
                ", color='" + color + '\'' +
                ", genre=" + genre +
                ", type='" + type + '\'' +
                '}';
    }
}
```

Jeans.java

1. Overriding methods

```
public class Main {

    public static void main(String[] args) {

        Jeans jeans = new Jeans();
        jeans.cod = "9663/310";
        jeans.price = 39.95;
        jeans.size = "40";
        jeans.color = "blue";
        jeans.genre = 'M';
        jeans.type = "slim";
        System.out.println(jeans.display());
            //Prints: Jeans{cod='9663/310', price=39.95, size='40', color='blue', genre=M,        //type='slim'}
    }
}
```

Main.java

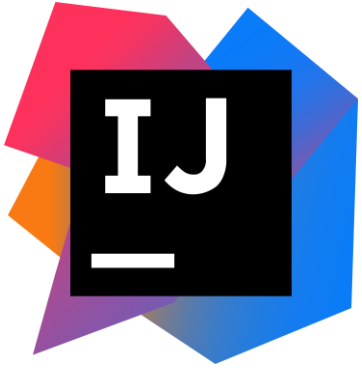
1. Overriding methods

```
public class Socks extends Clothing{


   String length; //knee-high, mid-calf, ...


   public String display() {
     return ”Socks{" +
         "cod='" + cod + '\'' +
         ", price=" + price +
         ", size='" + size + '\'' +
         ", color='" + color + '\'' +
         ", genre=" + genre +
         ", length='" + length + '\'' +
         '}';
   }
}
```

Socks.java



ClothingManager_overrideMethod

1. Overriding methods

¿Cuál es la salida por consola al ejecutar el siguiente código?

```
Truck myCar = new Truck();
myCar.m1();
myCar.m2();
```

```
public class Car{
  public void m1(){ System.out.println("car 1"); }
  public void m2(){ System.out.println("car 2"); }

}
```

```
public class Truck extends Car{
  public void m1(){ System.out.println("truck 1"); }
}
```

1. Overriding methods

¿Cuál es la salida por consola al ejecutar el siguiente código?

```
Truck myCar = new Truck();
myCar.m1(); //Prints: truck 1
myCar.m2();
```

```
public class Car{
  public void m1(){ System.out.println("car 1"); }
  public void m2(){ System.out.println("car 2"); }

}
```

```
public class Truck extends Car{
  public void m1(){ System.out.println("truck 1"); }
}
```

1. Overriding methods

¿Cuál es la salida por consola al ejecutar el siguiente código?

```
Truck myCar = new Truck();
myCar.m1(); //Prints: truck 1
myCar.m2(); //Prints car 2
```

```
public class Car{
  public void m1(){ System.out.println("car 1"); }
  public void m2(){ System.out.println("car 2"); }

}
```

```
public class Truck extends Car{
  public void m1(){ System.out.println("truck 1"); }
}
```

Final keyword

2. Final keyword

To **protect** your method from being **overridden** in a child class use keyword **final**.

2. Final keyword

```
public class Room{
  double width;
  double length;

  public final double getArea(){
    return width*length;
  }
}
```

Room.java

2. Final keyword

```
public class LivingRoom extends Room{

  // Not allowed to override getArea() here
}
```

LivingRoom.java

2. Final keyword

```
public class Main {

    public static void main(String[] args) {

        LivingRoom livingRoom = new LivingRoom(5,3);
        double area = livingRoom.getArea();
        System.out.println(area); //prints 15

    }
}
```

Main.java

2. Final keyword

```
public class Main {

    public static void main(String[] args) {

        final double MAX_ROOMS = 10;



        LivingRoom livingRoom = new LivingRoom(5,3);
        double area = livingRoom.getArea();
        System.out.println(area); //prints 15

    }
}
```

Main.java

2. Final keyword

```
public class Main {

    public static void main(String[] args) {

        final double MAX_ROOMS = 10;
        MAX_ROOMS = 0; // This is not allowed and will show a compiler error!

        LivingRoom livingRoom = new LivingRoom(5,3);
        double area = livingRoom.getArea();
        System.out.println(area); //prints 15

    }
}
```

Main.java

“Advice to graduates: Do something you really enjoy doing. If it isn’t fun to get up in the morning and do your job or your school program, you’re in the wrong field“

Brian Kernigan, científico de la computación que trabajó el laboratorios Bell y ayudó en la creación del sistema operativo Unix.