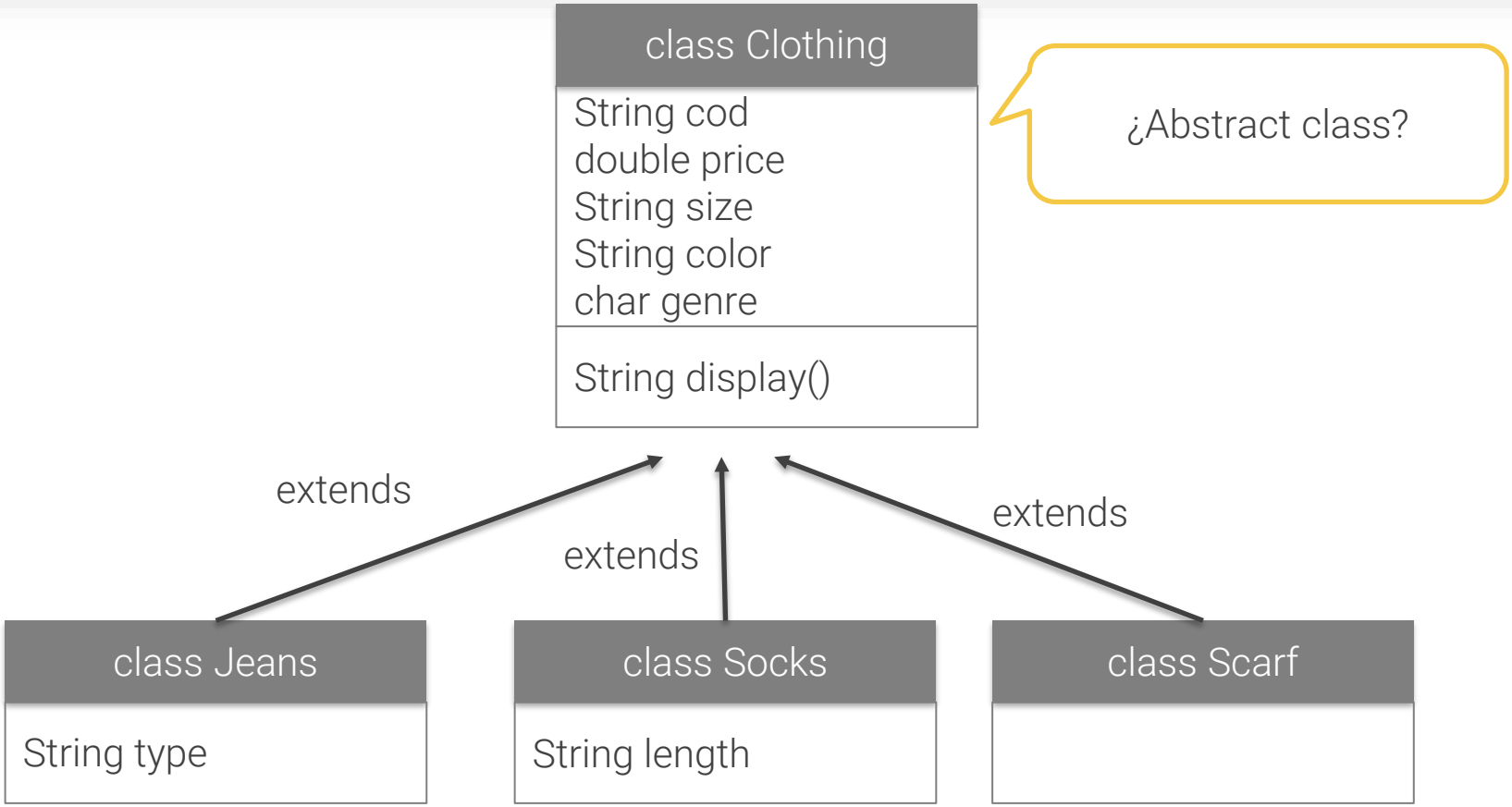
Inheritance: Abstract classes

1. Abstract classes
2. Abstract methods

Abstract classes

1. Abstract classes



1. Abstract classes

- **Abstract class:** it groups the common properties and behavior of its derived classes, <u>but it prevents itself from being instantiated</u>

1. Abstract classes

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

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

An abstract class must be declared with the abstract keyword

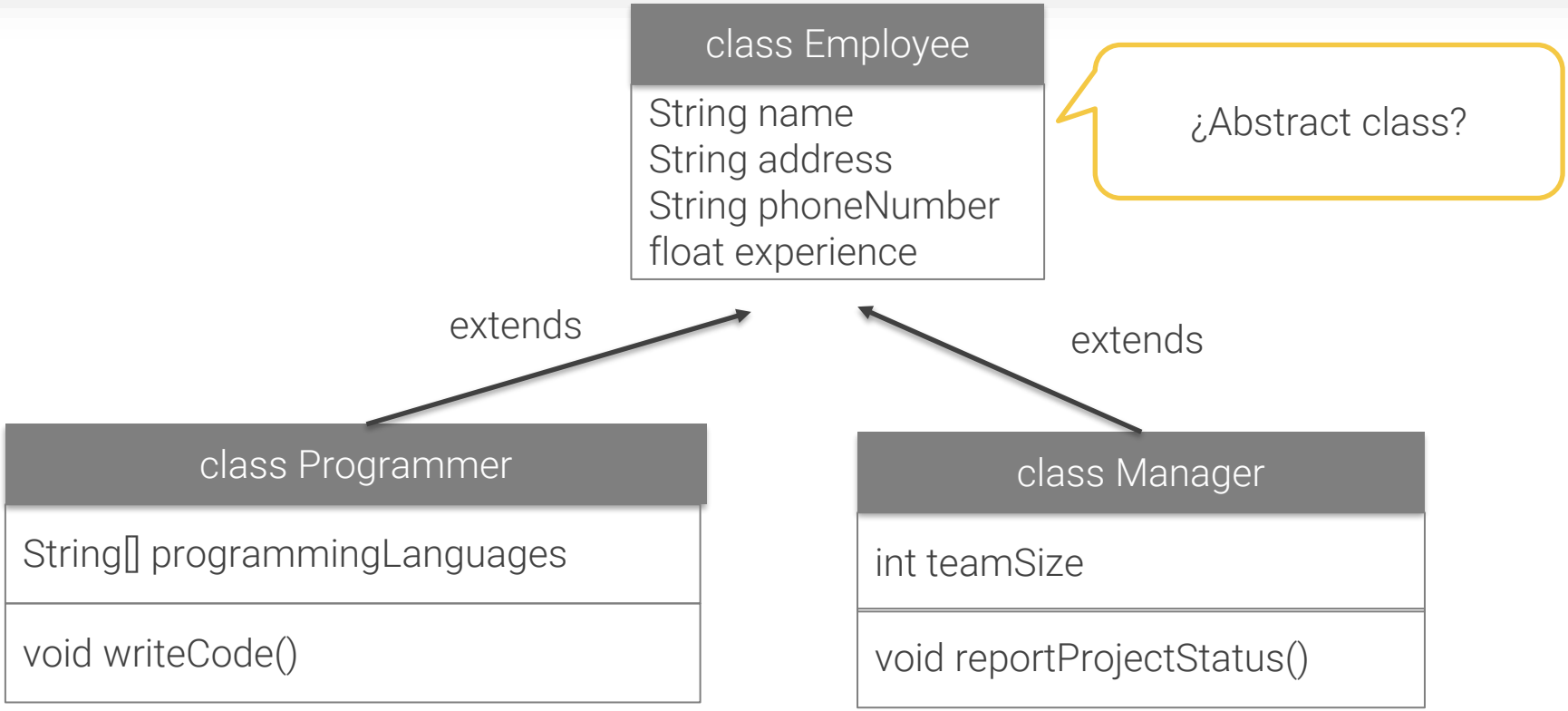
Clothing.java

1. Abstract classes

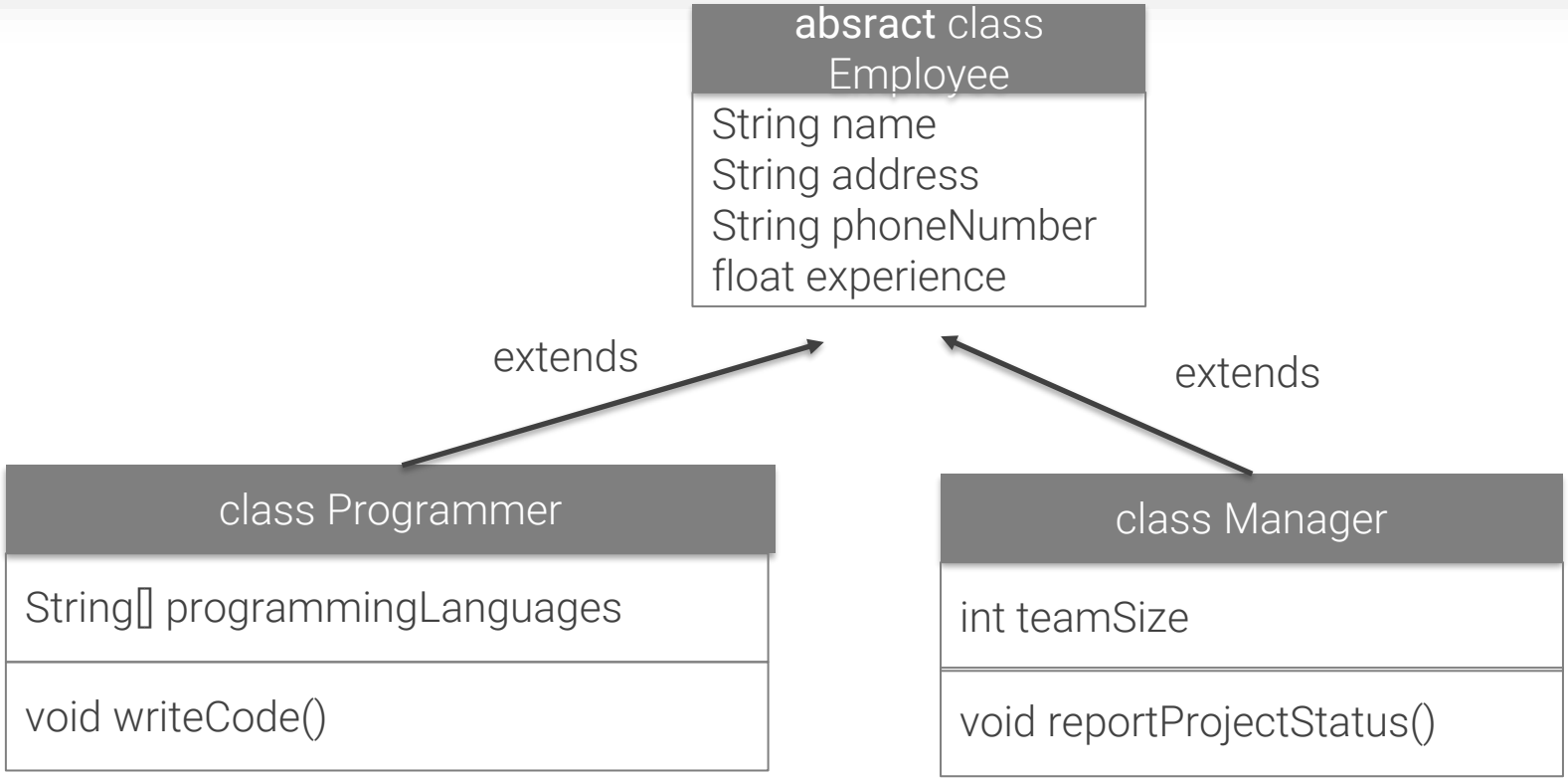
Main.java | Clothing.java | Jeans.java | Socks.java | Scarf.java

```java
package com.company;

public class Main {

    public static void main(String[] args) {

        Clothing clothing = new Clothing();

        Jeans jeans = 'Clothing' is abstract; cannot be instantiated size: "40", color: "blue", genre: 'M', type: "slim");
        System.out.println(jeans.display());

        Socks socks = new Socks( cod: "7347/305", price: 5.95, size: "M", color: "grey", genre: 'W', length: "mid-calf" );
        System.out.println(socks.display());

        Scarf scarf = new Scarf( cod: "7747/205", price: 15.95, size: "U", color: "grey", genre: 'W');
        System.out.println(scarf.display());
    }
}
```

1. Abstract classes



1. Abstract classes



Abstract methods

2. Abstract methods

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

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

It can have abstract and non abstract methods

Clothing.java

2. Abstract methods

Abstract method: a method which is declared as abstract and does not have implementation

```
public abstract double finalPrice();
```

Does not have implementation

2. Abstract methods

finalPrice = price / (1-profitMargin)

2. Abstract methods

finalPrice = price / (1-profitMargin)

Jeans

0.10-0.20

Socks

0.05

Scarf

0.10

2. Abstract methods

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

   public Clothing(String cod, double price, String size, String color, char genre) {
    this.cod = cod;
    this.price = price;
    this.size = size;
    this.color = color;
    this.genre = genre;
   }
   public String display() {
      return "cod='" + cod + '\'' +
            ", price=" + price +
            ", size='" + size + '\'' +
            ", color='" + color + '\'' +
            ", genre=" + genre;
   }
   public abstract double finalPrice();
}
```

Clothing.java

2. Abstract methods

Main.java | Clothing.java | Jeans.java | Socks.java | Scarf.java

```java
package com.company;

public class Jeans extends Clothing{
    private String type; //clim, f/, etc

    public Jeans(String cod, double price, String size, String color, char genre, String type) {
        super(cod, price, size, color, genre);
        this.type = type;
    }

    public String display() {
        return "Jeans{" +
                super.display() +
                ", type='" + type + '\'' +
                "}";
    }

}
```

Class 'Jeans' must either be declared abstract or implement abstract method 'finalPrice()' in 'Clothing'

2. Abstract methods

```
public class Jeans extends Clothing{

    private String type; //slim, fit, etc

    ...
    public double finalPrice(){
        double profitMargin;
        switch (type){
            case "slim":
                profitMargin=0.10;
                break;
            case "fit":
                profitMargin=0.15;
                break;
            default:
                profitMargin=0.20;
                break;
        }
        return price / (1-profitMargin);

    }
}
```

Jeans

profitMargin
0.10-0.20

Jeans.java

2. Abstract methods

```
public class Socks extends Clothing{

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

    ...

    public double finalPrice() {
        double profitMargin = 0.05;
        return price / (1-profitMargin);
    }
}
```

Socks

profitMargin
0.05

Socks.java

2. Abstract methods

```
public class Scarf extends Clothing{

   public Scarf(String cod, double price, String size, String color, char genre) {
      super(cod, price, size, color, genre);
   }


   public double finalPrice() {
      double profitMargin = 0.10;
      return price / (1-profitMargin);
   }
}
```

Scarf

profitMargin:
0.10

Scarf.java

“Un día sin reír es un día perdido“

Charles Chaplin, actor y humorista inglés