Fundamentos de Java

Parte 8

Presenta

Alan Badillo Salas

Marzo 2023

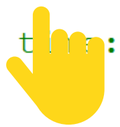
```java
class Product {

    String name;
    double price;

    public void describe() {
        System.out.println(x: "PRODUCT");
        System.out.println(x: "-----------------------");
        System.out.printf(format: "NAME: %s %n", args: name);
        System.out.printf(format: "PRICE: %.2f %n", args: price);
        System.out.println(x: "-----------------------");
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

}
```

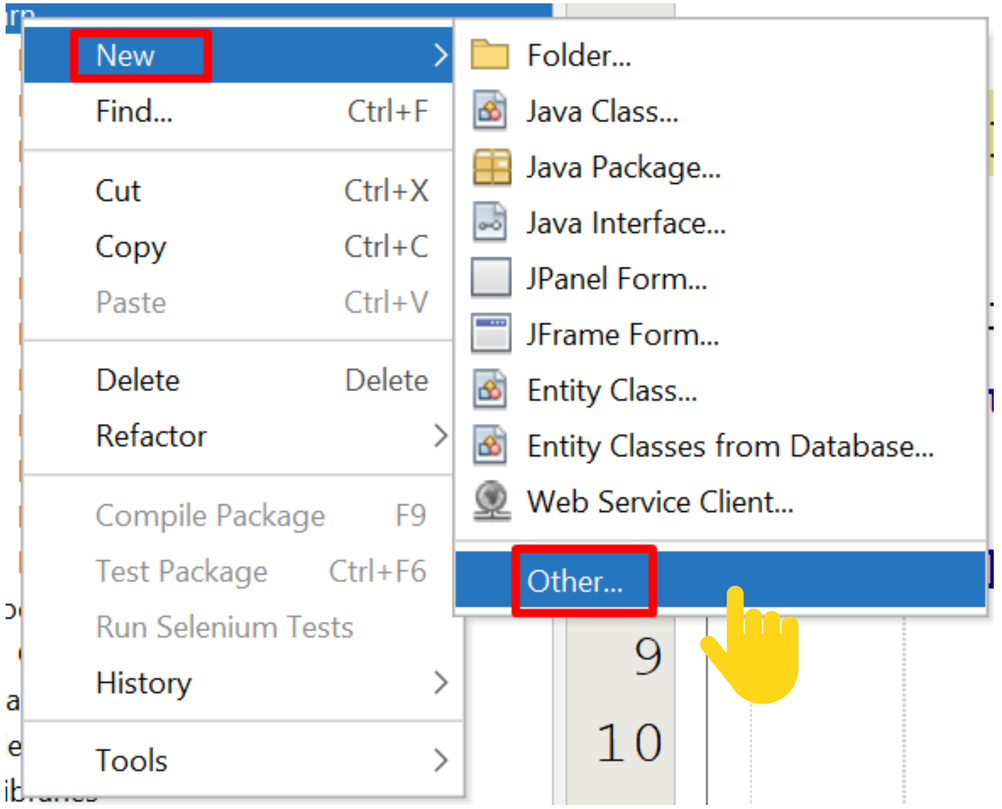
```java
class DigitalProduct extends Product {

    String url;
    int size;

    @Override
    public void describe() {
        System.out.println(x: "PRODUCT (DIGITAL)");
        System.out.println(x: "----------------------");
        System.out.printf(format: "NAME: %s %n", args: name);
        System.out.printf(format: "PRICE: %.2f %n", args: price);
        System.out.println(x: "----------------------");
        System.out.printf(format: "URL: %s %n", args: url);
        System.out.printf(format: "size: %d bytes %n", args: size);
        System.out.println(x: "----------------------");
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

}
```

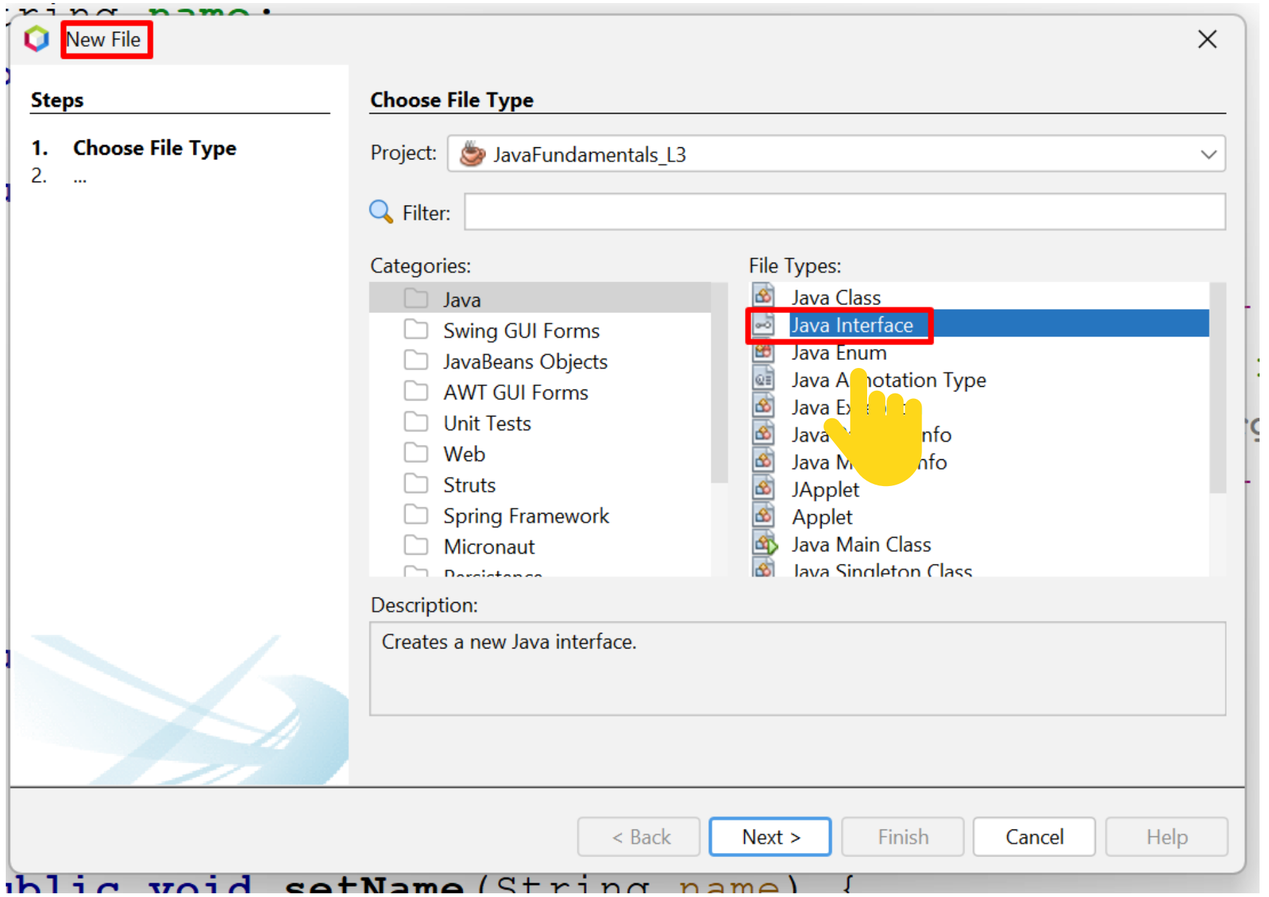
```java
public static void main(String[] args) {

    Product mySimpleProduct = new Product();

    mySimpleProduct.setName(name: "Coca Cola");
    mySimpleProduct.setPrice(price: 17.99);

    mySimpleProduct.describe();

    DigitalProduct myDigitalProduct = new DigitalProduct();

    myDigitalProduct.setName(name: "Naruto 3D Model");
    myDigitalProduct.setPrice(price: 9.99);
    myDigitalProduct.setUrl(url: "https://skfb.ly/oCFW8");
    myDigitalProduct.setSize(size: 12514);

    myDigitalProduct.describe();

}
```

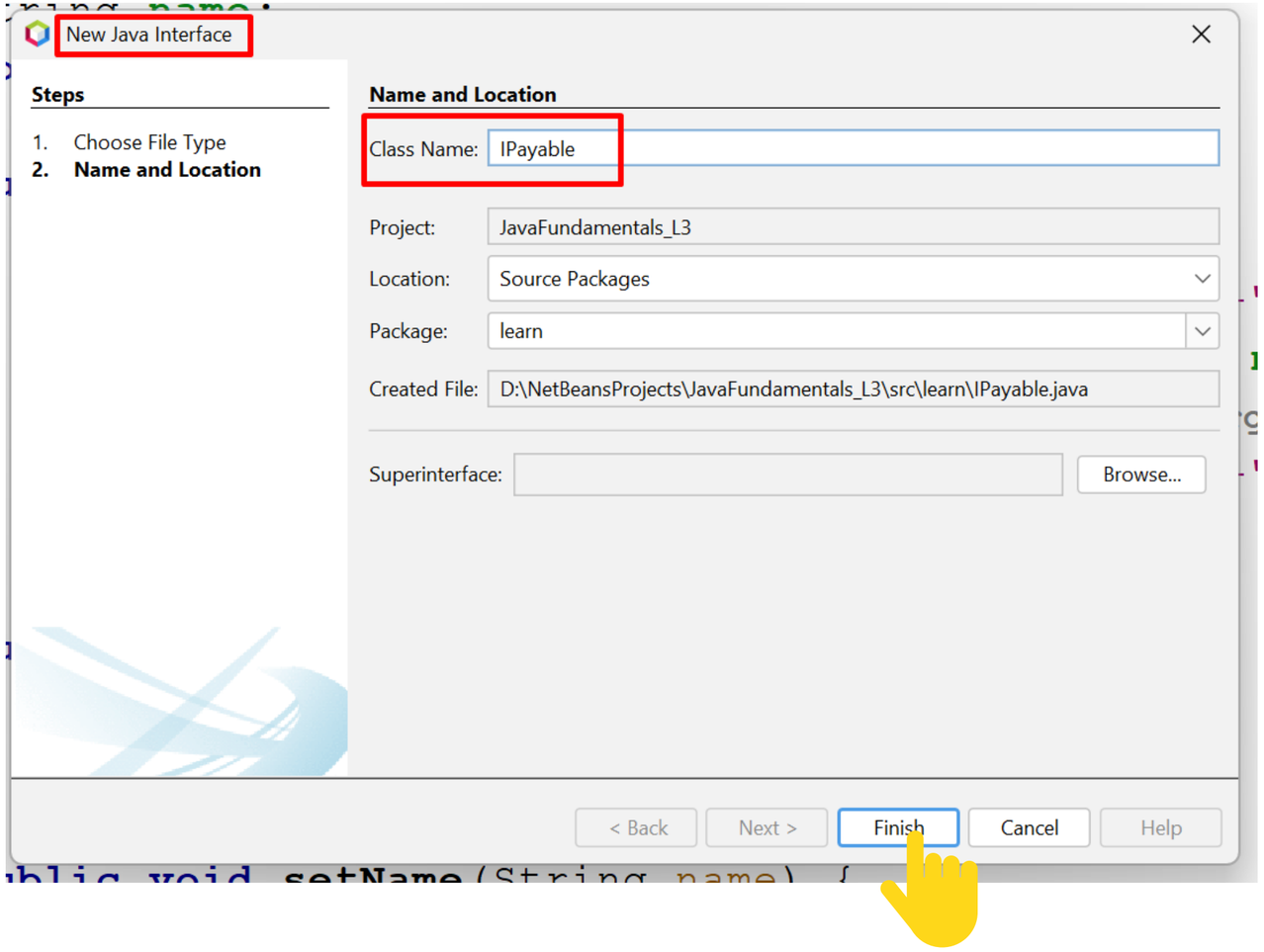
```java
public static void main(String[] args) {

    Product mySimpleProduct = new Product();

    mySimpleProduct.setName(name: "Coca Cola");
    mySimpleProduct.setPrice(price: 17.99);

    mySimpleProduct.describe();

    DigitalProduct myDigitalProduct = new DigitalProduct();

    myDigitalProduct.setName(name: "Naruto 3D Model");
    myDigitalProduct.setPrice(price: 9.99);
    myDigitalProduct.setUrl(url: "https://skfb.ly/oCFW8");
    myDigitalProduct.setSize(size: 12514);

    myDigitalProduct.describe();

}
```

```
run:
PRODUCT
----------------------
NAME: Coca Cola
PRICE: 17.99
----------------------
PRODUCT (DIGITAL)
----------------------
NAME: Naruto 3D Model
PRICE: 9.99
----------------------
URL: https://skfb.ly/oCFW8
size: 12514 bytes
----------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```




New
Find... Ctrl+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Refactor
Compile Package F9
Test Package Ctrl+F6
Run Selenium Tests
History
Tools
Folder...
Java Class...
Java Package...
Java Interface...
JPanel Form...
JFrame Form...
Entity Class...
Entity Classes from Database...
Web Service Client...
Other...
9
10

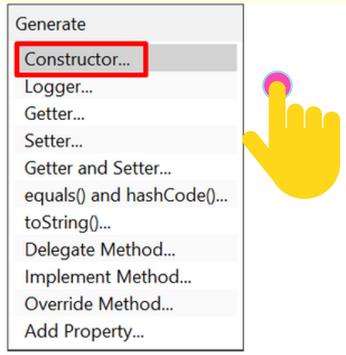

New File
Steps
1. Choose File Type
2. ...
Choose File Type
Project:
JavaFundamentals_L3
Filter:
Categories:
Java
Swing GUI Forms
JavaBeans Objects
AWT GUI Forms
Unit Tests
Web
Struts
Spring Framework
Micronaut
File Types:
Java Class
Java Interface
Java Enum
JApplet
Applet
Java Main Class
Description:
Creates a new Java interface.
< Back
Next >
Finish
Cancel
Help

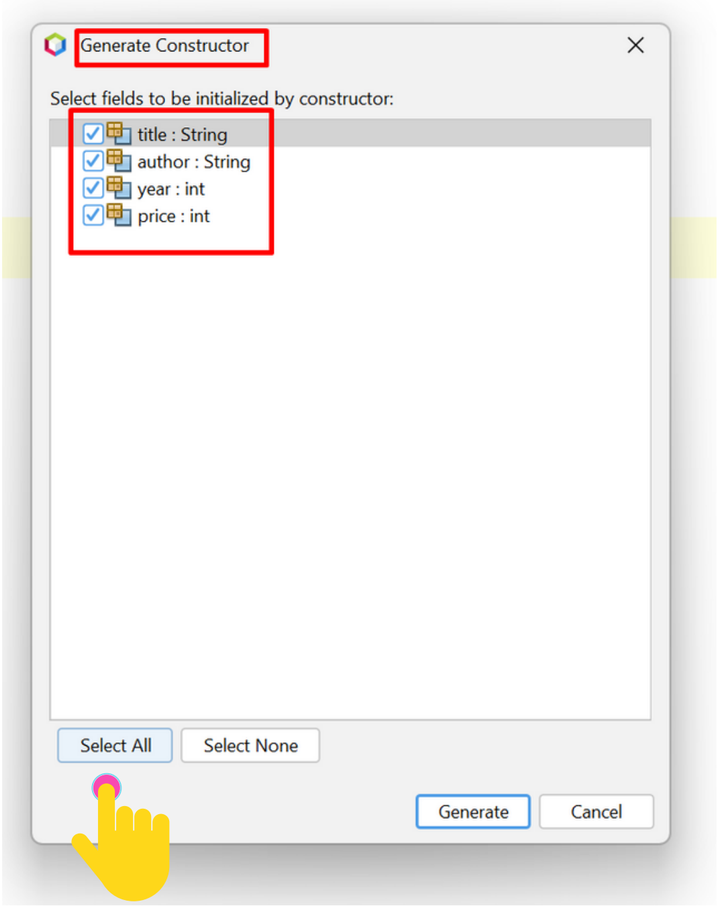

New Java Interface
Steps
1. Choose File Type
2. Name and Location
Name and Location
Class Name: IPayable
Project: JavaFundamentals_L3
Location: Source Packages
Package: learn
Created File: D:\NetBeansProjects\JavaFundamentals_L3\src\learn\IPayable.java
Superinterface:
Browse...
< Back
Next >
Finish
Cancel
Help

```java
public interface IPayable {

    public String getDescription();

    public double getPrice();

    public void paymentSuccess(int payStatus, String paymentVerificationCode);

    public void paymentFail(String reason);

}
```

```
class Book implements IPayable {

    String title;
    String author;
    int year;
    int price;


}
```

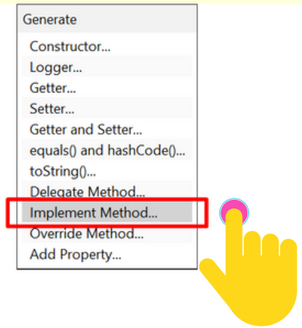
```
class Book implements IPayable {

    String title;
    String author;
    int year;
    int price;

}
```

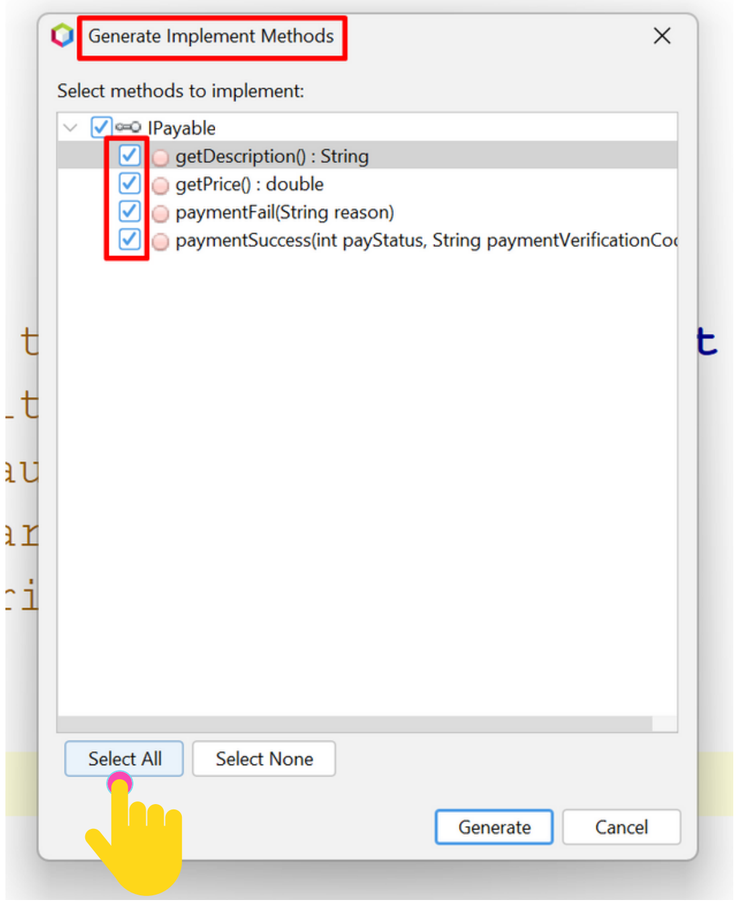



Generate Constructor
Select fields to be initialized by constructor:
title : String
author : String
year : int
price : int
Select All
Select None
Generate
Cancel

```
class Book implements IPayable {

    String title;
    String author;
    int year;
    int price;

    public Book(String title, String author, int year, int price) {
        this.title = title;
        this.author = author;
        this.year = year;
        this.price = price;
    }

}
```

```
class Book implements IPayable {

    String title;
    String author;
    int year;
    int price;

    public Book(String title, String author, int year, int price) {
        this.title = title;
        this.author = author;
        this.year = year;
        this.price = price;
    }


}
```



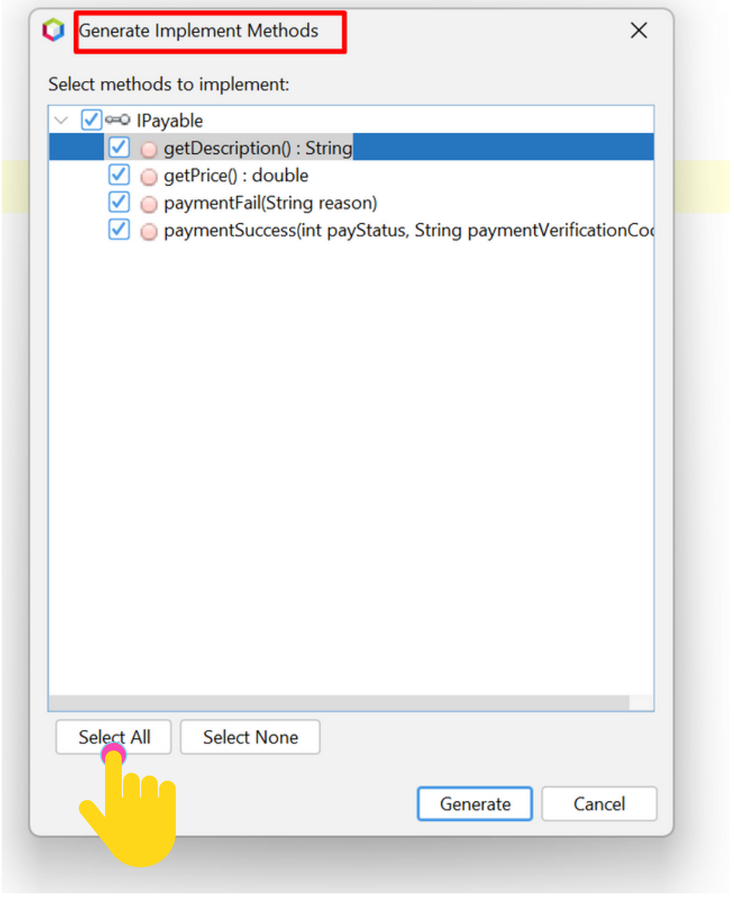

Generate Implement Methods
Select methods to implement:
IPayable
getDescription() : String
getPrice() : double
paymentFail(String reason)
paymentSuccess(int payStatus, String paymentVerificationCo
Select All
Select None
Generate
Cancel

```java
@Override
public String getDescription() {
    throw new UnsupportedOperationException(message:"Not supported yet.");
}

@Override
public double getPrice() {
    throw new UnsupportedOperationException(message:"Not supported yet.");
}

@Override
public void paymentSuccess(int payStatus, String paymentVerificationCode)
    throw new UnsupportedOperationException(message:"Not supported yet.");
}

@Override
public void paymentFail(String reason) {
    throw new UnsupportedOperationException(message:"Not supported yet.");
}
```

```java
@Override
public String getDescription() {
    return String.format(format:"Book: %s - %s (%d)",
            args: title, args: author, args: year);
}

@Override
public double getPrice() {
    return price + 0.55; // Add 0.55 of book's tax
}

@Override
public void paymentSuccess(int payStatus, String paymentVerificationCode) {
    System.out.println(x:"Book payed:");
    System.out.printf(format:"Verification Code: %s (%d) %n",
            args: paymentVerificationCode, args: payStatus);
    System.out.println(x:"Download from https://book.com/1234");

}

@Override
public void paymentFail(String reason) {
    System.out.printf(format:"Book not payed: %s %n", args: reason);
}
```

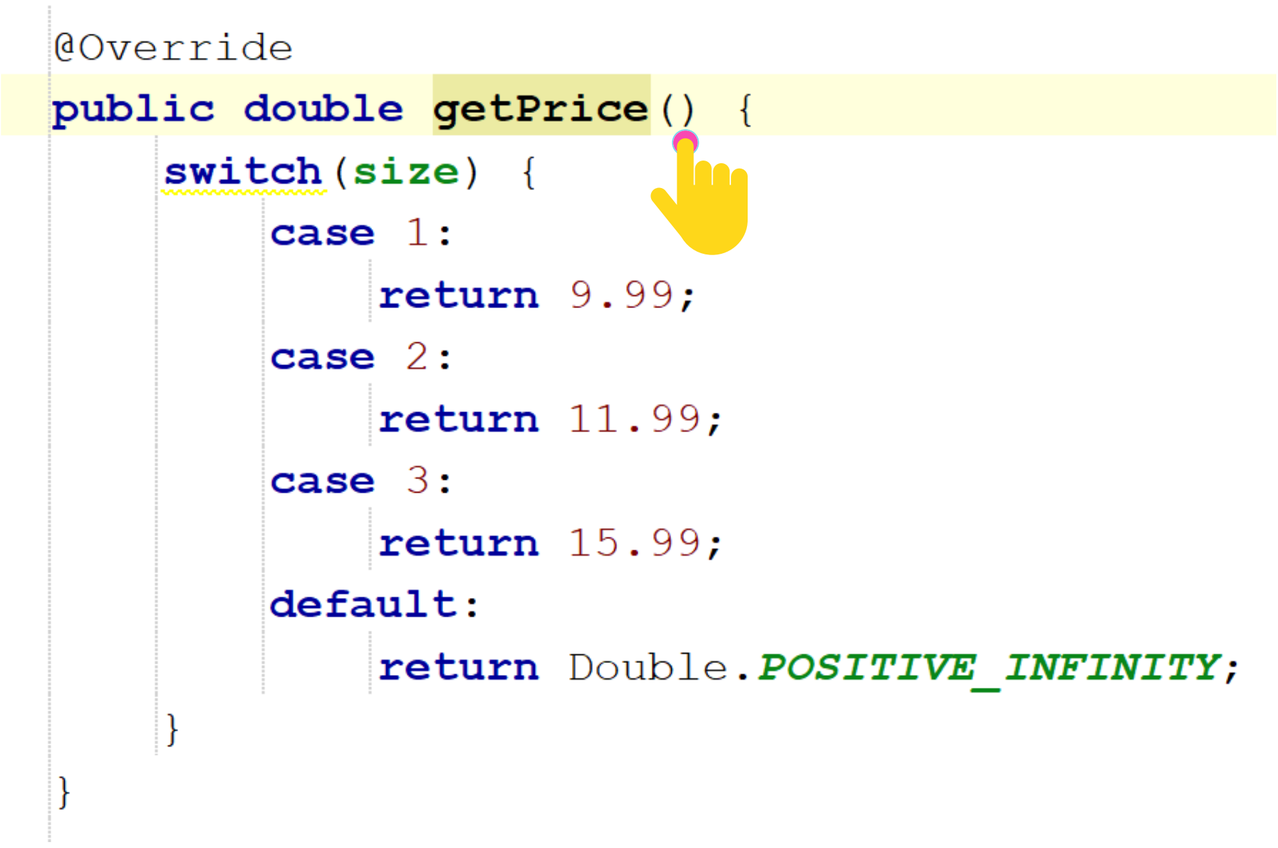
```
class CoffeeCup implements IPayable {

    String type;
    int size; // 1 - Middle, 2 - Venti, 3 - Alto


}
```

```
class CoffeeCup implements IPayable {

    String type;
    int size; // 1 - Middle, 2 - Venti, 3 - Alto

    public CoffeeCup(String type, int size) {
        this.type = type;
        this.size = size;
    }

    

}
```


Generate Implement Methods
Select methods to implement:
IPayable
getDescription() : String
getPrice() : double
paymentFail(String reason)
paymentSuccess(int payStatus, String paymentVerificationCo
Select All
Select None
Generate
Cancel

```java
@Override
public String getDescription() {
    throw new UnsupportedOperationException(message: "Not supported yet.");
}

@Override
public double getPrice() {
    throw new UnsupportedOperationException(message: "Not supported yet.");
}

@Override
public void paymentSuccess(int payStatus, String paymentVerificationCode)
    throw new UnsupportedOperationException(message: "Not supported yet.");
}

@Override
public void paymentFail(String reason) {
    throw new UnsupportedOperationException(message: "Not supported yet.");
}
```

```java
@Override
public String getDescription() {

    String sizeText;

    switch(size) {
        case 1:
            sizeText = "MIDDLE";
            break;
        case 2:
            sizeText = "VENTI";
            break;
        case 3:
            sizeText = "ALTO";
            break;
        default:
            sizeText = "UNKWNON";
            break;
    }

    return String.format(format:"Coffee Cup - %s (%s)",
            args: type, args: sizeText);
}
```

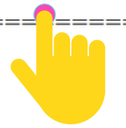

@Override
public double getPrice() {
switch(size) {
case 1:
return 9.99;
case 2:
return 11.99;
case 3:
return 15.99;
default:
return Double.POSITIVE_INFINITY;
}
}

```java
@Override
public void paymentSuccess(int payStatus, String paymentVerificationCode) {
    System.out.println(x: "Thank you :D");
    System.out.println(x: "Take your coffee cup next door");
}

@Override
public void paymentFail(String reason) {
     System.out.println(x: "Your payment failed >:(");
    System.out.println(x: "Exit is in next door");
}
```

```
public static void main(String[] args) {

    IPayable item1 = new Book(title: "Tom Sawyer", author:"Mark Twain",
            year: 1876, price: 9.87);

    IPayable item2 = new CoffeeCup(type: "Mocca", size: 2);

    Book item3 = new Book(title: "El Periquillo Sarniento",
            author:"Jose-Joaquin Fernandez-Lizardi", year: 1816, price: 15.32);

    CoffeeCup item4 = new CoffeeCup(type: "Expresso", size: 3);

    paySomething(itemPayable: item1, amount:10);
    paySomething(itemPayable: item2, amount:12);
    paySomething(itemPayable: item3, amount:20);
    paySomething(itemPayable: item4, amount:15);

}
```

```java
static void paySomething(IPayable itemPayable, double amount) {

    System.out.println(x: "=========================================");
    System.out.printf(format: "ITEM: %s %n", args: itemPayable.getDescription());
    System.out.printf(format: "PRICE: $%.2f %n", args: itemPayable.getPrice());
    System.out.printf(format: "AMOUNT: $%.2f %n", args: amount);
    System.out.println(x: "-----------------------------------------");

    System.out.println(x: "PAYING...");
    double change = amount - itemPayable.getPrice();

    if (change >= 0.0) {
        System.out.printf(format: "CHANGE: $%.2f %n", args: change);
        itemPayable.paymentSuccess(payStatus: 1, paymentVerificationCode: "12345");
    } else {
        System.out.println(x: "INSUFFICIENT FUNDS :O %n");
        itemPayable.paymentFail(reason: "Insufficient founds");
    }

    System.out.println(x: "-----------------------------------------");

    System.out.println(x: "DONE.");

    System.out.println(x: "=========================================");

}

public static void main(String[] args) {
```

```
run:
==========================================
ITEM: Book: Tom Sawyer - Mark Twain (1876)
PRICE: $10.42
AMOUNT: $10.00
------------------------------------------
PAYING...
INSUFFICIENT FUNDS :O %n
Book not payed: Insufficient founds
------------------------------------------
DONE.
==========================================
==========================================
ITEM: Coffee Cup - Mocca (VENTI)
PRICE: $11.99
AMOUNT: $12.00
------------------------------------------
PAYING...
CHANGE: $0.01
Thank you :D
Take your coffee cup next door
------------------------------------------
DONE.
==========================================
```



```
==========================================
ITEM: Book: El Periquillo Sarniento - Jose-Joaquin Fernandez-Lizardi (1816)
PRICE: $15.87
AMOUNT: $20.00
------------------------------------------
PAYING...
CHANGE: $4.13
Book payed:
Verification Code: 12345 (1)
Download from https://book.com/1234
------------------------------------------
DONE.
==========================================
==========================================
ITEM: Coffee Cup - Expresso (ALTO)
PRICE: $15.99
AMOUNT: $15.00
------------------------------------------
PAYING...
INSUFFICIENT FUNDS :O %n
Your payment failed >:(
Exit is in next door
------------------------------------------
DONE.
==========================================
BUILD SUCCESSFUL (total time: 0 seconds)
```