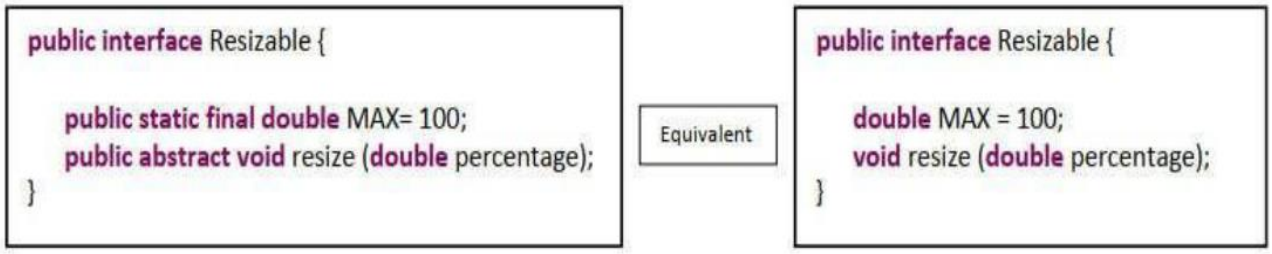
# CSE 215: Programming Language II Lab
## Sec – 8, Faculty - MUO
### Lab Officer: Tanzina Tazreen
## Lab – 11
## Interface

An interface is a completely "**abstract class**" that is used to group related abstract methods (methods with empty bodies).

- An interface cannot have regular methods, only abstract methods. But you do not need to use the keyword "abstract" to declare a method abstract.
- Like abstract classes, interfaces cannot be used to create objects.
- To use an interface, you have to inherit it from another class using the keyword "**implements**".
- If you inherit an interface, you have to implement all the abstract methods in it. thus, making overriding compulsory.
- Interface methods are by default abstract and public.
- Interface attributes are by default public, static and final.
- An interface cannot contain a constructor.




## Why And When To Use Interfaces?

1) To achieve security by complete abstraction - hide certain details and only show the important details of an object (interface).

2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can **implement** multiple interfaces. **Note:** To implement multiple interfaces, separate them with a comma

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a
body)
```

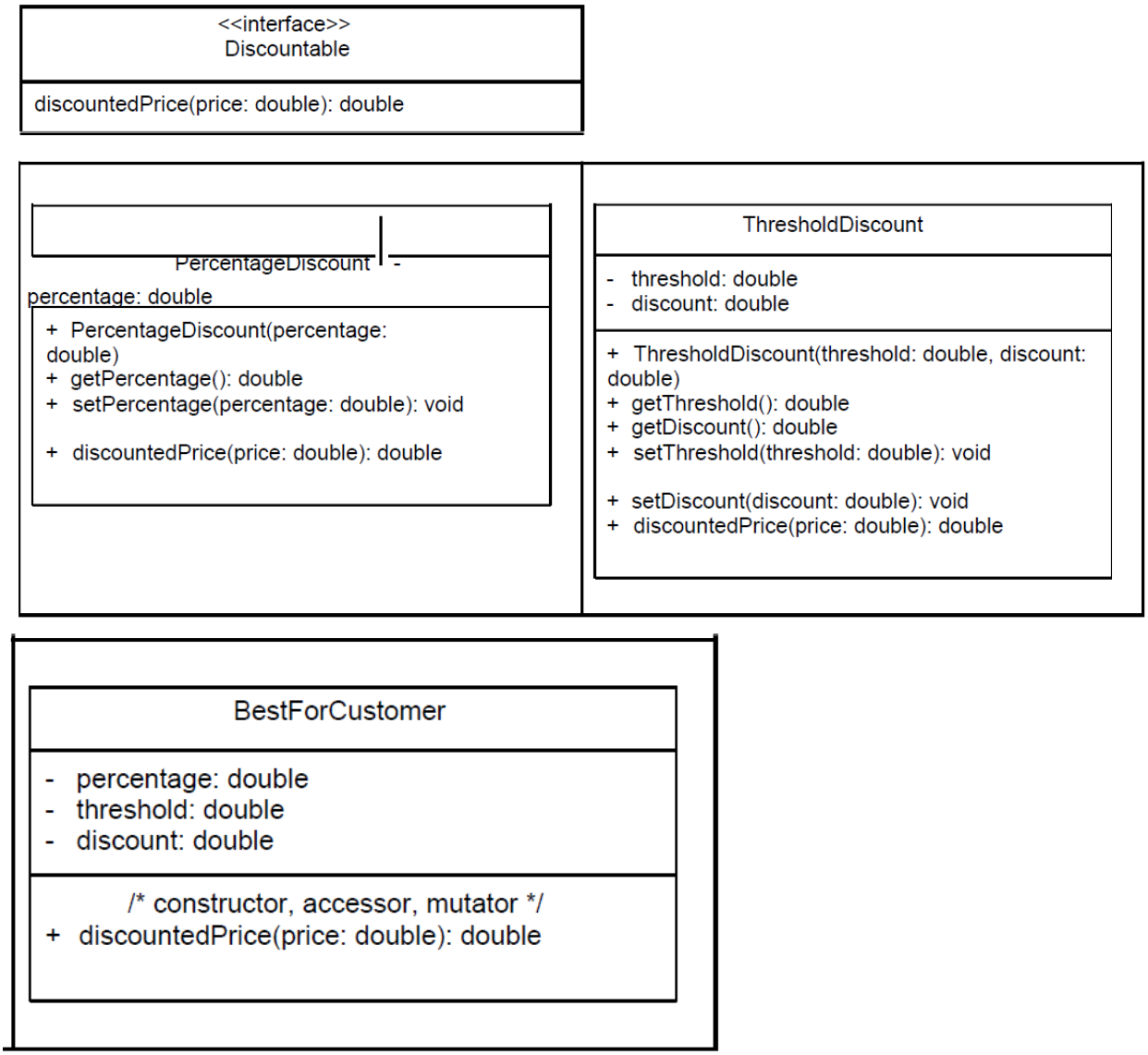
```
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal
{
  public void animalSound()
  {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep()
  {
    // The body of sleep() is provided here
    System.out.println("Pigs sleep all night long…..Zzzz");
  }
}

// Dog "implements" the Animal interface
class Pig implements Animal
{
  public void animalSound()
  {
    // The body of animalSound() is provided here
    System.out.println("The pig says: vow vow");
  }
  public void sleep()
  {
    // The body of sleep() is provided here
    System.out.println("Dogs hardly sleeps at night ");
  }
}


class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

## Difference between abstract class and interface

| Abstract class | Interface |
|---|---|
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 5) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6) An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| 9)**Example:**<br>public abstract class Shape{<br>public abstract void draw();<br>} | **Example:**<br>public interface Drawable{<br>void draw();<br>} |

# Tasks:

Implement the following classes and invoke discountedPrice() for object of each class.




discountedPrice() from BestForCustomer class will consider both percentage and threshold discount and give the customer the best possible sales price.