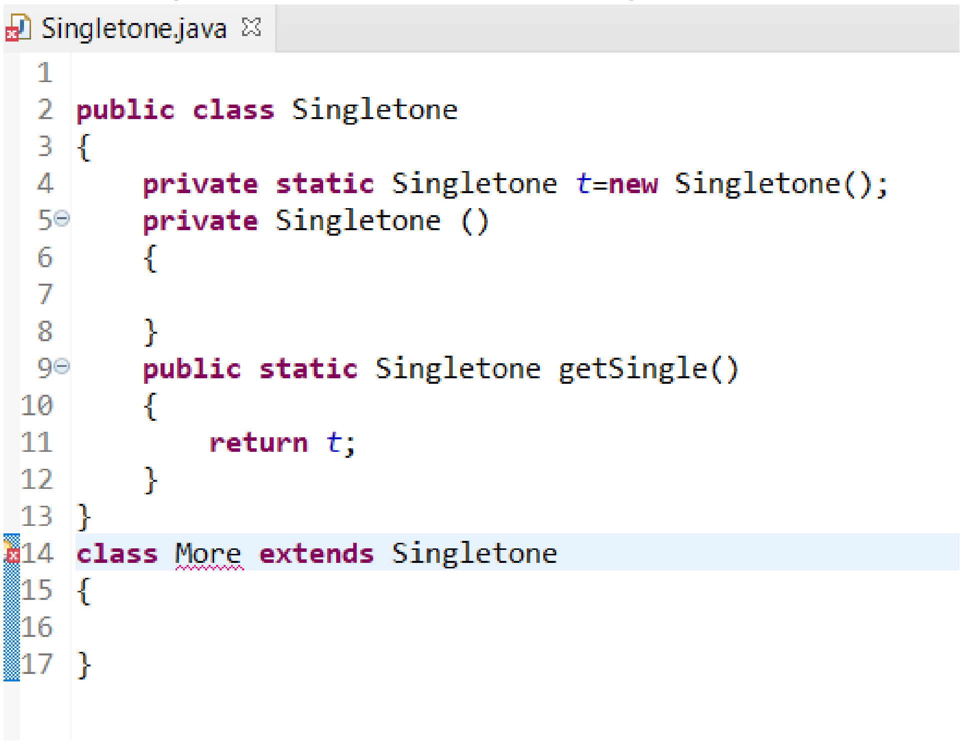
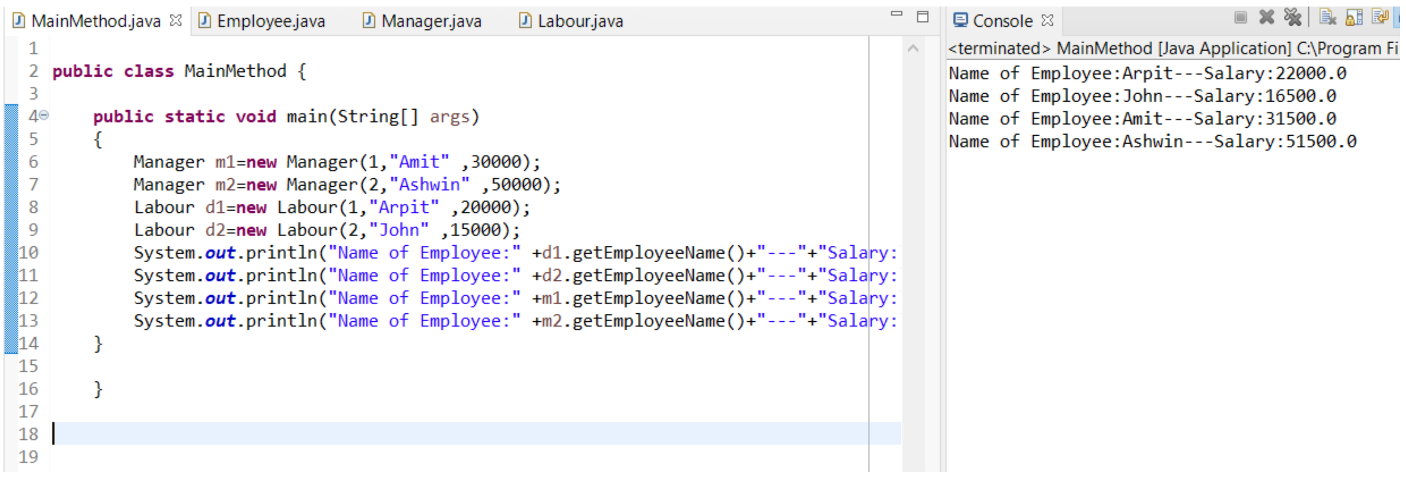
OOPS Assignment

1. Write a singletone class. Confirm that singletone class cannot be inherited.



```java

public class Singletone
{
    private static Singletone t=new Singletone();
    private Singletone ()
    {

    }
    public static Singletone getSingle()
    {
        return t;
    }
}
class More extends Singletone
{

}
```

2. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.



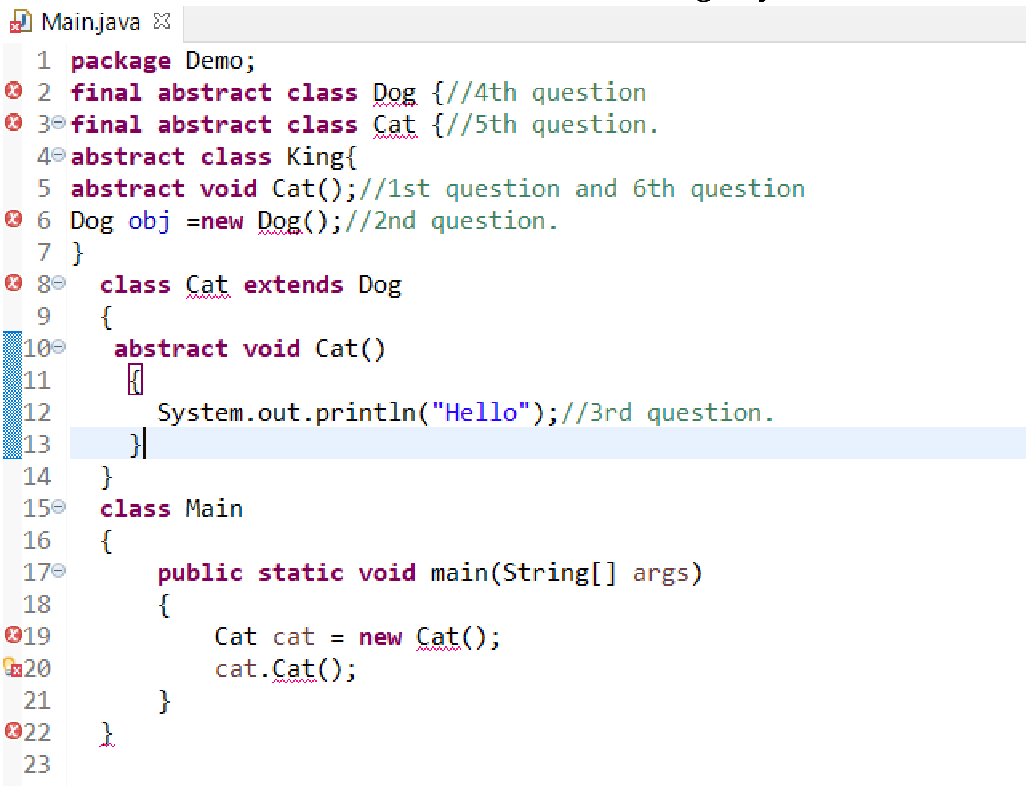
```java

public class MainMethod {

    public static void main(String[] args)
    {
        Manager m1=new Manager(1,"Amit" ,30000);
        Manager m2=new Manager(2,"Ashwin" ,50000);
        Labour d1=new Labour(1,"Arpit" ,20000);
        Labour d2=new Labour(2,"John" ,15000);
        System.out.println("Name of Employee:" +d1.getEmployeeName()+"---"+"Salary:
        System.out.println("Name of Employee:" +d2.getEmployeeName()+"---"+"Salary:
        System.out.println("Name of Employee:" +m1.getEmployeeName()+"---"+"Salary:
        System.out.println("Name of Employee:" +m2.getEmployeeName()+"---"+"Salary:
    }

    }


```

```
<terminated> MainMethod [Java Application] C:\Program Fi
Name of Employee:Arpit---Salary:22000.0
Name of Employee:John---Salary:16500.0
Name of Employee:Amit---Salary:31500.0
Name of Employee:Ashwin---Salary:51500.0
```

3. Write a program to consider saving & current account in the bank. Saving account holder has 'Fixed Deposits' whereas Current account holder has cash credit. Apply polymorphism to find out total cash in the bank.

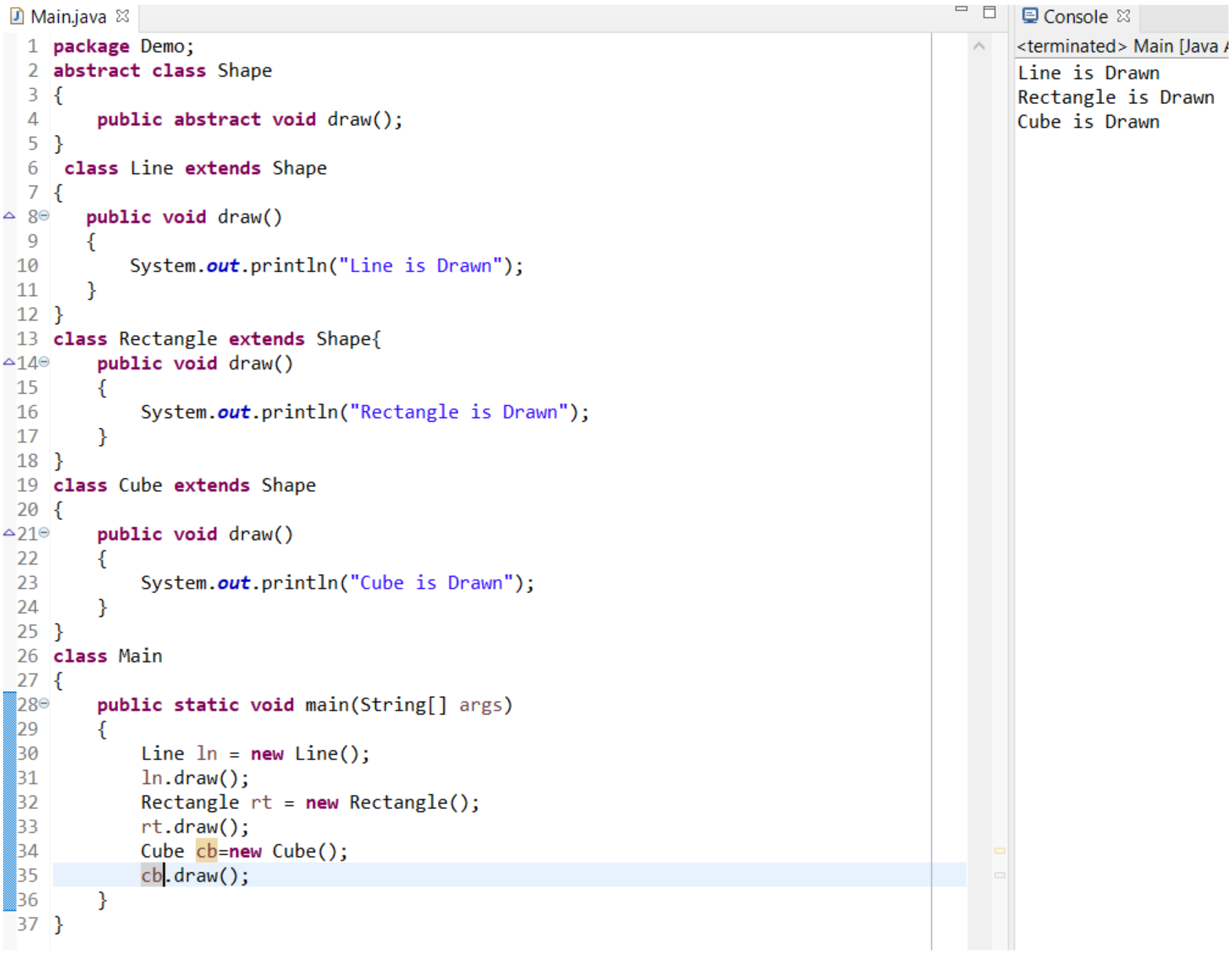
4. Test the following principles of an abstract class:

- If any class has any of its method abstract then you must declare entire class abstract.
- Abstract class cannot be instantiated.
- When we extend an abstract class, we must either override all the abstract methods in sub class or declare subclass as abstract.
- Abstract class cannot be private.
- Abstract class cannot be final.
- You can declare a class abstract without having any abstract method.



```java
package Demo;
final abstract class Dog {//4th question
final abstract class Cat {//5th question.
abstract class King{
abstract void Cat();//1st question and 6th question
Dog obj =new Dog();//2nd question.
}
  class Cat extends Dog
  {
   abstract void Cat()
    {
      System.out.println("Hello");//3rd question.
    }
  }
  class Main
  {
      public static void main(String[] args)
      {
          Cat cat = new Cat();
          cat.Cat();
      }
  }
```

5. Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.

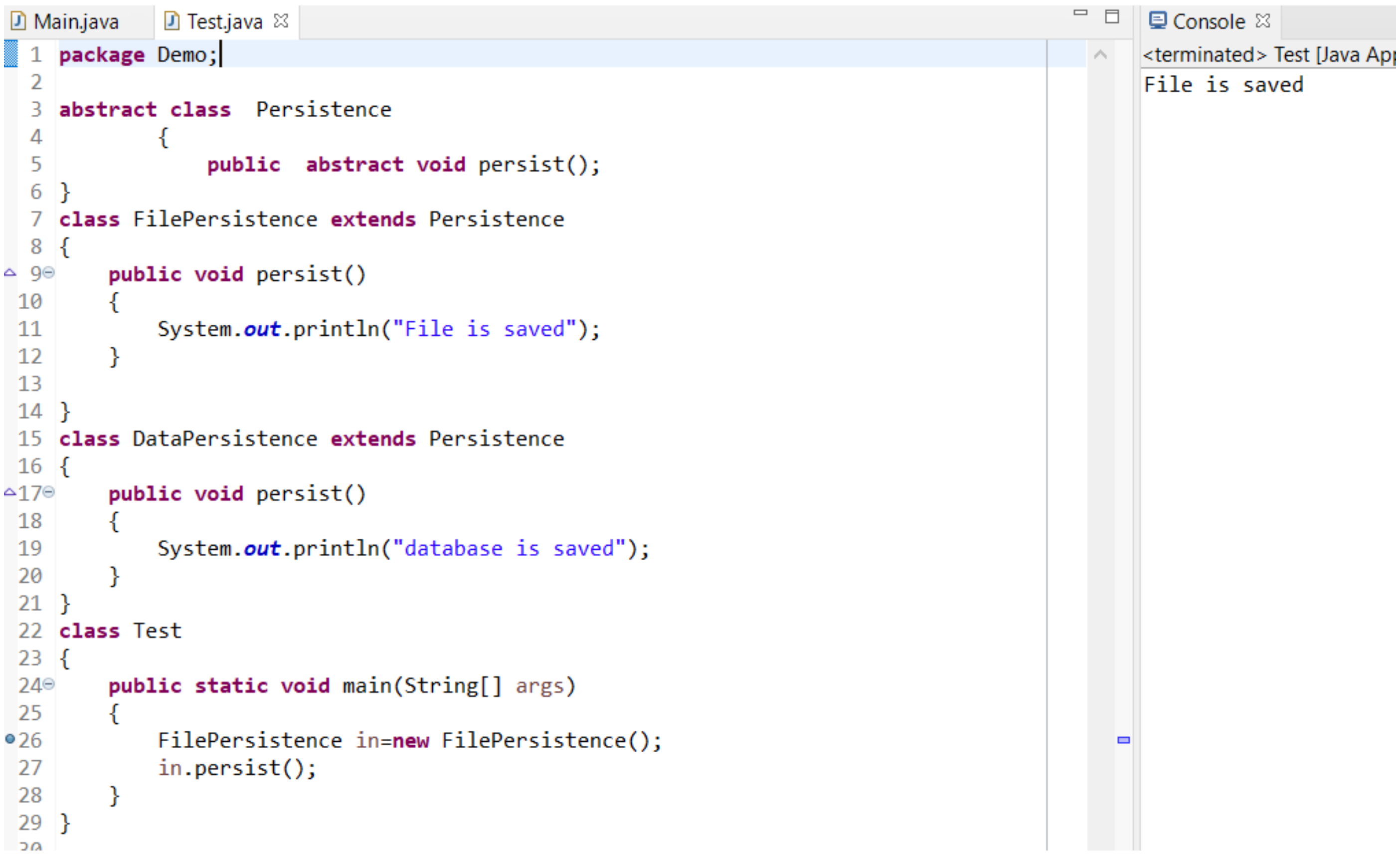

Main.java

```java
package Demo;
abstract class Shape
{
    public abstract void draw();
}
 class Line extends Shape
{
   public void draw()
   {
       System.out.println("Line is Drawn");
   }
}
class Rectangle extends Shape{
    public void draw()
    {
        System.out.println("Rectangle is Drawn");
    }
}
class Cube extends Shape
{
    public void draw()
    {
        System.out.println("Cube is Drawn");
    }
}
class Main
{
    public static void main(String[] args)
    {
        Line ln = new Line();
        ln.draw();
        Rectangle rt = new Rectangle();
        rt.draw();
        Cube cb=new Cube();
        cb.draw();
    }
}
```

Console

```
<terminated> Main [Java A
Line is Drawn
Rectangle is Drawn
Cube is Drawn
```

6. Write an abstract class 'Persistence along with two sub classes 'FilePersistence & 'DatabasePersistence'. The base class with have an abstract method persist() which will be overridden by its sub classes. Write a client who gets the Persistence object at runtime & invokes persist() method on it without knowing whether data is being saved in File or in Database.



```
package Demo;

abstract class  Persistence
        {
            public  abstract void persist();
}
class FilePersistence extends Persistence
{
    public void persist()
    {
        System.out.println("File is saved");
    }

}
class DataPersistence extends Persistence
{
    public void persist()
    {
        System.out.println("database is saved");
    }
}
class Test
{
    public static void main(String[] args)
    {
        FilePersistence in=new FilePersistence();
        in.persist();
    }
}
```

```
<terminated> Test [Java Ap
File is saved
```

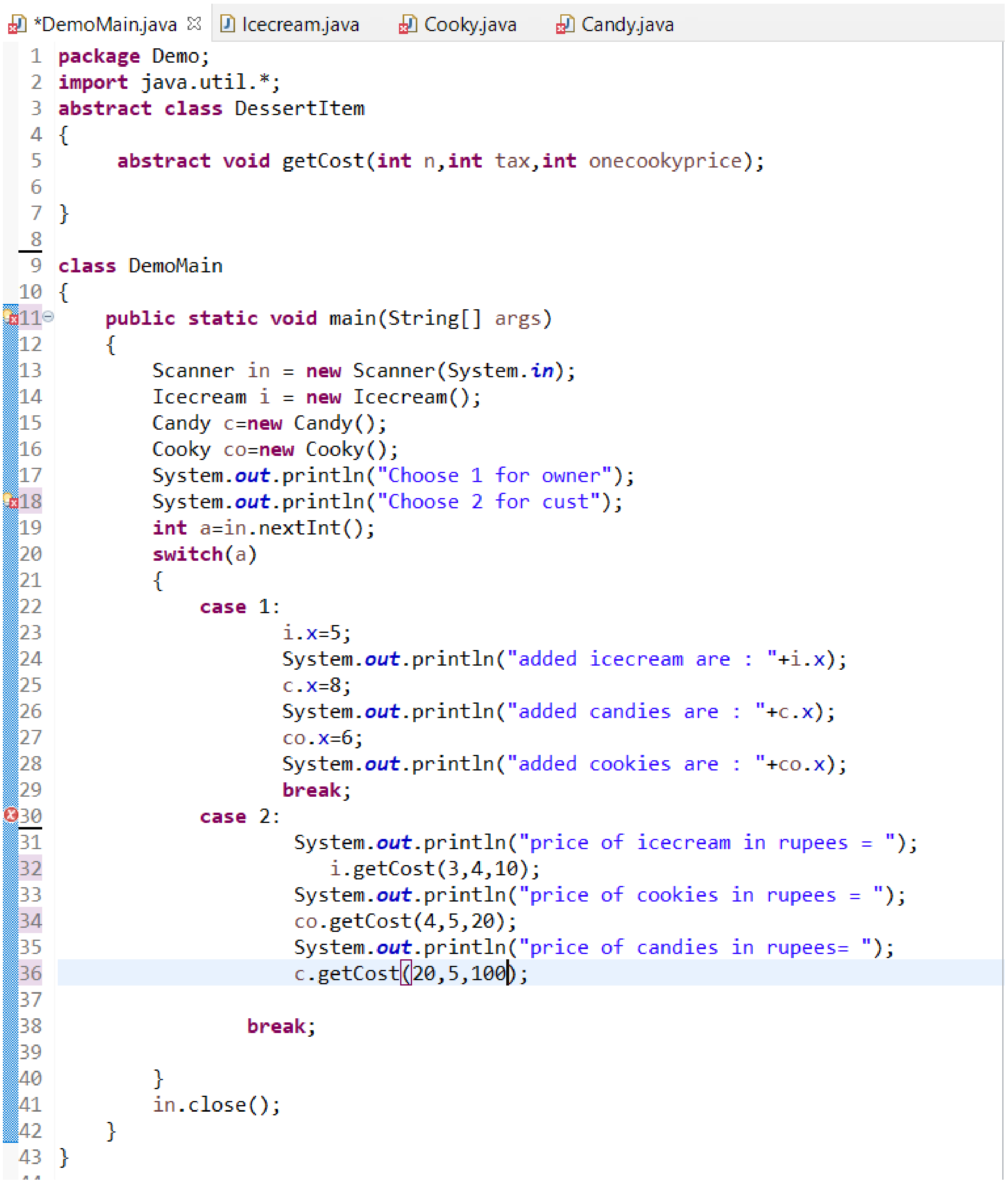
7. Develop an application for Dessert shop. The application should allow owner to add items like Candy, Cookie or IceCream in the shop storage. Also customers should be able to place an order. Dessert item is an abstract class having an abstract method getCost().

Every dessert item has tax associated.

Candy item is sold in dollar currency, Cookie in Euro currency & IceCream in Rupees currency. The sub classes are supposed to override these methods. When we run the application, it should ask us our role i.e. owner or customer. If role is owner, we should be able to add dessert items in our storage. If role is customer, then we should be able to place an order. The currency conversion rates are:

1 dollar = 60 rupees.

1 euro = 70 rupees.



```java
package Demo;
import java.util.*;
abstract class DessertItem
{
     abstract void getCost(int n,int tax,int onecookyprice);

}

class DemoMain
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Icecream i = new Icecream();
        Candy c=new Candy();
        Cooky co=new Cooky();
        System.out.println("Choose 1 for owner");
        System.out.println("Choose 2 for cust");
        int a=in.nextInt();
        switch(a)
        {
            case 1:
                   i.x=5;
                   System.out.println("added icecream are : "+i.x);
                   c.x=8;
                   System.out.println("added candies are : "+c.x);
                   co.x=6;
                   System.out.println("added cookies are : "+co.x);
                   break;
            case 2:
                    System.out.println("price of icecream in rupees = ");
                       i.getCost(3,4,10);
                    System.out.println("price of cookies in rupees = ");
                    co.getCost(4,5,20);
                    System.out.println("price of candies in rupees= ");
                    c.getCost(20,5,100);

                break;

        }
        in.close();
    }
}
```