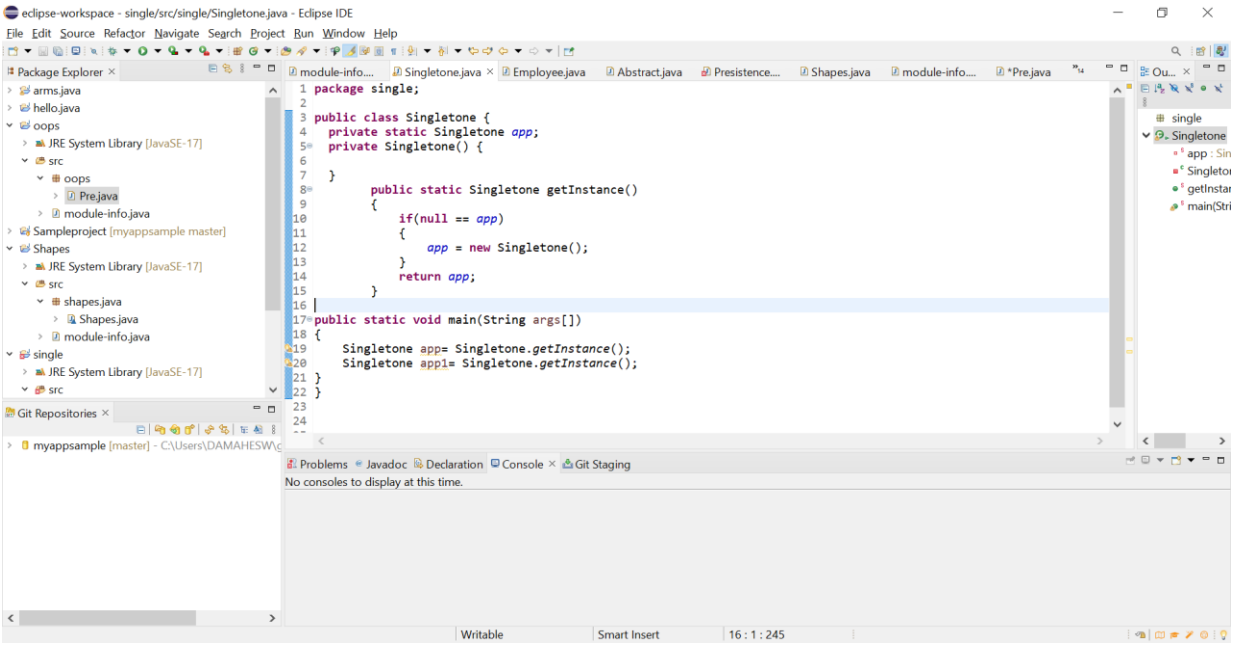
JAVA

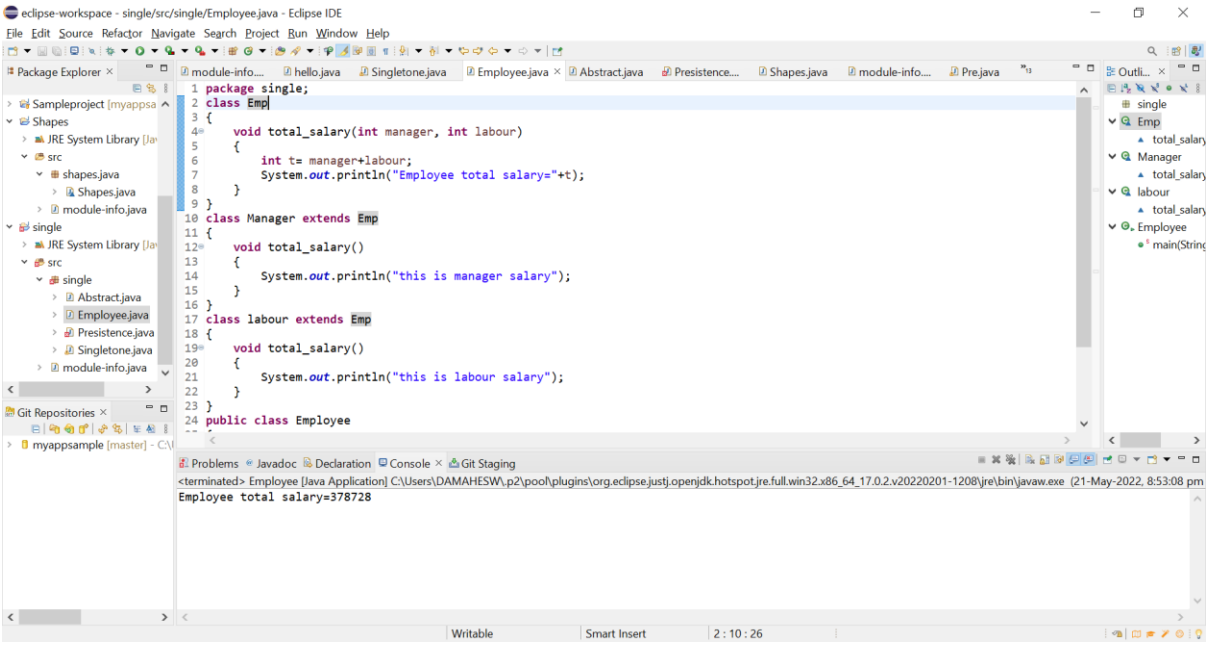
Assignement-2

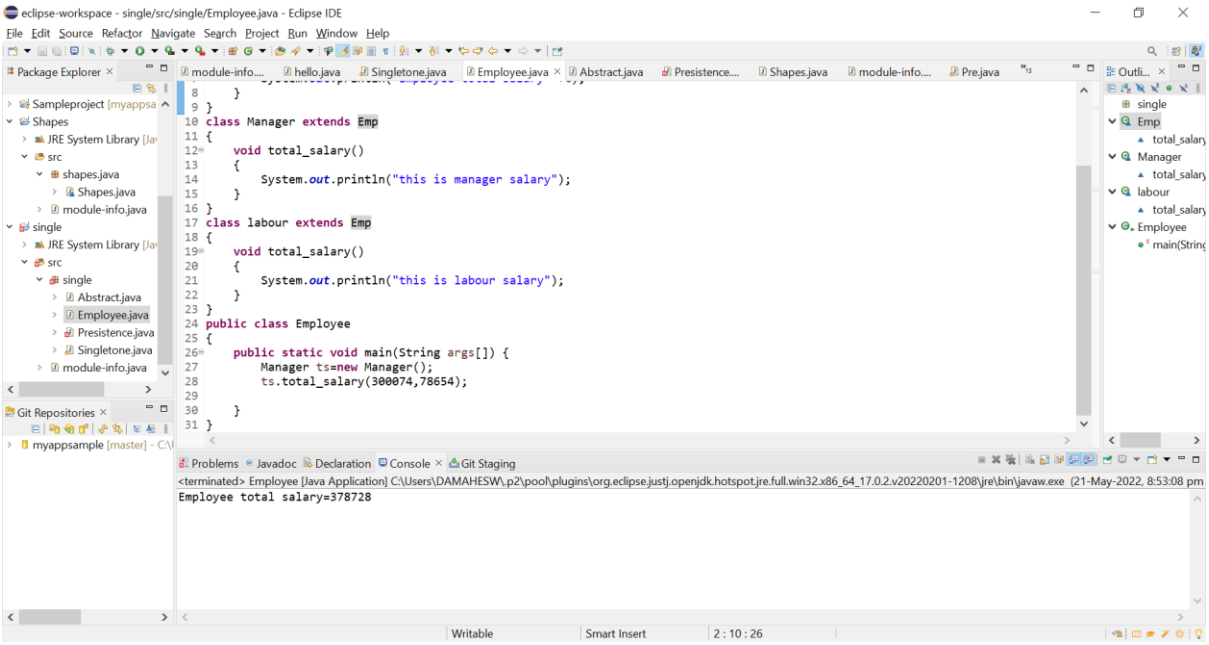
1.write a singletone class. confirm that singletone class cannot be inherited.



2)

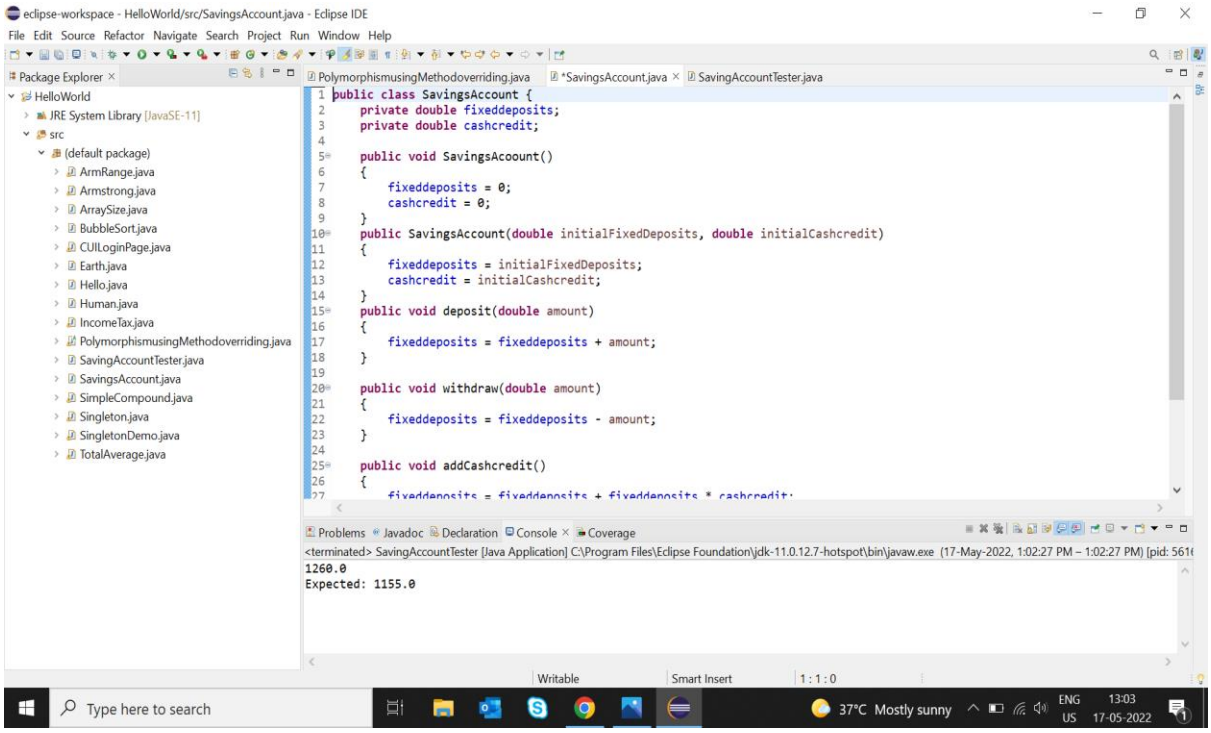
Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use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

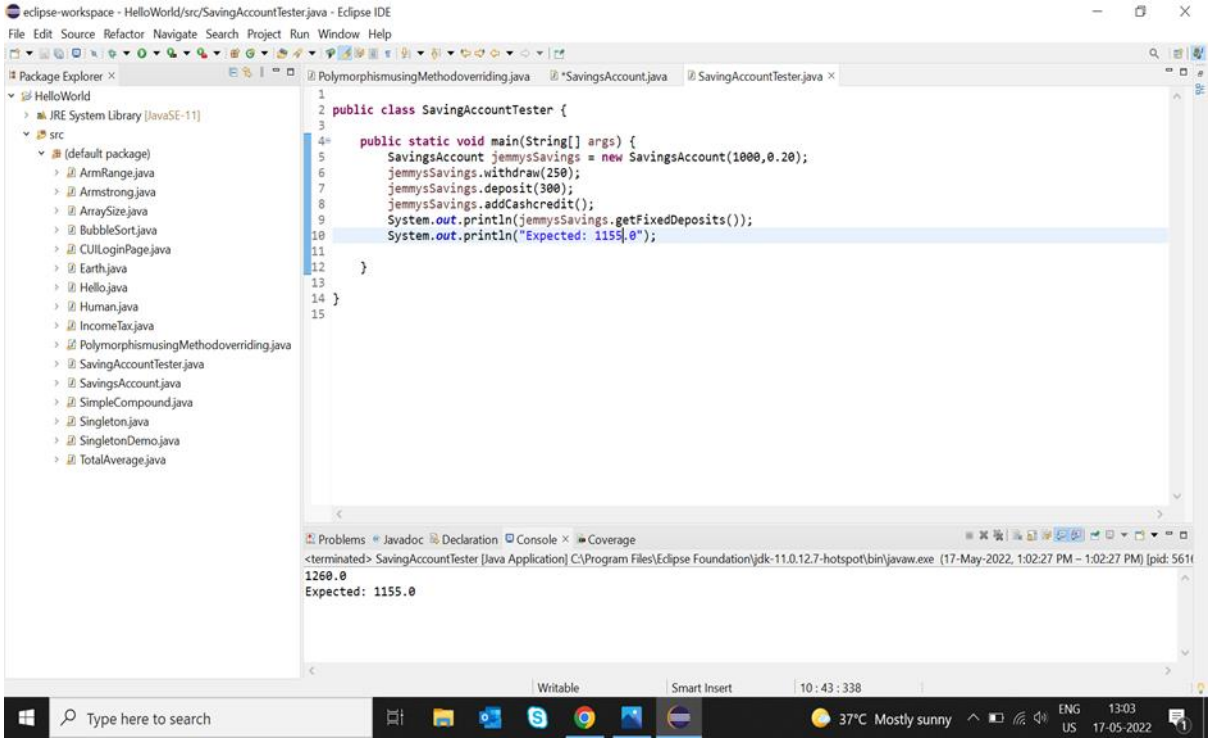




3)

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.

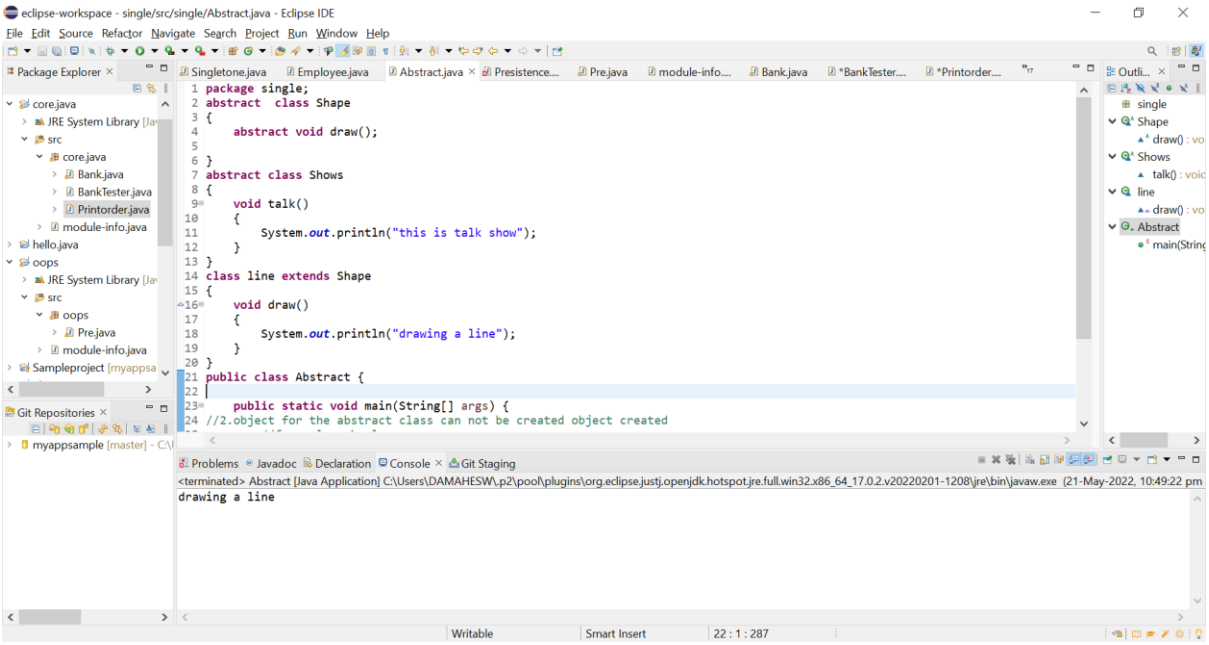


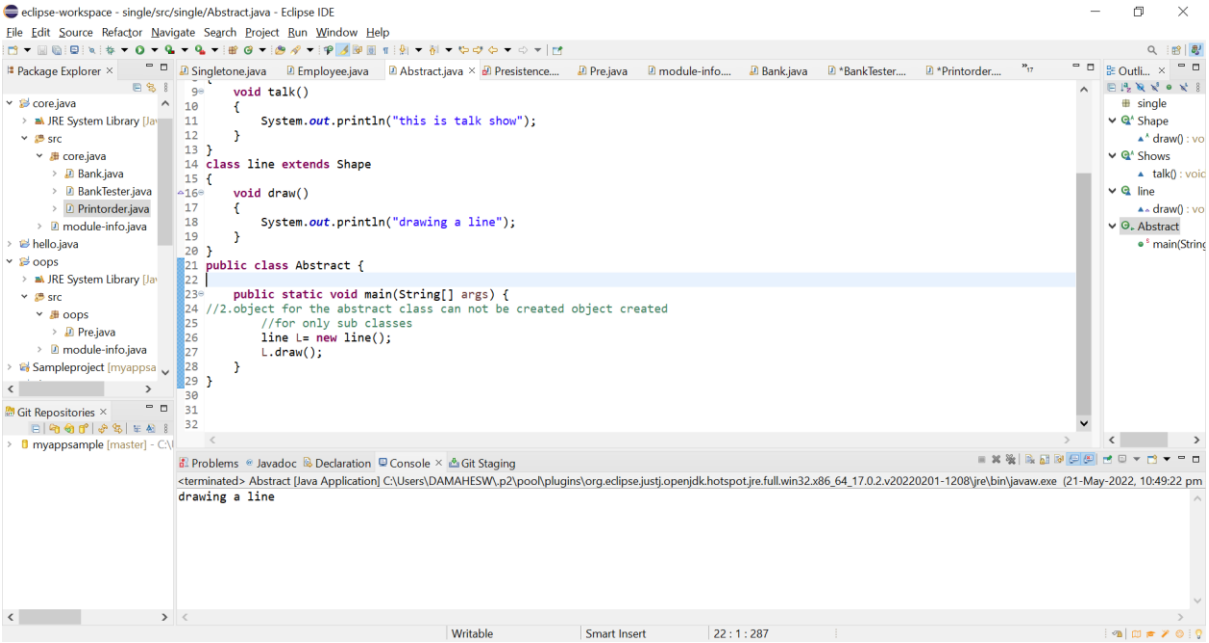


1. Test the following principles of an abstract class:

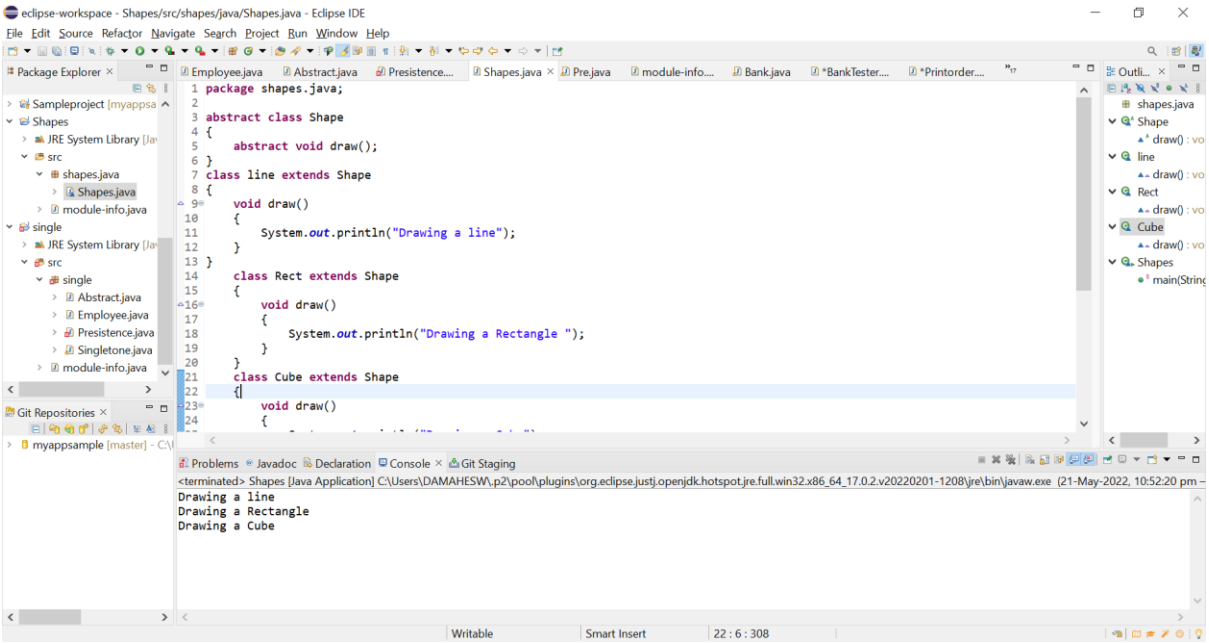
- If any class has any of its method abstract then you must declare entire class abstract
- Abstract class cannot be instantiated.
- When we extend an abstract class, we must either override all the abstract methods

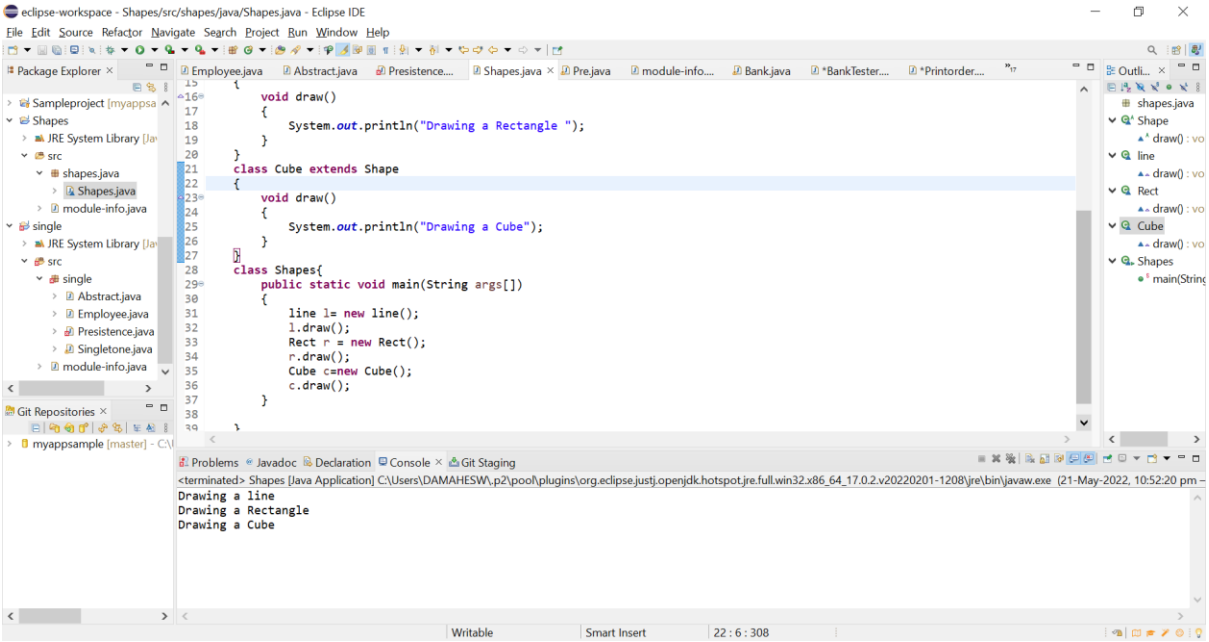
- Abstract class cannot be private.
- Abstract class cannot be final.
- You can declare a class abstract without having any abstract method.



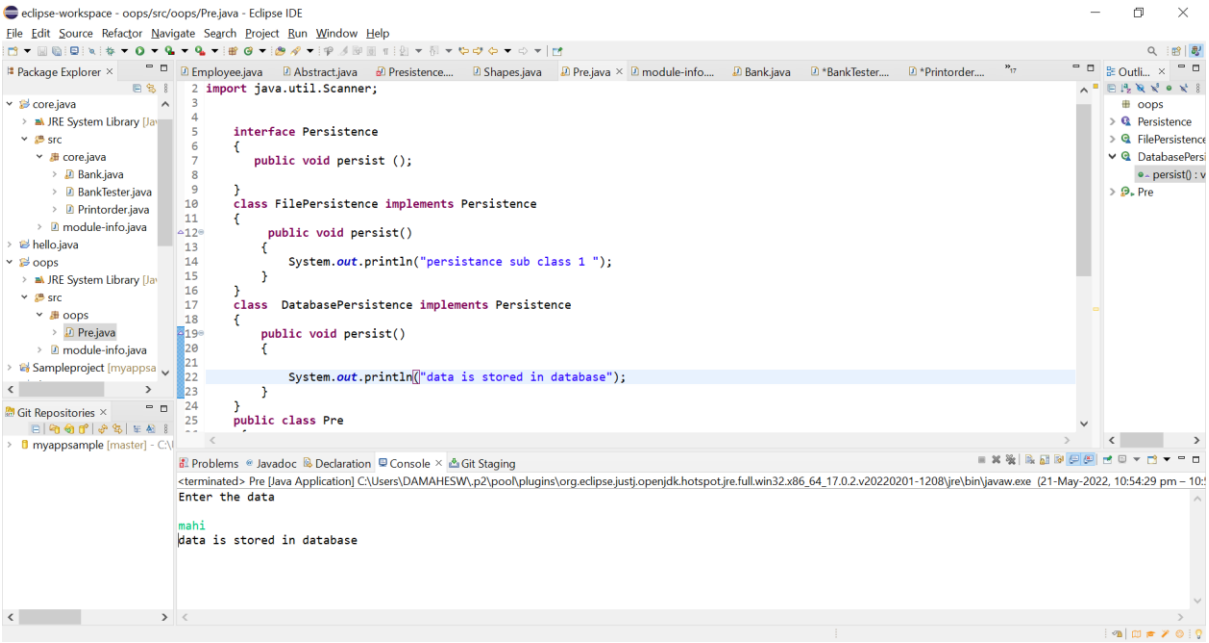


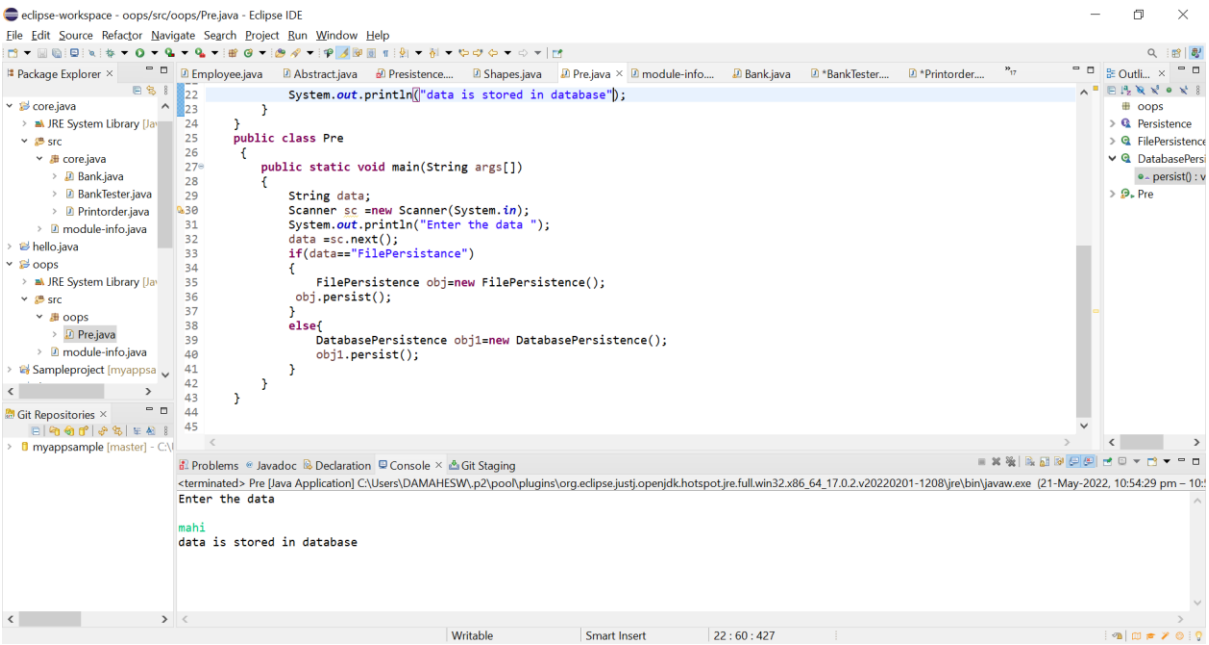
5.Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.





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.





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. DessertItem is an abstract class having an abstract methodgetCost(). 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
public class Printorder {
public static void main(String[] args) {
    Order checkout = new Order();

    checkout.enterItem(new Candy("Peanut Butter Fudge", 2.25, 399));
    checkout.enterItem(new IceCream("Vanilla Ice Cream",105));

    checkout.enterItem(new Cookie("Oatmeal Raisin Cookies", 4, 399));

    System.out.println("\nNumber of items: " + checkout.numberOfItems() + "\n");
    System.out.println("\nTotal cost: " + checkout.totalCost() + "\n");
    System.out.println("\nTotal tax: " + checkout.totalTax() + "\n");
    System.out.println("\nCost + Tax: " + (checkout.totalCost() + checkout.totalTax()) + "\n");
    System.out.println(checkout);

    checkout.clear();

    checkout.enterItem(new IceCream("Strawberry Ice Cream",145));

    checkout.enterItem(new Candy("Gummy Worms", 1.33, 89));
    checkout.enterItem(new Cookie("Chocolate Chip Cookies", 4, 399));
    checkout.enterItem(new Candy("Salt Water Taffy", 1.5, 209));
    checkout.enterItem(new Candy("Candy Corn",3.0, 109));

    System.out.println("\nNumber of items: " + checkout.numberOfItems() + "\n");
    System.out.println("\nTotal cost: " + checkout.totalCost() + "\n");
    System.out.println("\nTotal tax: " + checkout.totalTax() + "\n");
    System.out.println("\nCost + Tax: " + (checkout.totalCost() + checkout.totalTax()) + "\n");
    System.out.println(checkout);
}
}
```

```java
public abstract class DessertItem {
    protected String name;

      public DessertItem() {
        this("");
      }

      public DessertItem(String name) {
        if (name.length() <= DessertShop.MAX_ITEM_NAME_SIZE)
          this.name = name;
        else
          this.name = name.substring(0,DessertShop.MAX_ITEM_NAME_SIZE);
      }

      public final String getName() {
        return name;
      }

      public abstract int getCost();

}
```

```java
public class DessertShop {
     public final static double TAX_RATE = 6.5;
      public final static int MAX_ITEM_NAME_SIZE = 25;
      public final static int COST_WIDTH = 6;

      public static String dollars2rupeesAndollars(int dollars) {
        String s = "";


        int rupees = dollars/60;
        dollars = dollars % 100;

        if (rupees > 0)
          s += dollars;

        s +=".";

        if (rupees < 10)
          s += "0";

        s += rupees;
        return s;
      }

      public static String euro2rupeesAndeuro(int euro) {
            String s = "";


            int rupees = euro/70;
            euro = euro% 100;

            if (rupees > 0)
              s += euro;
```

```java
            s +=".";

            if (rupees < 10)
              s += "0";

            s += rupees;
            return s;
          }


      }

```

```java
public class Cookie extends DessertItem {
     protected double number;
      protected double pricePerDozen;

      public Cookie(String n, double ppd, int nb){
        super(n);
        pricePerDozen = ppd;
        number = nb;
      }

      public int getCost(){
        return  (int)Math.round(number / 12 * pricePerDozen);
      }

}

```

```java

public class Candy extends DessertItem {
     protected double weight;
      protected double pricePerPound;

      public Candy(String n, double ppp, int w){
        super(n);
        pricePerPound = ppp;
        weight = w;
      }

      public int getCost(){
        return  (int)Math.round(weight * pricePerPound);
      }

}

```


```java

public class Candy extends DessertItem {
     protected double weight;
      protected double pricePerPound;

      public Candy(String n, double ppp, int w){
        super(n);
        pricePerPound = ppp;
        weight = w;
      }

      public int getCost(){
        return  (int)Math.round(weight * pricePerPound);
      }

}

```

```java
public class Cookie extends DessertItem {
     protected double number;
      protected double pricePerDozen;

      public Cookie(String n, double ppd, int nb){
        super(n);
        pricePerDozen = ppd;
        number = nb;
      }

      public int getCost(){
        return  (int)Math.round(number / 12 * pricePerDozen);
      }

}
```

```java

public class IceCream extends DessertItem {
     protected int cost;

      public IceCream(String n, int ct){
        super(n);
        cost = ct;
      }

      public int getCost(){
        return  cost;
      }

}
```

```java

public class Order {
    protected int size;
      protected DessertItem[] dessertItems;
      protected int amount;
      protected int sum;
      protected final double taxRate;

      Order(){
        size = 100;
        dessertItems = new DessertItem[size];
        amount = 0;
        sum = 0;
        taxRate = DessertShop.TAX_RATE;
      }

      public void enterItem(DessertItem d){
        dessertItems[amount] = d;
        amount ++;
      }

      public int numberOfItems(){
        return amount;
      }

      public int totalCost(){
        sum = 0;
        for(int i = 0; i < amount; i ++){
          sum += dessertItems[i].getCost();
        }
        return sum;
      }

```

```java

    public int totalTax(){
      return (int)(Math.round(this.totalCost() * taxRate / 100));
    }

    public void clear(){
      for(DessertItem d : dessertItems){
        d = null;
      }
      amount = 0;
      sum = 0;
    }
    public String toString(){
      String result = "Thank You! \n";
    result += "Purchased: \n" ;

      String totalPay = DessertShop.dollars2rupeesAndollars( totalCost()+totalTax());
      String totalPay1 = DessertShop.euro2rupeesAndeuro( totalCost()+totalTax() );
      if(totalPay.length() > DessertShop.COST_WIDTH){
        totalPay = totalPay.substring(0, DessertShop.COST_WIDTH);

      }


      else if(totalPay1.length() > DessertShop.COST_WIDTH){
            totalPay1 = totalPay1.substring(0, DessertShop.COST_WIDTH);

}
      result += "$" + totalPay ;
      result+=     " euro"+totalPay1;
      return result;
    }
}
```

```
Number of items: 3


Total cost: 1136


Total tax: 74


Cost + Tax: 1210

Thank You!
Purchased:
$10.20 euro10.17

Number of items: 5


Total cost: 1037


Total tax: 67


Cost + Tax: 1104

Thank You!
Purchased:
$4.18 euro4.15
```