Oops Assignment

3.

```
public class Bank {


	private static String name;
	protected static int FDMoney;
	protected static int Cashcredit;
	protected int TotalMoney;

	public Bank(String name,int TotalMoney) {
		this.name=name;
		this.TotalMoney=TotalMoney;

	}
	public int getTotalMoney() {
		return FDMoney+Cashcredit;
	}
	}


public class Savings extends Bank{


	    public Savings(int AccountNo , String name, int FDAmount) {
	      super(name,FDAmount);


	    }
	    public int getMoney() {
	      return FDMoney;
	    }
	 }


public class Current extends Bank {
	public Current(int AccountNo, String name, int Cashcredit)
	{
		super(name,Cashcredit);



	    }

	public int getMoney() {
		return Cashcredit;
	 }



}

public class Main {

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

```
	{
		int AccountNo;
		Savings s1=new
Savings(AccountNo=12345678,name="Shruti",FDMoney=20000);
		Current c1=new
Current(AccountNo=92345678,name="Rohit",Cashcredit=10000);
		int tol = FDMoney+Cashcredit;

		System.out.print("The total Money in bank is:" +tol);

	}
	private static String name;
	protected static int FDMoney;
	protected static int Cashcredit;
	protected int TotalMoney;

	public Main(String name,int TotalMoney) {
		this.name=name;
		this.TotalMoney=TotalMoney;

	}
	}
```

```
The total Money in bank is:30000
```

1.

```
	class Singleton
	{
	    // static variable single_instance of type Singleton
	    private static Singleton single_instance = null;

	    // variable of type String
	    public String s;

	    // private constructor restricted to this class itself
	    private Singleton()
	    {
	        s = "Hello I am a string part of Singleton class";
	    }

	    // static method to create instance of Singleton class
	    public static Singleton getInstance()
	    {
	        if (single_instance == null)
	            single_instance = new Singleton();

	        return single_instance;
	    }
	}
```

4. abstract class ExampleOfAbstractClass

```
{
public abstract void showData();
}
public class MainClass
{
public static void main(String arg[])
{
ExampleOfAbstractClass object = new ExampleOfAbstractClass();
}
}
```

Output:

```
Exception in thread “main” java.lang.Error: Unresolved
compilation problem: Cannot instantiate the type
ExampleOfAbstractClass at
create.MainClass.main(MainClass.java:13)
```

5.
```
class Shapes
   {
      public static void main(String[] args)
      {
         Shape[] shapes = { new Circle(10, 20, 30),
                            new Rectangle(20, 30, 40, 50) };
         for (int i = 0; i < shapes.length; i++)
            shapes[i].draw();
      }
   }


class Shape
{
   void draw()
   {
   }
}

class Circle extends Shape
{
   private int x, y, r;

   Circle(int x, int y, int r)
   {
      this.x = x;
      this.y = y;
      this.r = r;
   }


   @Override
   void draw()
```

```
    {
        System.out.println("Drawing circle (" + x + ", "+ y + ", " + r + ")");
    }
}

class Rectangle extends Shape
{
    private int x, y, w, h;

    Rectangle(int x, int y, int w, int h)
    {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
    }


    @Override
    void draw()
    {
        System.out.println("Drawing rectangle (" + x + ", "+ y + ", " + w + "," +
                           h + ")");
    }




}
```

7.

```
public class DessertShoppe {

     public final static double TAX_RATE = 6.5;         // 6.5%
     public final static String STORE_NAME = "Riddhi";
     public final static int MAX_ITEM_NAME_SIZE = 25;
     public final static int COST_WIDTH = 6;

     public static String cents2dollarsAndCents(int cents) {
         String s = "";

         if (cents < 0) {
             s += "-";
             cents *= -1;
         }

         int dollars = cents/100;
         cents = cents % 100;

         if (dollars > 0)
             s += dollars;

         s +=".";
```

```
        if (cents < 10)
            s += "0";

        s += cents;
        return s;
    }
}


public abstract class DessertItem {

    protected String name;

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

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

    public final String getName() {
        return name;
    }

    public abstract int getCost();
}


public class Cookie extends DessertItem{

    protected double number;
    protected double pricePerDoze;

    public Cookie(String _n, double _ppd, int _number){
        super(_n);
        pricePerDoze = _ppd;
        number = _number;
    }

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


public class Candy extends DessertItem{

    protected double weight;
    protected double pricePerPound;

    public Candy(String _n, double _ppp, int _w){
        //using parent's constructor with name while storing its own properties
        super(_n);
        pricePerPound = _ppp;
        weight = _w;
```

```
    }

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


public class IceCream extends DessertItem{

    protected int cost;

    public IceCream(String _n, int _cost){
        super(_n);
        cost = _cost;
    }

    public int getCost(){
        return  cost;
    }
}


public class Sundae extends IceCream{

    protected String topName;
    protected int topCost;

    public Sundae(String _n0, int _cost0, String _n1, int _cost1){
        //put the icecream name in icecream while putting top name and cost in a
separate property
        super(_n0, _cost0);
        topName = _n1;
        topCost = _cost1;
    }

    public final String getName(){
        //return both the icecream name and the topping name
        return name + " " + topName;
    }

    public int getCost(){
        //return the sum of the icecream and the topping
        return cost + topCost;
    }
}


public class Checkout{

    protected int size;
    protected DessertItem[] dessertItems;
    protected int amount;
    protected int sum;
    protected final double taxRate;

    Checkout(){
        size = 100;
        dessertItems = new DessertItem[size];
```

```
        amount = 0;
        sum = 0;
        taxRate = DessertShoppe.TAX_RATE;
    }

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

    public int numberOfItems(){
        return amount;
    }

    public int totalCost(){
        //make sum into zero, and calculate price from every item
        sum = 0;
        for(int i = 0; i < amount; i ++){
            sum += dessertItems[i].getCost();
        }
        return sum;
    }

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

    public void clear(){
        //clear the array
        for(DessertItem d : dessertItems){
            d = null;
        }
        amount = 0;
        sum = 0;
    }

    public String toString(){
        String result = "Thank You! \n";

        result += DessertShoppe.STORE_NAME + "\n";

        result += "Purchased: ";

        String totalPay = DessertShoppe.cents2dollarsAndCents(
totalCost()+totalTax() );
        if(totalPay.length() > DessertShoppe.COST_WIDTH){
            totalPay = totalPay.substring(0, DessertShoppe.COST_WIDTH);
        }
        result += "$" + totalPay;
        return result;
    }
}


public class TestCheckout {

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

```
        Checkout checkout = new Checkout();

        checkout.enterItem(new Candy("Peanut Butter Fudge", 2.25, 399));
        checkout.enterItem(new IceCream("Vanilla Ice Cream",105));
        checkout.enterItem(new Sundae("Choc. Chip Ice Cream",145, "Hot Fudge",
50));
        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 Sundae("Vanilla Ice Cream",105, "Caramel", 50));
        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);
    }
}
```

2.

```java
public class Employee{

    public static class Manager extends Employee {
        public static final double Bonusrate=0.2;
        public Manager(int employeeid,String employeename,double incentive) {
            super(employeeid,employeename,incentive);
        }
            public double getSalary() {
                return salary+salary*Bonusrate;
            }


    }

    public static class Labour extends Employee {
        public static final double Bonus=0.1;

        public Labour(int employeeid,String employeename,double salary){
            super(employeeid,employeename,salary);
            }

    public double getsalary() {
        return salary+salary*Bonus;
    }

    }

    int employeeid;
    String employeename;
    double salary;

    public Employee(int employeeid,String employeename,double salary) {
        super();
        this.employeeid=employeeid;
        this.employeename=employeename;
        this.salary=salary;
```

```java
    public int getEmployeeid() {
        return employeeid;
    }

    public void setEmployeeid(int employeeid) {
        this.employeeid = employeeid;
    }

    public String getEmployeename() {
        return employeename;
    }

    public void setEmployeename(String employeename) {
        this.employeename = employeename;
    }

    public double getsalary() {
        return salary;
    }
}
```

```java
public class Overridding {

    public static void main(String[] args) {
        Employee.Labour l1=new Employee.Labour(23,"ramesh",2000);
        Employee.Labour l2=new Employee.Labour(24,"suresh",3000);
        Employee.Manager m1=new Employee.Manager(34,"nick",5000);
        Employee.Manager m2=new Employee.Manager(24,"jayesh",6000);

        System.out.println("Name of Employee"+l1.getEmployeename()+"-----"+"Salary"+l1.getsalary());
        System.out.println("Name of Employee"+l2.getEmployeename()+"-----"+"Salary"+l2.getsalary());
        System.out.println("Name of Employee"+m1.getEmployeename()+"-----"+"Salary"+m1.getsalary());
        System.out.println("Name of Employee"+m2.getEmployeename()+"-----"+"Salary"+m2.getsalary());
    }

}

```

Problems Javadoc Declaration Console

<terminated> Overridding [Java Application] C:\Users\Ajay\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\ja

```
Name of Employeeramesh-----Salary2200.0
Name of Employeesuresh-----Salary3300.0
Name of Employeenick-----Salary5000.0
Name of Employeejayesh-----Salary6000.0
```