OOPS

2

```
package oops;

public class Employee {
    int employeeId;
    String employeeName;
    double salary;

    public Employee(int employeeId, String employeeName, double salary) {
        super();
        this.employeeId = employeeId;
        this.employeeName = employeeName;
        this.salary = salary;
    }

    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;
    }
    public static class Manager extends Employee{

        public static final double Bonus=0.3;
        public Manager(int employeeId, String employeeName, double incentive) {
            super(employeeId, employeeName, incentive);
        }
        public double getSalary() {
            return salary+salary*Bonus;
        }
    }
    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;
        }
    }
}
```

&

```
package oops;

public class MethodOverridingMain {

	public static void main(String[] args) {
		Employee.Labour l1=new Employee.Labour(23,"rutuja" ,2200);
	  Employee.Labour l2=new Employee.Labour(24,"himani" ,1100);
	  Employee.Manager m1=new Employee.Manager(1,"priyank" ,15000);
	  Employee.Manager m2=new Employee.Manager(2,"kirti" ,10000);

	  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());


	}

}
```

```
Name of Employee:rutuja, Salary:2420.0
Name of Employee:himani, Salary:1210.0
Name of Employee:priyank, Salary:19500.0
Name of Employee:kirti, Salary:13000.0
```

3

```
package oops;

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;
}
}

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

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

```
 }
}

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


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

public class Main1 {
	public static void main(String[] args)
{
		int AccountNo;
		Savings s1=new
		Savings(AccountNo=12345678,name="sonal",FDMoney=100000);
		Current c1=new
		Current(AccountNo=92873478,name="rajan",Cashcredit=300000);
		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 Main1(String name,int TotalMoney) {
		this.name=name;
		this.TotalMoney=TotalMoney;
}
}
```

```
The total Money in bank is:400000
```

4

```
package oops;

abstract class ExampleOfAbstractClass
{
	public abstract void showData();
}
public class MainClass {

	public static void main(String[] args) {
		ExampleOfAbstractClass object = new ExampleOfAbstractClass();

	}
```

```
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot instantiate the type ExampleOfAbstractClass

        at oops.MainClass.main(MainClass.java:10)
```

5

```
package oops;

class Shapes
{
	public static void main(String[] args)
{
		Shape[] shapes = { new Circle(1, 2, 3),
		new Rectangle(20, 25, 10, 40) };
		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 a 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 a rectangle (" + x + ", "+ y + ", " + w + "," +
h + ")");
}




}
```

```
<terminated> Shapes [Java Application] C:\Program Fi
Drawing a circle (1, 2, 3)
Drawing a rectangle (20, 25, 10,40)
```

7

```
package oops;

public class DessertShop {

    public final static double TAX_RATE = 6;
    public final static String STORE_NAME = "Rutuja Dessert Shop";
    public final static int MAX_ITEM_NAME_SIZE = 30;
    public final static int COST_WIDTH = 5;

    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;
    }
}


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 String getName() {
        return name;
    }

    public abstract int getCost();
}


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);
    }
}


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);
    }
}


class IceCream extends DessertItem{

    protected int cost;

    public IceCream(String _n, int _cost){
```

```
        super(_n);
        cost = _cost;
    }

    public int getCost(){
        return  cost;
    }
}

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;
    }
}

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 = DessertShop.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, Visit Again!! \n";

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

        result += "Purchased: ";

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


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);
    }
}
```

```
Number of items: 4


Total cost: 1331


Total tax: 80


Cost + Tax: 1411
```