Assignment 01

Problem Statement :

1. Create an abstract class Product with the following properties and methods:

Properties: productId (String), productName (String), price (double)

abstract void displayDetails() - This method should be implemented by subclasses to display the details of the product.

Create two subclasses of Product:

• Book:

1. Additional Property: author (String)

2. Implement the displayDetails() method to display all the details of the book.

• Electronics:

1. Additional Property: warrantyPeriod (int)

2. Implement the displayDetails() method to display all the details of the electronic product.

In the Main class, create instances of Book and Electronics, set their properties, and display their Detail.

Code :

```
abstract class Product {
protected String productId;
protected String productName;
protected double price;

public Product(String productId, String productName, double price) {
this.productId = productId;
this.productName = productName;
this.price = price;
```

```
}

public abstract void displayDetails();
}

class Book extends Product {
private String author;

public Book(String productId, String productName, double price, String author) {
super(productId, productName, price);
this.author = author;
}

@Override
public void displayDetails() {
System.out.println("Book Details:");
System.out.println("Product ID: " + productId);
System.out.println("Product Name: " + productName);
System.out.println("Price: " + price);
System.out.println("Author: " + author);
}
}

class Electronics extends Product {
private int warrantyPeriod;

public Electronics(String productId, String productName, double price, int warrantyPeriod) {
super(productId, productName, price);
```

```
this.warrantyPeriod = warrantyPeriod;
}

@Override
public void displayDetails() {
System.out.println("Electronics Details:");
System.out.println("Product ID: " + productId);
System.out.println("Product Name: " + productName);
System.out.println("Price: " + price);
System.out.println("Warranty Period: " + warrantyPeriod + " months");
}
}

public class aMainProducts {
public static void main(String[] args) {
Book book = new Book("B01", "Wings Of Fire", 100.99, "A. P. J. Abdul Kalam");
book.displayDetails();

Electronics electronics = new Electronics("E01", "Smartphone", 10000.99, 24);
electronics.displayDetails();
}
}
```

Output :

```
Book Details:
Product ID: B01
Product Name: Wings Of Fire
Price: 100.99
Author: A. P. J. Abdul Kalam
Electronics Details:
Product ID: E01
Product Name: Smartphone
Price: 10000.99
Warranty Period: 24 months
```

Assignment 02

Problem Statement :

2. You are tasked with designing a system to manage different types of engineers in a company. Create an interface named Engineer with the following methods:
• void work() - Represents the work performed by an engineer.
Create two classes implementing the Engineer interface:
SoftwareEngineer:
• Properties: name (String), experienceYears(int), programmingLanguage (String)
• Implement the work() method to represent software engineering tasks such as coding and debugging.
HardwareEngineer:
• Properties: name (String), experienceYears (int), specializedArea (String)
• Implement the work() method to represent hardware engineering tasks such as designing and testing circuits.
In the Main class, create instances of both SoftwareEngineer and HardwareEngineer, invoke their work() method's, and display the tasks performed by each engineer along with their properties.

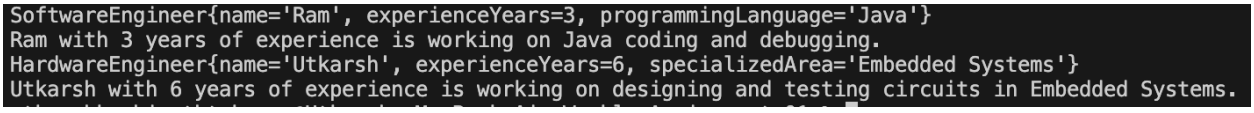
Code :

```
interface Engineer {

void work();

}


class SoftwareEngineer implements Engineer {

private String name;

private int experienceYears;

private String programmingLanguage;


public SoftwareEngineer(String name, int experienceYears, String programmingLanguage) {

this.name = name;

this.experienceYears = experienceYears;

this.programmingLanguage = programmingLanguage;
```

```
}

@Override
public void work() {
System.out.println(name + " with " + experienceYears + " years of experience is working on " + programmingLanguage + " coding and debugging.");
}

@Override
public String toString() {
return "SoftwareEngineer{name='" + name + "', experienceYears=" + experienceYears + ", programmingLanguage='" + programmingLanguage + "'}";
}
}

class HardwareEngineer implements Engineer {
private String name;
private int experienceYears;
private String specializedArea;

public HardwareEngineer(String name, int experienceYears, String specializedArea) {
this.name = name;
this.experienceYears = experienceYears;
this.specializedArea = specializedArea;
}

@Override
public void work() {
```

```
System.out.println(name + " with " + experienceYears + " years of experience is working on designing and testing circuits in " + specializedArea + ".");

}

@Override

public String toString() {

return "HardwareEngineer{name='" + name + "', experienceYears=" + experienceYears + ", specializedArea='" + specializedArea + "'}";

}

}

public class aMainProducts {

public static void main(String[] args) {

Book book = new Book("B01", "Wings Of Fire", 100.99, "A. P. J. Abdul Kalam");

book.displayDetails();

Electronics electronics = new Electronics("E01", "Smartphone", 10000.99, 24);

electronics.displayDetails();

}

}
```

Output :



Assignment 03

Problem Statement :

3. You are tasked with designing a system to manage cricket players who can be batsmen, bowlers,

or all-rounders. Each player has a name and specific skill levels in batting and bowling. Design

the following entities:

Interfaces:

• Batsman with a method void bat() representing batting actions.

• Bolwer with a method void bowl() representing bowling actions.

Class:

• AllRounder class representing a cricket player who can both bat and bowl. This class

should implement both Batsman and Bowler interfaces and have properties name,

battingSkill, and bowlingSkill

In the Main class, create instances of AllRounder, set their properties, and display the actions

performed by each player along with their properties.

Code :

```
interface Batsman {
void bat();
}

interface Bowler {
void bowl();
}

class AllRounder implements Batsman, Bowler {
private String name;
```

```
private int battingSkill;
private int bowlingSkill;

public AllRounder(String name, int battingSkill, int bowlingSkill) {
this.name = name;
this.battingSkill = battingSkill;
this.bowlingSkill = bowlingSkill;
}

public String getName() {
return name;
}

public int getBattingSkill() {
return battingSkill;
}

public int getBowlingSkill() {
return bowlingSkill;
}

@Override
public void bat() {
System.out.println(name + " is batting with a skill level of " + battingSkill);
}

@Override
public void bowl() {
```

```
System.out.println(name + " is bowling with a skill level of " + bowlingSkill);
}
}

public class cMainCPMS {
public static void main(String[] args) {
// Creating instances of AllRounder
AllRounder player1 = new AllRounder("Player One", 85, 75);
AllRounder player2 = new AllRounder("Player Two", 78, 82);

System.out.println("Player: " + player1.getName());
System.out.println("Batting Skill: " + player1.getBattingSkill());
System.out.println("Bowling Skill: " + player1.getBowlingSkill());
player1.bat();
player1.bowl();

System.out.println();

System.out.println("Player: " + player2.getName());
System.out.println("Batting Skill: " + player2.getBattingSkill());
System.out.println("Bowling Skill: " + player2.getBowlingSkill());
player2.bat();
player2.bowl();
}
}
```

Output :

```
Player: Player One
Batting Skill: 85
Bowling Skill: 75
Player One is batting with a skill level of 85
Player One is bowling with a skill level of 75

Player: Player Two
Batting Skill: 78
Bowling Skill: 82
Player Two is batting with a skill level of 78
Player Two is bowling with a skill level of 82
```

Assignment 04

Problem Statement :

4. You are tasked with designing a system to manage employees' salaries. Create the following entities:

• Interface:

1. Salary Calculator interface with a method double calculateSalary()

representing calculating the salary of an employee.

• In the Main class, create an instance of SalaryCalculator using an anonymous inner class to calculate and print the salary of an employee.

Code :

```
interface SalaryCalculator {
double calculateSalary();
}

public class dMaiEmpSalMS {
public static void main(String[] args) {

SalaryCalculator employeeSalaryCalculator = new SalaryCalculator() {
@Override
public double calculateSalary() {
double baseSalary = 50000.0;
double bonus = 5000.0;
return baseSalary + bonus;
}
};
```

```
double salary = employeeSalaryCalculator.calculateSalary();
System.out.println("The salary of the employee is: Rs" + salary);
}
}
```

Output :

Assignment 5

Problem Statement :

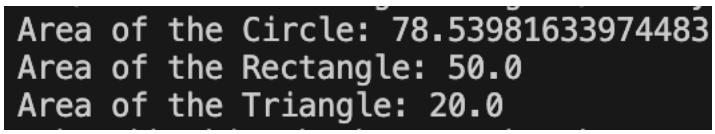
5. You are tasked with designing a program to calculate the area of different shapes using lambda
expressions. Create a Java program that performs the following tasks:
• Define a functional interface Shape with a method double calculateArea()
representing calculating the area of a shape.
• Implement lambda expressions for the following shapes:
1. Circle: Area = π * radius ^2
2. Rectangle: Area = length * width
3. Triangle: Area = 0.5 * base * height
In the Main class, create instances of Shape using lambda expressions for each shape.
Calculate and display the area of each shape.

Code :

```
@FunctionalInterface
interface Shape {
    double calculateArea();
}


public class eMainShapeAreaCal {
    public static void main(String[] args) {
        Shape circle = () -> {
            double radius = 5;
            return Math.PI * Math.pow(radius, 2);
        };

        Shape rectangle = () -> {
            double length = 10;
            double width = 5;
```

```
        return length * width;
    };

    Shape triangle = () -> {
        double base = 8;
        double height = 5;
        return 0.5 * base * height;
    };

    System.out.println("Area of the Circle: " + circle.calculateArea());
    System.out.println("Area of the Rectangle: " + rectangle.calculateArea());
    System.out.println("Area of the Triangle: " + triangle.calculateArea());
  }
}
```

Output :



Assignment 06

Problem Statement :

6. You are tasked with designing a program to model a school's student and teacher information. Create the following entities:

Outer Class:

• School class representing the school. It should contain:

• An instance variable name to store the school's name.

• A static inner class Student to represent student information. Each student should have a name and a roll number.

• A static inner class Teacher to represent teacher information. Each teacher should have a name and a subject they teach.

• Methods to display student and teacher information.

In the Main class:

• Create an instance of School.

• Create instances of Student and Teacher using the static inner classes.

• Set their information.

• Display the student and teacher information.

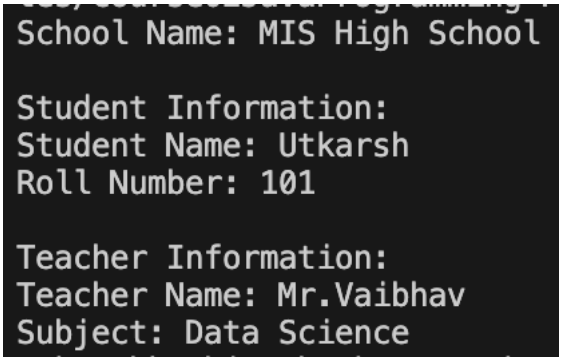
Code :

```
class School {
private String name;

School(String name) {
this.name = name;
}

static class Student {
```

```
private String name;
private int rollNumber;

Student(String name, int rollNumber) {
this.name = name;
this.rollNumber = rollNumber;
}

void displayInfo() {
System.out.println("Student Name: " + name);
System.out.println("Roll Number: " + rollNumber);
}
}

static class Teacher {
private String name;
private String subject;

Teacher(String name, String subject) {
this.name = name;
this.subject = subject;
}

void displayInfo() {
System.out.println("Teacher Name: " + name);
System.out.println("Subject: " + subject);
}
}
```

```
void displaySchoolInfo() {
System.out.println("School Name: " + name);
}
}

public class fMainSchoolMS {
public static void main(String[] args) {
School school = new School("MIS High School");

School.Student student = new School.Student("Utkarsh", 101);
School.Teacher teacher = new School.Teacher("Mr.Vaibhav", "Data Science");

school.displaySchoolInfo();

System.out.println("\nStudent Information:");
student.displayInfo();

System.out.println("\nTeacher Information:");
teacher.displayInfo();
}
}
```

Output :


School Name: MIS High School
Student Information:
Student Name: Utkarsh
Roll Number: 101
Teacher Information:
Teacher Name: Mr.Vaibhav
Subject: Data Science

Assignment 07

Problem Statement :

7. You are tasked with designing a program to model a mobile phone. Create the following entities:

Outer Class:

Phone class representing a mobile phone. It should contain:

• An instance variable brand to store the phone's brand.

• An instance variable model to store the phone's model.

• An inner class Battery to represent the phone's battery. Each battery should have a type (e.g., lithium-ion, nickel-cadmium) and a capacity (in mAh).

• An inner class Screen to represent the phone's screen. Each screen should have a size (in inches) and resolution (e.g., 1920x1080).

• Methods to display phone information, battery information, and screen information.

In the Main class:

• Create an instance of Phone.

• Set the phone's brand, model, battery type and capacity, screen size, and resolution.

• Display the phone information, battery information, and screen Information

Code :

```
class Battery {
private String type;
private int capacity;
public Battery(String type, int capacity) {
this.type = type;
this.capacity = capacity;
}
```

```
@Override
public String toString() {
return "Battery: Type: " + type + ", Capacity: " + capacity + "mAh";
}
}
class Screen {
private double size;
private String resolution;
public Screen(double size, String resolution) {
this.size = size;
this.resolution = resolution;
}
@Override
public String toString() {
return "Screen: Size: " + size + " inches, Resolution: " + resolution;
}
}
class Phone {
private String brand;
private String model;
private Battery battery;
private Screen screen;
public Phone(String brand, String model, Battery battery, Screen screen) {
this.brand = brand;
this.model = model;
this.battery = battery;
this.screen = screen;
}
```

```
public void displayPhoneInfo() {
System.out.println("Phone: Brand: " + brand + ", Model: " + model);
}
public void displayBatteryInfo() {
System.out.println(battery);
}
public void displayScreenInfo() {
System.out.println(screen);
}
}
public class jMainMobile {
public static void main(String[] args) {
Phone phone = new Phone(
"Samsung",
"F 12",
new Battery("Lithium-ion", 6000),
new Screen(6.8, "3088x1440")
);
phone.displayPhoneInfo();
phone.displayBatteryInfo();
phone.displayScreenInfo();
}
}
```

Output :

```
Phone: Brand: Samsung, Model: F 12
Battery: Type: Lithium-ion, Capacity: 6000mAh
Screen: Size: 6.8 inches, Resolution: 3088x1440
```