**SUPER KEYWORD**

**Exercise:**

class EventRegistration {

String name;

String nameOfEvent;

double registrationFee;

public void registerEvent() {

System.out.println("Please choose a valid event");

}

}

class SingleEventRegistration extends EventRegistration {

int participantNo;

public void registerEvent() {

switch (nameOfEvent) {

case "ShakeALeg":

registrationFee = 100;

break;

case "Sing&Win":

registrationFee = 150;

break;

case "PlayAway":

registrationFee = 130;

break;

default:

System.out.println("Please choose a valid event");

return;

}

System.out.println("Thank You " + name + " for your participation. Your registration fee is: " + registrationFee);

System.out.println("You are participant no: " + participantNo);

}

}

class TeamEventRegistration extends EventRegistration {

int teamNo;

int noOfParticipants;

public void registerEvent() {

switch (nameOfEvent) {

case "ShakeALeg":

registrationFee = 50;

break;

case "Sing&Win":

registrationFee = 60;

break;

case "Actathon":

registrationFee = 80;

break;

case "PlayAway":

registrationFee = 100;

break;

default:

System.out.println("Please choose a valid event");

return;

}

registrationFee \*= noOfParticipants;

System.out.println("Thank You " + name + " for your participation. Your registration fee is: " + registrationFee);

System.out.println("You are participant no: " + teamNo);

}

}

public class ShowYourTalentRegistration {

public static void main(String[] args) {

SingleEventRegistration participant1 = new SingleEventRegistration();

participant1.name = "Jenny";

participant1.nameOfEvent = "Sing&Win";

participant1.participantNo = 1;

participant1.registerEvent();

TeamEventRegistration team1 = new TeamEventRegistration();

team1.name = "Aura";

team1.nameOfEvent = "ShakeALeg";

team1.teamNo = 1;

team1.noOfParticipants = 5;

team1.registerEvent();

SingleEventRegistration participant2 = new SingleEventRegistration();

participant2.name = "Hudson";

participant2.nameOfEvent = "PlayAway";

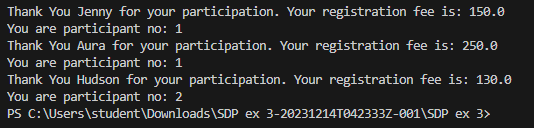
participant2.participantNo = 2;

participant2.registerEvent();

}

}

Output:



**STATIC MODIFIER**

**Exercise 1:**

**class Loan {**

**private static int loanCounter = 0;**

**private double amount;**

**public Loan() {**

**loanCounter++;**

**}**

**public Loan(double amount) {**

**this.amount = amount;**

**loanCounter++;**

**}**

**public int getLoanCounter() {**

**return loanCounter;**

**}**

**}**

**public class LoanTester {**

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

**Loan loan1 = new Loan();**

**Loan loan2 = new Loan(1000);**

**Loan loan3 = new Loan(2000);**

**Loan loan4 = new Loan(); // Creating another loan object using default constructor**

**System.out.println("Loan Counter for loan1: " + loan1.getLoanCounter());**

**System.out.println("Loan Counter for loan2: " + loan2.getLoanCounter());**

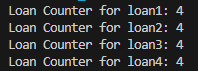
**System.out.println("Loan Counter for loan3: " + loan3.getLoanCounter());**

**System.out.println("Loan Counter for loan4: " + loan4.getLoanCounter());**

**}**

**}**

Output:



**Exercise 2:**

**class Employee {**

**private int point;**

**public int getPoint() {**

**return point;**

**}**

**public void setPoint(int point) {**

**this.point = point;**

**}**

**}**

**class PerformanceRating {**

**private static final int Outstanding = 5;**

**private static final int Good = 4;**

**private static final int Average = 3;**

**private static final int Poor = 2;**

**public static int calculatePerformance(Employee employee) {**

**int point = employee.getPoint();**

**if (point >= 80 && point <= 100) {**

**return Outstanding;**

**} else if (point >= 60 && point <= 79) {**

**return Good;**

**} else if (point >= 50 && point <= 59) {**

**return Average;**

**} else if (point >= 1 && point <= 49) {**

**return Poor;**

**} else {**

**return -1; // Invalid point**

**}**

**}**

**}**

**public class PerformanceCalculator {**

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

**Employee employee1 = new Employee();**

**employee1.setPoint(90);**

**Employee employee2 = new Employee();**

**employee2.setPoint(70);**

**Employee employee3 = new Employee();**

**employee3.setPoint(55);**

**int rating1 = PerformanceRating.calculatePerformance(employee1);**

**int rating2 = PerformanceRating.calculatePerformance(employee2);**

**int rating3 = PerformanceRating.calculatePerformance(employee3);**

**System.out.println("Employee 1 rating: " + getRatingDescription(rating1));**

**System.out.println("Employee 2 rating: " + getRatingDescription(rating2));**

**System.out.println("Employee 3 rating: " + getRatingDescription(rating3));**

**}**

**private static String getRatingDescription(int rating) {**

**switch (rating) {**

**case 5:**

**return "Outstanding";**

**case 4:**

**return "Good";**

**case 3:**

**return "Average";**

**case 2:**

**return "Poor";**

**default:**

**return "Invalid rating";**

**}**

**}**

**}**

Output:



**VARIABLE ARGUMENTS**

**Exercise:**

**import java.util.Arrays;**

**public class VarargsExercise1 {**

**public void displayList(int... input) {**

**System.out.println("Items in the list:");**

**for (int item : input) {**

**System.out.print(item + " ");**

**}**

**System.out.println();**

**}**

**public int maxOfList(int... input) {**

**if (input.length == 0) {**

**System.out.println("List is empty.");**

**return -1; // Indicating error or absence of a maximum value**

**}**

**int max = input[0];**

**for (int i = 1; i < input.length; i++) {**

**if (input[i] > max) {**

**max = input[i];**

**}**

**}**

**return max;**

**}**

**public void sortList(int... input) {**

**Arrays.sort(input);**

**System.out.println("Sorted list in ascending order:");**

**for (int item : input) {**

**System.out.print(item + " ");**

**}**

**System.out.println();**

**}**

**public double averageList(int... input) {**

**if (input.length == 0) {**

**System.out.println("List is empty.");**

**return 0.0; // Indicating error or absence of an average**

**}**

**int sum = 0;**

**for (int item : input) {**

**sum += item;**

**}**

**return (double) sum / input.length;**

**}**

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

**VarargsExercise1 varargsExercise1 = new VarargsExercise1();**

**varargsExercise1.displayList(10, 20, 30, 40, 50);**

**System.out.println("Maximum value: " + varargsExercise1.maxOfList(10, 20, 30, 40, 50));**

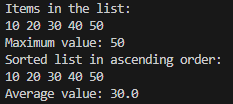
**varargsExercise1.sortList(50, 40, 30, 20, 10);**

**System.out.println("Average value: " + varargsExercise1.averageList(10, 20, 30, 40, 50));**

**}**

**}**

Output:



**ENUMERATED DATA TYPES**

**Exercise:**

**// Enum for Grade values**

**enum Grade {**

**A, B, C, D, F**

**}**

**class Student {**

**private int marks1;**

**private int marks2;**

**private int marks3;**

**public Student(String name, int marks1, int marks2, int marks3) {**

**this.marks1 = marks1;**

**this.marks2 = marks2;**

**this.marks3 = marks3;**

**}**

**public Grade calculateGrade() {**

**int totalMarks = marks1 + marks2 + marks3;**

**double averageMarks = totalMarks / 3.0;**

**if (averageMarks >= 90) {**

**return Grade.A;**

**} else if (averageMarks >= 80) {**

**return Grade.B;**

**} else if (averageMarks >= 70) {**

**return Grade.C;**

**} else if (averageMarks >= 60) {**

**return Grade.D;**

**} else {**

**return Grade.F;**

**}**

**}**

**public double calculateScholarship() {**

**Grade grade = calculateGrade();**

**switch (grade) {**

**case A:**

**return 5000.0;**

**case B:**

**return 4000.0;**

**case C:**

**return 3000.0;**

**case D:**

**return 2000.0;**

**default:**

**return 0.0;**

**}**

**}**

**}**

**public class StudentTest {**

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

**Student student = new Student("John", 95, 85, 90);**

**Grade grade = student.calculateGrade();**

**double scholarship = student.calculateScholarship();**

**System.out.println("Grade: " + grade);**

**System.out.println("Scholarship: " + scholarship + " rupees");**

**}**

**}**

Output:



**ABSTRACT CLASS**

**Exercise:**

**abstract class RRPaymentServices {**

**protected double balance;**

**public RRPaymentServices(double balance) {**

**this.balance = balance;**

**}**

**public abstract void payBill(double amount);**

**}**

**class ShoppingPayment extends RRPaymentServices {**

**private static int counter = 1000;**

**private String paymentID;**

**public ShoppingPayment(double balance) {**

**super(balance);**

**}**

**@Override**

**public void payBill(double amount) {**

**if (amount != balance) {**

**System.out.println("Error: Invalid payment amount.");**

**return;**

**}**

**paymentID = "S" + counter++;**

**System.out.println("Shopping Payment ID: " + paymentID);**

**}**

**}**

**class CreditCardPayment extends RRPaymentServices {**

**private static int counter = 5000;**

**private String paymentID;**

**private double balanceDue = 0;**

**public CreditCardPayment(double balance) {**

**super(balance);**

**}**

**@Override**

**public void payBill(double amount) {**

**if (amount > balance) {**

**paymentID = "C" + counter++;**

**} else if (amount < balance) {**

**balanceDue = balance - amount;**

**} else {**

**paymentID = "C" + counter++;**

**}**

**System.out.println("Credit Card Payment ID: " + paymentID);**

**System.out.println("Remaining Balance Due: " + balanceDue);**

**}**

**}**

**public class PaymentTester {**

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

**// Test Credit Card Payment**

**CreditCardPayment ccPayment1 = new CreditCardPayment(1000);**

**ccPayment1.payBill(800);**

**CreditCardPayment ccPayment2 = new CreditCardPayment(1500);**

**ccPayment2.payBill(2000);**

**// Test Shopping Payment**

**ShoppingPayment shoppingPayment1 = new ShoppingPayment(500);**

**shoppingPayment1.payBill(500);**

**ShoppingPayment shoppingPayment2 = new ShoppingPayment(1000);**

**shoppingPayment2.payBill(800);**

**ShoppingPayment shoppingPayment3 = new ShoppingPayment(1200);**

**shoppingPayment3.payBill(1200);**

**ShoppingPayment shoppingPayment4 = new ShoppingPayment(1500);**

**shoppingPayment4.payBill(1300);**

**ShoppingPayment shoppingPayment5 = new ShoppingPayment(2000);**

**shoppingPayment5.payBill(2200);**

**}**

**}**

Output:

