**Q1) Method Overloading: Write a class Calculator with overloaded methods add(). Implement add() methods that take: - Two integer**

**- Two double values**

**- Three integers**

**- A variable number of integers**

**Code:- package** Demo;

**class** Calculator {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **double** add(**double** a, **double** b) {

**return** a + b;

}

**public** **int** add(**int** a, **int** b, **int** c) {

**return** a + b + c;

}

**public** **int** add(**int**... numbers) {

**int** sum = 0;

**for** (**int** num : numbers) {

sum += num;

}

**return** sum;

}

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

Calculator calculator = **new** Calculator();

**int** sum1 = calculator.add(5, 3);

**double** sum2 = calculator.add(5.5, 3.2);

**int** sum3 = calculator.add(1, 2, 3);

**int** sum4 = calculator.add(1, 3, 5, 7);

System.***out***.println("Sum of 5 and 3 (int): " + sum1);

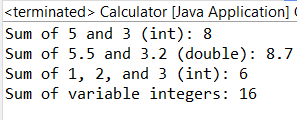
System.***out***.println("Sum of 5.5 and 3.2 (double): " + sum2);

System.***out***.println("Sum of 1, 2, and 3 (int): " + sum3);

System.***out***.println("Sum of variable integers: " + sum4);

}

}

**O/p:- **

**Q2) Super Keyword: Create a class Person with a constructor that accepts and sets name and age.**

**- Create a subclass Student that adds a grade property and initializes name and age using the super keyword in its constructor.**

**- Demonstrate the creation of Student objects and the usage of super to call the parent class constructor.**

**Code:- package** demo;

**class** Person {

**private** String name;

**private** **int** age;

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **int** getAge() {

**return** age;

}

}

**class** Student **extends** Person {

**private** **double** grade;

**public** Student(String name, **int** age, **double** grade) {

// Call superclass constructor using super keyword

**super**(name, age);

**this**.grade = grade;

}

**public** **double** getGrade() {

**return** grade;

}

}

**class** Main {

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

Student student1 = **new** Student("Peter Parker", 20, 3.8);

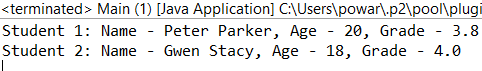
Student student2 = **new** Student("Gwen Stacy", 18, 4.0);

System.***out***.println("Student 1: Name - " + student1.getName() + ", Age - " + student1.getAge() + ", Grade - " + student1.getGrade());

System.***out***.println("Student 2: Name - " + student2.getName() + ", Age - " + student2.getAge() + ", Grade - " + student2.getGrade());

}

}

**O/p:- **

**Q3) Super Keyword: Create a base class Shape with a method draw() that prints "Drawing Shape"**

**- Create a subclass Circle that overrides draw() to print "Drawing Circle".**

**- Inside the draw() method of Circle, call the draw() method of the Shape class using super.draw().**

**- Write a main method to demonstrate calling draw() on a Circle object.**

**Code:- package** demo;

**public** **class** Shape {

**public** **void** draw() {

System.***out***.println("Drawing Shape");

}

}

**class** Circle **extends** Shape {

@Override

**public** **void** draw() {

// Call superclass draw() method using super.draw()

**super**.draw();

System.***out***.println("Drawing Circle");

}

}

**class** Main {

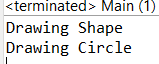
**public** **static** **void** main(String[] args) {

Circle circle = **new** Circle();

circle.draw();

}

}

**O/p:- **

**Q4) Create a base class BankAccount with a method deposit(amount) and a constructor that sets the initial balance.**

**- Create a subclass SavingsAccount that overrides deposit(amount) to add interest before depositing. Use the super keyword to call the deposit method of the base class.**

**- Write a main method to demonstrate creating a SavingsAccount and depositing an amount to see the effect of interest.**

**Code:- package** demo;

**class** BankAccount {

**protected** **double** balance;

**public** BankAccount(**double** initialBalance) {

balance = initialBalance;

}

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

balance += amount;

}

}

**class** SavingsAccount **extends** BankAccount {

**private** **double** interestRate;

**public** SavingsAccount(**double** initialBalance, **double** interestRate) {

**super**(initialBalance); // Call superclass constructor with initialBalance

**this**.interestRate = interestRate;

}

@Override

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

**double** interest = amount \* interestRate;

balance += interest; // Add interest before base class deposit

**super**.deposit(amount); // Call base class deposit method

}

}

**class** Main {

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

SavingsAccount savingsAccount = **new** SavingsAccount(1000, 0.05); // 5% interest rate

savingsAccount.deposit(500);

System.***out***.println("Balance after deposit with interest: ₹" + savingsAccount.balance);

}

}

**O/p:- **

**Q5) Define a class Employee with properties name and salary and a method displayDetails().**

**- Create a subclass Manager that adds a property department and overrides displayDetails() to include department details. Use the super keyword to call the displayDetails() method of Employee within Manager.**

**- In the main method, create objects of Employee and Manager and call displayDetails() to show the details.**

**Code:- package** demo;

**class** Employee {

**private** String name;

**private** **double** salary;

**public** Employee(String name, **double** salary) {

**this**.name = name;

**this**.salary = salary;

}

**public** **void** displayDetails() {

System.***out***.println("Name: " + name);

System.***out***.println("Salary: $" + salary);

}

}

**class** Manager **extends** Employee {

**private** String department;

**public** Manager(String name, **double** salary, String department) {

**super**(name, salary); // Call superclass constructor with name and salary

**this**.department = department;

}

@Override

**public** **void** displayDetails() {

**super**.displayDetails(); // Call superclass displayDetails() first

System.***out***.println("Department: " + department);

}

}

**class** Main {

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

Employee employee = **new** Employee("Peter Parker", 50000.0);

Manager manager = **new** Manager("Gwen stacy", 75000.0, "Engineering");

System.***out***.println("Employee Details:");

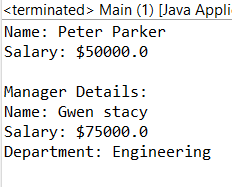
employee.displayDetails();

System.***out***.println("\nManager Details:");

manager.displayDetails();

}

}

**O/p:- **

**Q6) Write the same programme for the class ImmutableExample, to achieve object value ‘Hi’.**

**Code:- package** demo;

**public** **final** **class** ImmutableExample {

**private** **final** String value;

**public** ImmutableExample() {

**this**.value = "Hi";

}

**public** String getValue() {

// Return a copy of the value to prevent modification

**return** **new** String(value);

}

**public** **class** TestImmutableExample {

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

ImmutableExample example = **new** ImmutableExample();

String value = example.getValue();

System.***out***.println("Value: " + value);

}

}

}

**O/p:- **

**Q7) Write the same programme for the class MutableExample, to output the object values ‘hello 2’ and ‘hello3’.**

**Code:- package** demo;

**public** **class** MutableExample {

**private** String message;

**private** **int** number;

**public** MutableExample(String message, **int** number) {

**this**.message = message;

**this**.number = number;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

**public** **void** setNumber(**int** number) {

**this**.number = number;

}

**public** String getMessage() {

**return** message;

}

**public** **int** getNumber() {

**return** number;

}

**public** String toString() {

**return** "MutableExample [message=" + message + ", number=" + number + "]";

}

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

MutableExample example1 = **new** MutableExample("hello", 2);

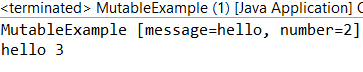
MutableExample example2 = **new** MutableExample("hello", 3);

System.***out***.println(example1); // Output: MutableExample [message=hello, number=2]

System.***out***.println(example2.getMessage() + " " + example2.getNumber()); // Output: hello 3

}

}

**O/p:- **

**Q8) Write a java class to implement any 10 string methods:**

**● replace ● contains ● replaceAll ● indexOf ● substring ● Equals ● lastIndexOf ● startsWith ● endsWith ● EqualsIgnoreCase ● toLowerCase ● toUpperCase ● isEmpty ● Length ● split**

**Code:- package** demo;

**public** **class** StringOperations {

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

String str = "This is a string example.";

String replaced = str.replace("string", "sentence");

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

**boolean** containsString = str.contains("string");

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

String replacedAll = str.replaceAll("\\s", "-");

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

**int** firstIndex = str.indexOf("is");

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

String subString = str.substring(8, 14);

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

**boolean** isEqual = str.equals("This is a string example.");

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

**int** lastIndex = str.lastIndexOf(" ");

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

**boolean** startsWithThis = str.startsWith("This");

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

**boolean** endsWithExample = str.endsWith("example.");

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

**boolean** equalsIgnoreCase = str.equalsIgnoreCase("THIS IS A STRING EXAMPLE.");

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

String lowercase = str.toLowerCase();

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

String uppercase = str.toUpperCase();

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

**boolean** isEmpty = str.isEmpty();

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

**int** length = str.length();

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

String[] words = str.split(" ");

**for** (String word : words) {

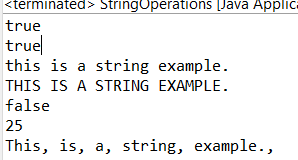
System.***out***.print(word + ", ");

}

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

}

}

**O/p:- **