**LAB 4**

**­**

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

     - Two integers

     - Two double values

     - Three integers

     - A variable number of integers

**Programe:**

**package** Demo;

**public** **class** Calculator {

// Method to add two integers

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

**return** a + b;

}

// Method to add two double values

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

**return** a + b;

}

// Method to add three integers

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

**return** a + b + c;

}

// Method to add a variable number of integers

**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();

// Example usage

System.***out***.println("Sum of 10 and 20: " + calculator.add(10, 20));

System.***out***.println("Sum of 10.5 and 20.5: " + calculator.add(10.5, 20.5));

System.***out***.println("Sum of 10, 20, and 30: " + calculator.add(10, 20, 30));

System.***out***.println("Sum of 1, 2, 3, 4, and 5: " + calculator.add(1, 2, 3, 4, 5));

}

}

**Output:**

Sum of 10 and 20: 30

Sum of 10.5 and 20.5: 31.0

Sum of 10, 20, and 30: 60

Sum of 1, 2, 3, 4, and 5: 15

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.

**Programe:**

**package** Demo;

**class** Person {

**private** String name;

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

// Constructor for Person

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

**this**.name = name;

**this**.age = age;

}

// Getters for name and age (optional)

**public** String getName() {

**return** name;

}

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

**return** age;

}

}

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

**private** **int** grade;

// Constructor for Student

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

**super**(name, age); // Call the parent class constructor using super

**this**.grade = grade;

}

// Getter for grade (optional)

**public** **int** getGrade() {

**return** grade;

}

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

// Create a Student object

Student student1 = **new** Student("Akshay", 18, 12);

// Access properties using getters

System.***out***.println("Student Name: " + student1.getName());

System.***out***.println("Student Age: " + student1.getAge());

System.***out***.println("Student Grade: " + student1.getGrade());

}

}

**Output:**

Student Name: Akshay

Student Age: 18

Student Grade: 12

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.

**Programe:**

**package** Demo;

**class** Shape {

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

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

}

}

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

@Override

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

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

**super**.draw(); // Call the draw() method of the Shape class

}

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

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

circle.draw();

}

}

**Output:**

Drawing Circle

Drawing Shape

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.

**Programe:**

**package** Demo;

**class** BankAccount {

**private** **double** balance;

// Constructor to set initial balance

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

**this**.balance = initialBalance;

}

// Deposit method

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

balance += amount;

}

// Getter for balance (optional)

**public** **double** getBalance() {

**return** balance;

}

}

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

**private** **static** **final** **double** ***INTEREST\_RATE*** = 0.05; // 5% interest rate

// Constructor for SavingsAccount

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

**super**(initialBalance); // Call the parent class constructor

}

// Override deposit method to add interest

@Override

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

**double** interestAmount = amount \* ***INTEREST\_RATE***;

**super**.deposit(amount + interestAmount); // Call parent class's deposit method

}

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

// Create a SavingsAccount with an initial balance of ₹1000

SavingsAccount savingsAccount = **new** SavingsAccount(1000);

// Deposit ₹200

savingsAccount.deposit(200);

// Check the updated balance

System.***out***.println("Updated balance: ₹" + savingsAccount.getBalance());

}

}

**Output:**

Updated balance: ₹1305.0

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.

**Programe:**

**package** Demo;

**class** Employee {

**private** String name;

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

// Constructor for Employee

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

**this**.name = name;

**this**.salary = salary;

}

// Method to display employee details

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

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

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

}

}

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

**private** String department;

// Constructor for Manager

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

**super**(name, salary); // Call the parent class constructor

**this**.department = department;

}

// Override displayDetails to include department details

@Override

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

**super**.displayDetails(); // Call parent class's displayDetails method

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

}

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

// Create an Employee object

Employee employee = **new** Employee("Jai Kumar", 50000);

// Create a Manager object

Manager manager = **new** Manager("Amit Sharma", 80000, "Sales");

// Display details for both

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

employee.displayDetails();

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

manager.displayDetails();

}

}

**Output:**

Employee Details:

Employee Name: Jai Kumar

Salary: ₹50000.0

Manager Details:

Employee Name: Amit Sharma

Salary: ₹80000.0

Department: Sales

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

**Programe:**

**package** Demo;

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

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

// Constructor to initialize the value

**public** ImmutableExample(String value) {

**this**.value = value;

}

// Getter for the value (no setter)

**public** String getValue() {

**return** value;

}

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

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

// Display the value

System.***out***.println("Object value: " + example.getValue());

}

}

**Output:**

Object value: Hi

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

**Programe:**

**package** Demo;

**class** MutableExample {

**private** String value;

// Constructor to initialize the value

**public** MutableExample(String value) {

**this**.value = value;

}

// Setter method to modify the value

**public** **void** setValue(String newValue) {

**this**.value = newValue;

}

// Getter method to retrieve the value

**public** String getValue() {

**return** value;

}

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

// Create an instance of MutableExample

MutableExample example = **new** MutableExample("hello");

// Modify the value

example.setValue("hello 2");

// Display the updated value

System.***out***.println("Updated value: " + example.getValue());

// Modify the value again

example.setValue("hello3");

// Display the final value

System.***out***.println("Final value: " + example.getValue());

}

}

**Output:**

Updated value: hello 2

Final value: hello3

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

**Programe:**

**package** Demo;

**public** **class** StringMethodsDemo {

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

String originalString = "Hello, World!";

// replace

String replacedString = originalString.replace("World", "Universe");

System.**out**.println("Replaced string: " + replacedString);

// contains

**boolean** containsHello = originalString.contains("Hello");

System.**out**.println("Contains 'Hello': " + containsHello);

// replaceAll

String replacedAllString = originalString.replaceAll("[aeiou]", "\*");

System.**out**.println("Replaced all vowels: " + replacedAllString);

// indexOf

**int** indexOfComma = originalString.indexOf(",");

System.**out**.println("Index of comma: " + indexOfComma);

// substring

String substring = originalString.substring(7);

System.**out**.println("Substring from index 7: " + substring);

// equals

**boolean** isEqual = originalString.equals("Hello, World!");

System.**out**.println("Is equal: " + isEqual);

// lastIndexOf

**int** lastIndexOfL = originalString.lastIndexOf("l");

System.**out**.println("Last index of 'l': " + lastIndexOfL);

// startsWith

**boolean** startsWithHello = originalString.startsWith("Hello");

System.**out**.println("Starts with 'Hello': " + startsWithHello);

// endsWith

**boolean** endsWithExclamation = originalString.endsWith("!");

System.**out**.println("Ends with '!': " + endsWithExclamation);

// equalsIgnoreCase

**boolean** isEqualIgnoreCase = originalString.equalsIgnoreCase("hello, world!");

System.**out**.println("Is equal (ignore case): " + isEqualIgnoreCase);

// toLowerCase

String lowerCaseString = originalString.toLowerCase();

System.**out**.println("Lowercase string: " + lowerCaseString);

// toUpperCase

String upperCaseString = originalString.toUpperCase();

System.**out**.println("Uppercase string: " + upperCaseString);

// isEmpty

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

System.**out**.println("Is empty: " + isEmpty);

// length

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

System.**out**.println("Length: " + length);

// split

String[] splitArray = originalString.split(",");

System.**out**.println("Split array:");

**for** (String part : splitArray) {

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

}

}

}

**Output:**

Replaced string: Hello, Universe!

Contains 'Hello': true

Replaced all vowels: H\*ll\*, W\*rld!

Index of comma: 5

Substring from index 7: World!

Is equal: true

Last index of 'l': 10

Starts with 'Hello': true

Ends with '!': true

Is equal (ignore case): true

Lowercase string: hello, world!

Uppercase string: HELLO, WORLD!

Is empty: false

Length: 13

Split array:

Hello

World!