**Lab4**

**1.     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**

**Program:**

**package** lab4;

**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();

// Testing the overloaded methods

System.***out***.println("Addition of two integers: " + calculator.add(5, 3));

System.***out***.println("Addition of two double values: " + calculator.add(2.5, 3.7));

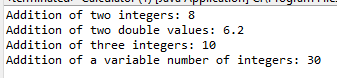
System.***out***.println("Addition of three integers: " + calculator.add(5, 3, 2));

System.***out***.println("Addition of a variable number of integers: " + calculator.add(2, 4, 6, 8, 10));

}

}

**Output:**

****

**2. 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.**

**Program:**

**package** lab4;

//Parent class Person

**class** Person {

String name;

**int** age;

// Constructor accepting name and age

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

**this**.name = name;

**this**.age = age;

}

}

//Subclass Student inheriting from Person

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

**int** grade;

// Constructor initializing name, age, and grade using super keyword

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

**super**(name, age); // Calling the constructor of the parent class

**this**.grade = grade;

}

}

**package** lab4;

**public** **class** Main {

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

// Creating Student objects and demonstrating the usage of super keyword

Student student1 = **new** Student("Saylee", 20, 12);

Student student2 = **new** Student("Sanket", 19, 11);

// Displaying information about students

System.***out***.println("Student 1: Name - " + student1.name + ", Age - " + student1.age + ", Grade - " + student1.grade);

System.***out***.println("Student 2: Name - " + student2.name + ", Age - " + student2.age + ", Grade - " + student2.grade);

}

}

**Output:**

****

**3. 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.**

**Program:**

**package** lab4;

//Base class Shape

**class** Shape {

// Method to draw shape

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

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

}

}

//Subclass Circle inheriting from Shape

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

// Override draw method to draw circle

@Override

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

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

**super**.draw(); // Calling draw method of Shape class using super

}

}

**package** lab4;

**public** **class** Main1{

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

// Creating a Circle object and calling draw method

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

circle.draw();

}

}

**Output:**

****

**4. 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.**

**Program:**

**package** lab4;

//Base class BankAccount

**class** BankAccount {

**double** balance;

// Constructor to set initial balance

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

**this**.balance = initialBalance;

}

// Method to deposit amount

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

balance += amount;

System.***out***.println("Deposited: $" + amount);

}

}

//Subclass SavingsAccount inheriting from BankAccount

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

**double** interestRate;

// Constructor to set initial balance and interest rate

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

**super**(initialBalance); // Calling the constructor of the base class

**this**.interestRate = interestRate;

}

// Override deposit method to add interest before depositing

@Override

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

**double** interest = balance \* interestRate / 100;

balance += interest;

**super**.deposit(interest); // Calling deposit method of the base class to add interest

**super**.deposit(amount); // Calling deposit method of the base class to deposit the amount

}

}

**package** lab4;

**public** **class** Main2{

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

// Creating a SavingsAccount with initial balance of $1000 and interest rate of 5%

SavingsAccount savingsAccount = **new** SavingsAccount(1000, 5);

// Depositing $500 to see the effect of interest

savingsAccount.deposit(500);

// Printing the final balance

System.***out***.println("Final Balance: $" + savingsAccount.balance);

}

}

**Output:**

**A black and white text

Description automatically generated with medium confidence**

**5. 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.**

**Program:**

**package** lab4;

//Parent class Employee

**class** Employee {

String name;

**double** salary;

// Constructor

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

**this**.name = name;

**this**.salary = salary;

}

// Method to display employee details

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

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

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

}

}

//Subclass Manager inheriting from Employee

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

String department;

// Constructor

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

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

**this**.department = department;

}

// Override displayDetails method to include department details

@Override

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

**super**.displayDetails(); // Calling displayDetails method of Employee using super

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

}

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

// Creating an Employee object

Employee employee = **new** Employee("Saylee Pawar", 50000);

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

employee.displayDetails();

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

// Creating a Manager object

Manager manager = **new** Manager("Sanket Pawar", 80000, "Sales");

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

manager.displayDetails();

}

}

**Output:**

**A screenshot of a computer

Description automatically generated**

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

**Program:**

**package** lab4;

//Immutable class ImmutableExample

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

**private** **final** String value = "Hi"; // Immutable property

// Method to get the value

**public** String getValue() {

**return** value;

}

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

// Creating an instance of ImmutableExample

ImmutableExample immutableExample = **new** ImmutableExample();

// Displaying the value

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

}

}

**Output:**

****

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

**Program:**

**package** lab4;

//Mutable class MutableExample

**class** MutableExample {

**private** String value; // Mutable property

// Constructor

**public** MutableExample(String value) {

**this**.value = value;

}

// Method to set the value

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

**this**.value = value;

}

// Method to get the value

**public** String getValue() {

**return** value;

}

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

// Creating an instance of MutableExample with initial value "hello 2"

MutableExample mutableExample1 = **new** MutableExample("hello 2");

// Displaying the value

System.***out***.println("Value 1: " + mutableExample1.getValue());

// Modifying the value to "hello 3"

mutableExample1.setValue("hello 3");

// Displaying the modified value

System.***out***.println("Value 2: " + mutableExample1.getValue());

}

}

**Output:**

**A screen shot of a computer

Description automatically generated**

**8.     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**

**Program:**

**package** lab4;

**public** **class** StringMethodsExample {

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

String str = "Hello, World!";

// Replace method

String replacedStr = str.replace('o', 'x');

System.***out***.println("Replace method: " + replacedStr);

// Contains method

**boolean** contains = str.contains("World");

System.***out***.println("Contains method: " + contains);

// ReplaceAll method

String replacedAllStr = str.replaceAll("o", "x");

System.***out***.println("ReplaceAll method: " + replacedAllStr);

// IndexOf method

**int** indexOf = str.indexOf('o');

System.***out***.println("IndexOf method: " + indexOf);

// Substring method

String substring = str.substring(7);

System.***out***.println("Substring method: " + substring);

// Equals method

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

System.***out***.println("Equals method: " + equals);

// LastIndexOf method

**int** lastIndexOf = str.lastIndexOf('o');

System.***out***.println("LastIndexOf method: " + lastIndexOf);

// StartsWith method

**boolean** startsWith = str.startsWith("Hello");

System.***out***.println("StartsWith method: " + startsWith);

// EndsWith method

**boolean** endsWith = str.endsWith("World!");

System.***out***.println("EndsWith method: " + endsWith);

// EqualsIgnoreCase method

**boolean** equalsIgnoreCase = str.equalsIgnoreCase("HELLO, WORLD!");

System.***out***.println("EqualsIgnoreCase method: " + equalsIgnoreCase);

// ToLowerCase method

String lowerCaseStr = str.toLowerCase();

System.***out***.println("ToLowerCase method: " + lowerCaseStr);

// ToUpperCase method

String upperCaseStr = str.toUpperCase();

System.***out***.println("ToUpperCase method: " + upperCaseStr);

// IsEmpty method

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

System.***out***.println("IsEmpty method: " + isEmpty);

// Length method

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

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

// Split method

String[] splitStr = str.split(",");

System.***out***.print("Split method: ");

**for** (String s : splitStr) {

System.***out***.print(s.trim() + " ");

}

}

}

**Output:**

**A screenshot of a computer program

Description automatically generated**