import java.util.HashMap;

import java.util.Scanner;

class GameCharacter {

private String name;

public GameCharacter(String name) {

this.name = name;

}

public void walk() {

System.out.println(name + " is walking...");

}

public void talk() {

System.out.println(name + " is talking...");

}

}

// Base class for student

class Student {

private String PRN;

private String name;

private int age;

// Constructor

public Student(String PRN, String name, int age) {

this.PRN = PRN;

this.name = name;

this.age = age;

}

// Getter methods

public String getPRN() {

return PRN;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

// Display method

public void displayDetails() {

System.out.println("PRN: " + PRN);

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

System.out.println("Age: " + age);

}

}

// UGStudent class (inherits from Student)

// UGStudent class (inherits from Student)

class UGStudent extends Student {

private int semester;

private double fees;

// Constructor

public UGStudent(String PRN, String name, int age, int semester, double fees) {

super(PRN, name, age); // Call superclass constructor

this.semester = semester;

this.fees = fees;

}

// Getter methods

public int getSemester() {

return semester;

}

public double getFees() {

return fees;

}

// Display method (overrides superclass method)

@Override

public void displayDetails() {

super.displayDetails(); // Call superclass method

System.out.println("Semester: " + semester);

System.out.println("Fees: " + fees);

}

}

// PGStudent class (inherits from Student)

// PGStudent class (inherits from Student)

class PGStudent extends Student {

private int semester;

private double stipend;

// Constructor

public PGStudent(String PRN, String name, int age, int semester, double stipend) {

super(PRN, name, age); // Call superclass constructor

this.semester = semester;

this.stipend = stipend;

}

// Getter methods

public int getSemester() {

return semester;

}

public double getStipend() {

return stipend;

}

// Display method (overrides superclass method)

@Override

public void displayDetails() {

super.displayDetails(); // Call superclass method

System.out.println("Semester: " + semester);

System.out.println("Stipend: " + stipend);

}

}

public class ass2{

public static void main(String[] args) {

System.out.println(" // Question 1: Method Overloading vs Method Overriding");

// Question 1: Method Overloading vs Method Overriding

explainMethodDifference();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println(

"// Question 2: Implement all string functions"

);

// Question 2: Implement all string functions

implementStringFunctions();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println("// Question 3: Implement all StringBuffer functions");

// Question 3: Implement all StringBuffer functions

implementStringBufferFunctions();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println("// Question 4: String declaration using string literal and new keyword");

// Question 4: String declaration using string literal and new keyword

demonstrateStringDeclaration();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println(" // Question 5: Implement a class hierarchy for shapes");

// Question 5: Implement a class hierarchy for shapes

printShapeDetails();

// Question 6: Create game characters using inheritance

// createGameCharacters();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println(

" // Question 8: Demonstrate StringBuilder"

);

System.out.println(" // Question 8: Demonstrate StringBuilder");

// Question 8: Demonstrate StringBuilder

demonstrateStringBuilder();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println("// Question 9: Demonstrate Method Overriding");

// Question 9: Demonstrate Method Overriding

demonstrateMethodOverriding();

// Question 10: Implement class hierarchy for students

// calculateSemesterWiseAverageAge();

System.out.println("-----------------------------");

System.out.println(" ");

System.out.println("// Question 11: Implement hybrid inheritance using access specifiers");

// Question 11: Implement hybrid inheritance using access specifiers

implementHybridInheritance();

// Question 12: Implement classes and interface for teacher and department

// displayCollegeDetails();

}

// Function for Question 1

public static void explainMethodDifference() {

System.out.println("Explanation of Method Overloading and Method Overriding...");

// Method Overloading

System.out.println("Method Overloading:");

System.out.println("- Involves defining multiple methods within the same class");

System.out.println("- Methods have the same name but differ in their parameters");

System.out.println("- Parameters can differ in number, order, or data type");

System.out.println("- Return type can be the same or different");

System.out.println("- Occurs within a single class (no inheritance required)");

// Method Overriding

System.out.println("\nMethod Overriding:");

System.out.println("- Occurs in inheritance hierarchies");

System.out.println("- Subclass redefines a method inherited from its parent class");

System.out.println("- Subclass provides its own implementation specific to its needs");

System.out.println("- Maintains the same method name and signature as the parent class method");

System.out.println("- Enables polymorphism, allowing objects of different subclasses to be treated uniformly");

}

// Function for Question 2

public static void implementStringFunctions() {

System.out.println("Implementation of all string functions...");

// Example string

String str = "Hello, World!";

// Basic Operations

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

System.out.println("Character at index 3: " + str.charAt(3));

System.out.println("Substring from index 7 to 12: " + str.substring(7, 12));

System.out.println("Concatenation with another string: " + str.concat(" Goodbye!"));

// Case Conversion

System.out.println("To lowercase: " + str.toLowerCase());

System.out.println("To uppercase: " + str.toUpperCase());

// Searching and Replacing

System.out.println("Index of 'o': " + str.indexOf('o'));

System.out.println("Last index of 'o': " + str.lastIndexOf('o'));

System.out.println("Replacing 'o' with 'x': " + str.replace('o', 'x'));

// Splitting and Joining

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

System.out.println("Split by comma: ");

for (String part : parts) {

System.out.println(part);

}

System.out.println("Joining parts with '-': " + String.join("-", parts));

// Checking and Trimming

System.out.println("Starts with 'Hello': " + str.startsWith("Hello"));

System.out.println("Ends with 'World!': " + str.endsWith("World!"));

System.out.println("Trimming whitespace: " + " Trimmed ".trim());

// Additional Methods

System.out.println("Equals to 'Hello, World!': " + str.equals("Hello, World!"));

System.out.println("Equals to 'hello, world!': " + str.equalsIgnoreCase("hello, world!"));

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

}

// Function for Question 3

public static void implementStringBufferFunctions() {

System.out.println("Implementation of all StringBuffer functions...");

// Create a StringBuffer object

StringBuffer sb = new StringBuffer("Hello");

// Appending and Inserting

sb.append(" World!");

sb.insert(5, " Beautiful");

System.out.println("Appended and Inserted: " + sb);

// Deleting and Replacing

sb.delete(6, 15);

sb.replace(6, 13, "Amazing");

System.out.println("Deleted and Replaced: " + sb);

// Capacity and Length

System.out.println("Capacity: " + sb.capacity());

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

// Reversing and Other

sb.reverse();

System.out.println("Reversed: " + sb);

System.out.println("String representation: " + sb.toString());

}

// Function for Question 4

public static void demonstrateStringDeclaration() {

System.out.println("Demonstration of String declaration using string literal and new keyword...");

// Declaration using string literal

String strLiteral = "Hello, using string literal!";

System.out.println("String Literal: " + strLiteral);

// Declaration using new keyword

String strNew = new String("Hello, using new keyword!");

System.out.println("String with new keyword: " + strNew);

}

// Function for Question 5

public static void printShapeDetails() {

System.out.println("Creating shape classes and printing their details...");

// Base class for shapes

class Shape {

public void printShape() {

System.out.println("This is a Shape");

}

}

// Subclasses for specific shapes

class Rectangle extends Shape {

public void printShape() {

System.out.println("This is a rectangular shape");

}

}

class Circle extends Shape {

public void printShape() {

System.out.println("This is a circular shape");

}

}

class Square extends Rectangle {

public void printSquare() {

System.out.println("Square is a rectangle");

}

}

// Creating instances of each shape

Shape shape = new Shape();

Rectangle rectangle = new Rectangle();

Circle circle = new Circle();

Square square = new Square();

// Printing details of each shape

shape.printShape();

rectangle.printShape(); // Calls overridden method from Rectangle

circle.printShape(); // Calls overridden method from Shape

square.printShape(); // Calls inherited method from Rectangle and overridden method from Shape

square.printSquare(); // Calls method specific to Square

}

// Function for Question 8

public static void demonstrateStringBuilder() {

System.out.println("Demonstrating StringBuilder usage...");

// Create a StringBuilder object

StringBuilder sb = new StringBuilder();

// Append strings

sb.append("Hello");

sb.append(" World!");

System.out.println("Current string: " + sb);

// Insert a string at specific index

sb.insert(5, " Beautiful");

System.out.println("String after insertion: " + sb);

// Replace a part of the string

sb.replace(7, 16, "Amazing");

System.out.println("String after replacement: " + sb);

// Delete a part of the string

sb.delete(7, 12);

System.out.println("String after deletion: " + sb);

// Reverse the string

sb.reverse();

System.out.println("Reversed string: " + sb);

// Get the length of the string

int length = sb.length();

System.out.println("Length of the string: " + length);

// Get a character at specific index

char character = sb.charAt(3);

System.out.println("Character at index 3: " + character);

}

public static void demonstrateMethodOverriding() {

System.out.println("Demonstrating Method Overriding...");

// Base class

class Result {

public void result() {

System.out.println("General Result");

}

}

// Subclasses overriding method

class UGResult extends Result {

@Override

public void result() {

System.out.println("Undergraduate Result");

}

}

class PGResult extends Result {

@Override

public void result() {

System.out.println("Postgraduate Result");

}

}

// Creating instances and calling overridden method

Result ug = new UGResult();

Result pg = new PGResult();

ug.result(); // Output: Undergraduate Result

pg.result(); // Output: Postgraduate Result

}

public static void implementHybridInheritance() {

System.out.println("Implementing hybrid inheritance using all access specifiers...");

// Base class

class Base {

private int privateField;

protected int protectedField;

public Base() {

privateField = 10;

protectedField = 20;

}

public void baseMethod() {

System.out.println("Base method: accessing privateField (not recommended) = " + privateField);

System.out.println("Base method: accessing protectedField = " + protectedField);

}

}

// Intermediate classes

class A extends Base {

public int publicField;

public A() {

publicField = 30;

}

public void methodA() {

System.out.println("Method A: accessing publicField (inherited) = " + publicField);

// Cannot access privateField from Base (not inherited)

// System.out.println("Method A: accessing privateField (not inherited) = " + privateField);

// Can access protectedField from Base (inherited)

System.out.println("Method A: accessing protectedField (inherited) = " + protectedField);

}

}

class B extends Base {

public int publicField;

public B() {

publicField = 40;

}

public void methodB() {

System.out.println("Method B: accessing publicField (inherited) = " + publicField);

// Cannot access privateField from Base (not inherited)

// System.out.println("Method B: accessing privateField (not inherited) = " + privateField);

// Can access protectedField from Base (inherited)

System.out.println("Method B: accessing protectedField (inherited) = " + protectedField);

}

}

// Subclass inheriting from multiple intermediate classes

class C extends A {

public int publicFieldC;

public C() {

publicFieldC = 50;

}

public void methodC() {

System.out.println("Method C: accessing publicFieldC (inherited) = " + publicFieldC);

// Cannot access privateField from Base (not inherited)

// System.out.println("Method C: accessing privateField (not inherited) = " + privateField);

// Can access protectedField from Base (inherited through A)

System.out.println("Method C: accessing protectedField (inherited) = " + protectedField);

}

}

// Creating instances of subclasses

A objA = new A();

B objB = new B();

C objC = new C();

// Calling methods of each subclass

objA.methodA();

objB.methodB();

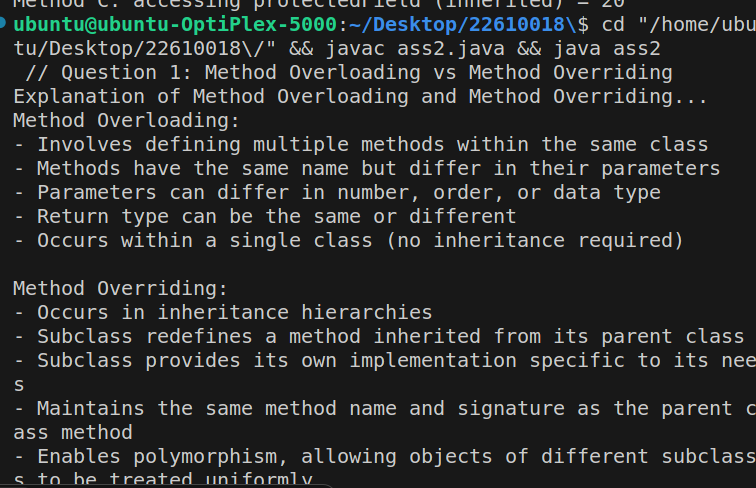
objC.methodA(); // Accessing methodA() from A

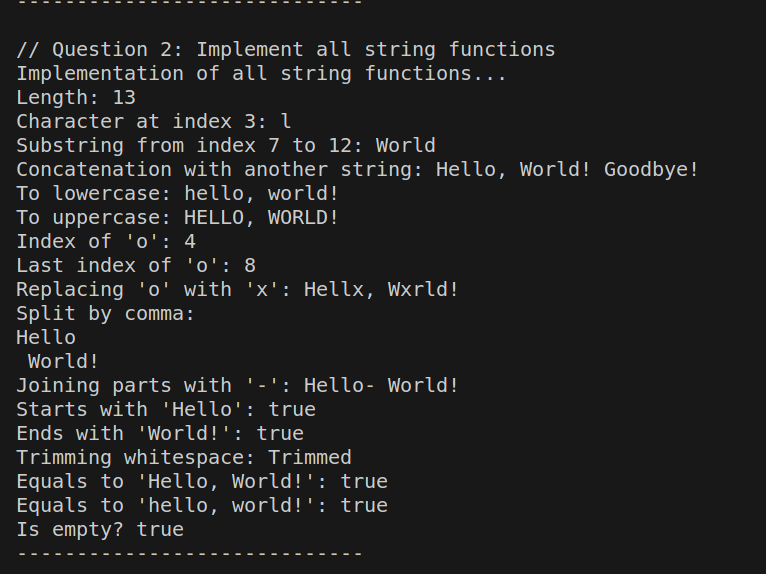
// objC.methodB(); // Accessing methodB() from B

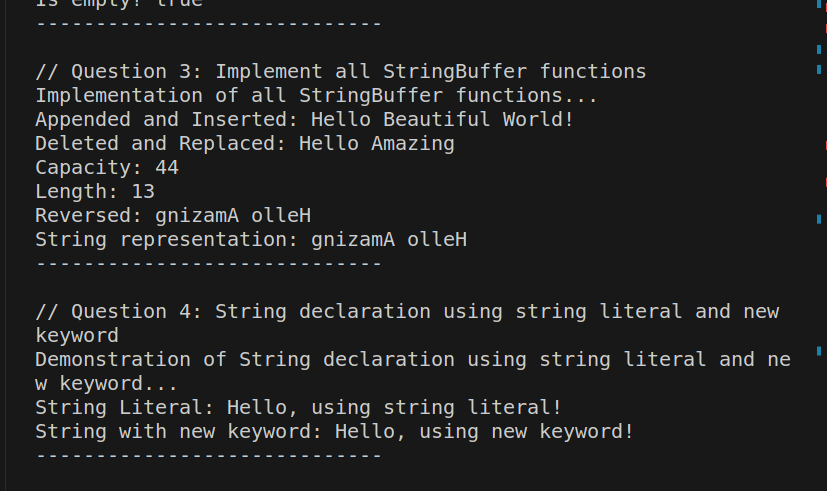
objC.methodC();

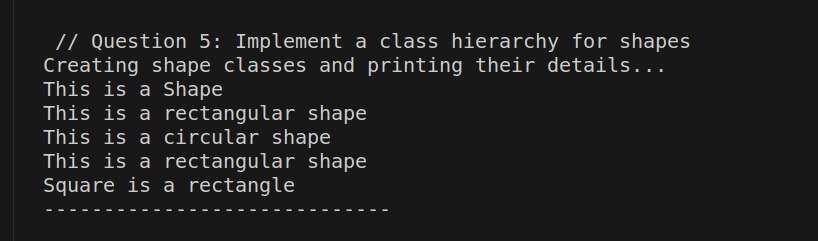
}

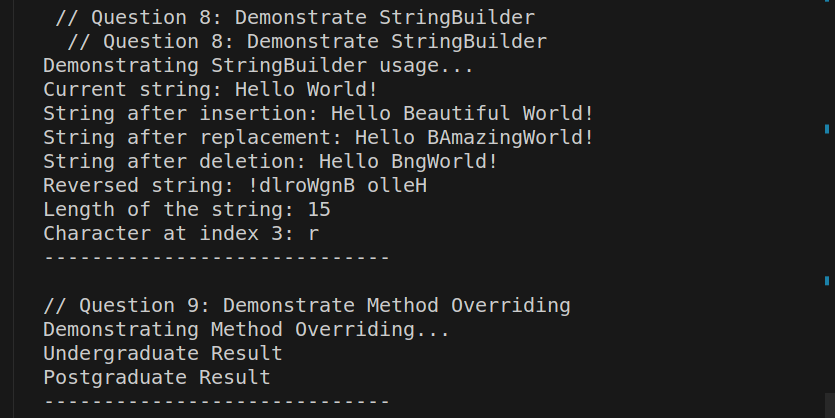
}

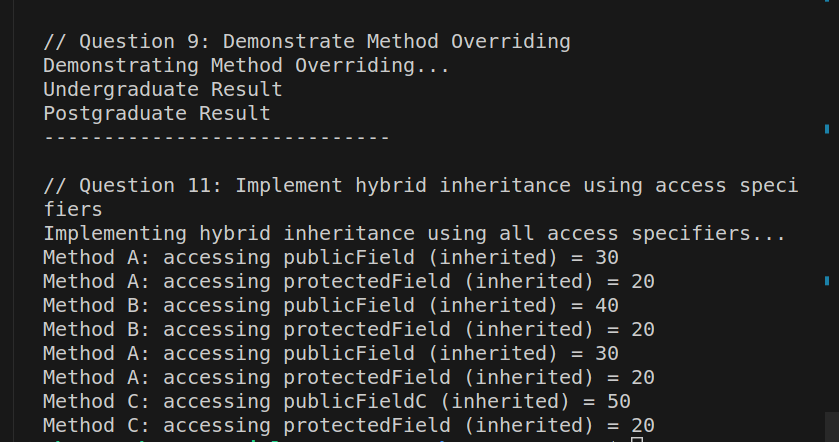












Colledge Details

// CollegeDetails.java

// Interface for College

interface College {

String getCollegeName();

}

// Teacher class

class Teacher {

private String name;

private String qualification;

// Constructor

public Teacher(String name, String qualification) {

this.name = name;

this.qualification = qualification;

}

// Getter methods

public String getName() {

return name;

}

public String getQualification() {

return qualification;

}

}

// Department class (inherits from Teacher, implements College)

class Department extends Teacher implements College {

private int deptNo;

private String deptName;

// Constructor

public Department(String name, String qualification, int deptNo, String deptName) {

super(name, qualification); // Call superclass constructor

this.deptNo = deptNo;

this.deptName = deptName;

}

// Method to get college name (from College interface)

@Override

public String getCollegeName() {

return "ABC College"; // Example college name

}

// Getter methods

public int getDeptNo() {

return deptNo;

}

public String getDeptName() {

return deptName;

}

}

public class game {

public static void displayCollegeDetails() {

// Creating an instance of Department

Department department = new Department("John Doe", "M.Sc.", 101, "Computer Science");

// Displaying college details

System.out.println("College Name: " + department.getCollegeName());

System.out.println("Department Details:");

System.out.println("\tDepartment No: " + department.getDeptNo());

System.out.println("\tDepartment Name: " + department.getDeptName());

System.out.println("Teacher Details:");

System.out.println("\tTeacher Name: " + department.getName());

System.out.println("\tTeacher Qualification: " + department.getQualification());

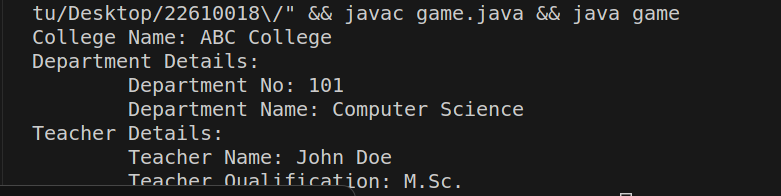
}

public static void main(String[] args) {

displayCollegeDetails();

}

}



SemesterWiseAverage

import java.util.HashMap;

import java.util.Scanner;

class Student {

protected String PRN;

protected String name;

protected int age;

public Student(String PRN, String name, int age) {

this.PRN = PRN;

this.name = name;

this.age = age;

}

public void displayDetails() {

System.out.println("PRN: " + PRN);

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

System.out.println("Age: " + age);

}

}

class UGStudent extends Student {

private int semester;

public UGStudent(String PRN, String name, int age, int semester) {

super(PRN, name, age);

this.semester = semester;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Semester: " + semester);

}

}

class PGStudent extends Student {

private int semester;

public PGStudent(String PRN, String name, int age, int semester) {

super(PRN, name, age);

this.semester = semester;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Semester: " + semester);

}

}

public class semavg {

public static void main(String[] args) {

calculateSemesterWiseAverageAge();

}

public static void calculateSemesterWiseAverageAge() {

System.out.println("Calculating semester-wise average age for UG and PG students...");

Scanner scanner = new Scanner(System.in);

HashMap<Integer, Double> ugSemesterWiseAvgAge = new HashMap<>();

HashMap<Integer, Double> pgSemesterWiseAvgAge = new HashMap<>();

// Enter data for 5 students

for (int i = 0; i < 5; i++) {

System.out.println("Enter student details (PRN, Name, Age, Semester):");

String PRN = scanner.next();

String name = scanner.next();

int age = scanner.nextInt();

int semester = scanner.nextInt();

if (i % 2 == 0) { // Even index -> UG student

UGStudent ugStudent = new UGStudent(PRN, name, age, semester);

ugStudent.displayDetails();

// Update UG semester-wise average age

ugSemesterWiseAvgAge.putIfAbsent(semester, 0.0);

ugSemesterWiseAvgAge.put(semester, ugSemesterWiseAvgAge.get(semester) + age);

} else { // Odd index -> PG student

PGStudent pgStudent = new PGStudent(PRN, name, age, semester);

pgStudent.displayDetails();

// Update PG semester-wise average age

pgSemesterWiseAvgAge.putIfAbsent(semester, 0.0);

pgSemesterWiseAvgAge.put(semester, pgSemesterWiseAvgAge.get(semester) + age);

}

}

// Calculate and print UG semester-wise average age

System.out.println("\nUG Semester Wise Average Age:");

for (int semester : ugSemesterWiseAvgAge.keySet()) {

double avgAge = ugSemesterWiseAvgAge.get(semester) / 2; // Assuming 2 UG students per iteration

System.out.println("Semester " + semester + ": " + String.format("%.2f", avgAge));

}

// Calculate and print PG semester-wise average age

System.out.println("\nPG Semester Wise Average Age:");

for (int semester : pgSemesterWiseAvgAge.keySet()) {

double avgAge = pgSemesterWiseAvgAge.get(semester) / 2; // Assuming 2 PG students per iteration

System.out.println("Semester " + semester + ": " + String.format("%.2f", avgAge));

}

}

}

