

**Assignment\_01**

**March 4, 2025**

**CSC241 – Object Oriented Programming**

*Muhammad Rizwan Shafiq*

*Roll no# SP24-BCS-069*

*Section: BCS-2B*

**Q1. Rewrite the following code using object-oriented approach.**

import java.util.Scanner;

class Student {

private String name;

private int marks;

public Student() {

this.name = null;

this.marks = 0;

}

public Student(String name, int marks) {

this.name = name;

this.marks = marks;

}

public String getName() {

return name;

}

public String getGrade() {

if (marks >= 90) {

return "A";

} else if (marks >= 80) {

return "B";

} else if (marks >= 70) {

return "C";

} else if (marks >= 60) {

return "D";

} else {

return "F";

}

}

public void display() {

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

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

}

}

public class StudentGradeCalculator {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Input for Student 1

System.out.println("Enter details for Student 1:");

System.out.print("Enter Name: ");

String name1 = input.next();

System.out.print("Enter Marks: ");

int marks1 = input.nextInt();

// Input for Student 2

System.out.println("Enter details for Student 2:");

System.out.print("Enter Name: ");

String name2 = input.next();

System.out.print("Enter Marks: ");

int marks2 = input.nextInt();

// Creating student objects

Student student1 = new Student(name1, marks1);

Student student2 = new Student(name2, marks2);

// Displaying results

student1.display();

student2.display();

input.close();

}

}

**Q2. Consider a class DateSixthTry. Would it be legal to add two method definitions with the following two method headings to the class DateSixthTry?**

### **Yes, it is legal.** The two method definitions:

public void setMonth(int newMonth)

public void setMonth(String newMonth)

can be legally added to the DateSixthTry class because of **method overloading** in Java which allows multiple methods with the same name, if their **parameter lists differ** in terms of:

* + The **number** of parameters.
  + The **types** of parameters.
  + The **order** of parameters (if multiple parameters exist).
* In this case:
  + setMonth(int newMonth) → Takes an int as a parameter.
  + setMonth(String newMonth) → Takes a String as a parameter.

**Example Methods:**

// Method that sets month using an integer (1-12)

public void setMonth(int newMonth) {

if (newMonth >= 1 && newMonth <= 12) {

this.month = newMonth;

} else {

System.out.println("Invalid month number!");

}

}

// Method that sets month using a string (January, February, etc.)

public void setMonth(String newMonth) {

switch (newMonth.toLowerCase()) {

case "january": month = 1; break;

case "february": month = 2; break;

case "march": month = 3; break;

case "april": month = 4; break;

case "may": month = 5; break;

case "june": month = 6; break;

case "july": month = 7; break;

case "august": month = 8; break;

case "september": month = 9; break;

case "october": month = 10; break;

case "november": month = 11; break;

case "december": month = 12; break;

default: System.out.println("Invalid month name!");

}

**Q3. Which of the statements are legal in a program class described in the assignment?**

|  |  |  |
| --- | --- | --- |
| Statement | Legal? | Reason |
| YourClass anObject = new YourClass(42, 'A'); | Yes | Correct parameter types (int, char). |
| YourClass anotherObject = new YourClass(41.99, 'A'); | No | double cannot be passed to int parameter without explicit casting. |
| YourClass yetAnotherObject = new YourClass(); yetAnotherObject.doStuff(); | No | No **default constructor** is defined. |
| YourClass oneMoreObject; oneMoreObject.doStuff(); | No | The object is **declared** but **not initialized** before calling a method. |
| oneMoreObject.YourClass(99, 'B"); | No | **Incorrect way to call a constructor** (should use new). |