Faculty of Engineering, University of Jaffna,

Department of Computer Engineering.

EC5080: Software Construction

Lab 01

Chapter 1: Introduction of features of a selected language

Duration: 3 Hours **Lecturer:** Ms. Sujanthika M.

Objectives:

1. Understanding the key features of Java as a programming language.
2. Able to implement control constructs in Java.
3. Learn to differentiate between static and dynamic typing and explore variable scope and namespaces.
4. Examine Java's automatic memory management and garbage collection

In this lab, you will implement a simple Student Management System in Java. This system will include multiple classes demonstrating Java's features, control constructs, variable scope, and memory management.

Part 1: Java features

1. Create a class `Student` with attributes `id`, `name`, `age`, and `grade`.
2. Add two more attributes other than in Part 1-1.
3. Implement a constructor and a method `displayStudentDetails()` that prints student information.
4. Implement a new constructor on your own for the student management system
5. Demonstrate exception handling by handling an `IllegalArgumentException` if the age is negative.

Part 2: Control constructs

1. Modify the `Student` class to include a method `categorizeStudent()`.
2. Use `if-else` conditions to categorize students as "Excellent", "Good", "Average", or "Needs Improvement" based on their grade.
3. Implement a `switch-case` statement to print messages for each category.
4. Create a `StudentListProcessor` class that processes a list of students using different loops:
    a. Use a `for` loop to print all students

   b. Use a `while` loop to count and print students with grades above a certain threshold
   c. Use a `do-while` loop to print student details until a particular condition is met

Part 3: Static Vs. Dynamic typing, scope and namespace

1. Modify the `Student` class to include a static variable `totalStudents` that keeps track of the total number of students.
2. Demonstrate variable scope by using local, instance, and static variables in different methods.
   a. Create a method `updateAge(int newAge)`, where a local variable is used to temporarily store `newAge` before updating the instance variable.
   b. Implement `incrementTotalStudents()` to modify the static variable.
   c. Use an instance variable `studentCategory` initialized in the constructor and accessed within different methods to show instance scope.
3. Implement a method `namespaceConflictExample()` to show how Java resolves name conflicts using the `this` keyword.

Part 4: Automatic memory management

1. Create a new class `StudentMemoryDemo` that creates and removes `Student` objects.
2. Implement a `finalize()` method in the `Student` class to print a message when an object is garbage collected.
3. Create multiple objects and set some of them to `null` to observe garbage collection.
4. Use `System.gc()` to request garbage collection.

Submission Guidelines:

- Submit a well-structured Java source file with comments
- Provide screenshots for the outputs.
- Provide correct names for your code files and outputs
- Any plagiarized work will be given zero marks
- Late submissions are not allowed

Section	**Criteria**	**Allocated Marks**
Part 1	Correct class and attributes	5
	Constructor and method implementation	5
	Exception Handling	5
	Code structure, & comments	5
	Execution & correct output	5
Part 2	Correct categorization using if-else	5
	Switch-case implementation	5
	Proper use of loops	5
	Code structure, & comments	5
	Execution & correct output	5
Part 3	Correct use of static Vs instance variables	5
	Proper demonstration of variable scope	5
	Implementation of namespaceConflictExample()	5
	Code structure, & comments	5
	Execution & correct output	5
Part 4	Correct implementation of StudentMemoryDemo	5
	Proper of use finalize() method	5
	Demonstration of garbage collection	5
	Code structure, & comments	5
	Execution & correct output	5