University of Botswana

Department of Computer Science
Course Code: CSI142 (Object-Oriented Programming)
Date: 25 February 2025

Lab Test 1

General Instructions

1. **Time Allowed:** Each lab test lasts **1 Hour**.
2. **Rooms & Time**:
   - **12:00 PM – 1:00 PM:** Rooms 247-295/296/297/294/298
3. **Submission:** Submit your `.java` files (or a single ZIP) by the end of the lab test.
4. **Tasks & Marks:** Each paper has **4 tasks**, each worth **25 marks**, for a total of **100 marks**.
5. **Clarity of Code:** Provide clear, compilable code. Partial solutions can earn partial credit if they demonstrate correct understanding.
6. **Comments:** Place short explanations in code comments (e.g., `// This method calculates the sum...`). Keep them concise.
7. **No Over-Complexity:** Simple, straightforward solutions are preferred.
8. **Fairness:** All lab tests cover the same fundamental topics (*loops, arrays/ArrayList, methods, OOP basics, UML*) in similar depth, ensuring equitable assessment.
9. **Academic Integrity:** Follow university policies—no unauthorized collaboration or code sharing.

Good luck with your Lab Tests!

Task 1 (25 marks): While vs. Do-While

1. Create a class `MenuLoop` with `main`.
2. Implement a **do-while** loop that repeatedly shows a mini menu:

```
1) Print Greeting
2) Print Farewell
0) Exit
```

3. Read the user's choice (via `Scanner`) inside the loop:
   - Use a **switch** on the choice:
     - If `1`, print `"Hello!"`
     - If `2`, print `"Goodbye!"`
     - If `0`, exit the loop.
4. End when the user selects `0`.

Comment (briefly): One difference between a `while` loop and a `do-while` loop.

Task 2 (25 marks): Arrays and Method Overloading

1. In the same class, create a static method `sumArray(int[] arr)` that returns the sum of all elements.

2. Overload it with `sumArray(ArrayList<Integer> list)`, which similarly returns the total of the list's elements.
3. In `main`, demonstrate both by:
    - Creating a small `int[]`, calling `sumArray(...)`.
    - Creating a small `ArrayList<Integer>`, calling the overloaded `sumArray(...)`.

Comment (briefly): Why can method overloading be convenient?

Task 3 (25 marks): Command-Line Arguments

1. Still in `MenuLoop`, check if **any command-line arguments** are given.
    - If `args.length > 0`, print `"Command-line arg detected: " + args[0]`.
    - Otherwise, print `"No arguments provided."`
2. Run the program from command-line (or IDE configuration) to test passing an argument, e.g., `java MenuLoop teacher`.

Comment (briefly): Why do we check `args.length` before using `args[0]`?

Task 4 (25 marks): Basic OOP – Constructor & Display Method

1. Create a **new class** `Course` with **private** fields:

```
private String courseCode;
private int creditHours;
```

2. Write a constructor `Course(String courseCode, int creditHours)` to initialize these fields using `this`.
3. Add a method `displayCourse()` that prints something like:

```
Course: CSI142, Credits: 4
```

4. In `MenuLoop.main`, create one `Course` object and call `displayCourse()`.

Comment (briefly): How does `this` help distinguish constructor parameters from fields?