Programming 2

Tutorial 5

Activity 1 - Trace the execution sequence

Determine the precise execution order of method calls and prints for the program below:

```
public class Instructor { // Instructor.java
    void prepareLesson() {
        System.out.println("Instructor: Preparing lesson plan...");
    }

    void startLecture() {
        System.out.println("Instructor: Starting the lecture.");
    }
}

public class Classroom { // Classroom.java
    void beginClass(Instructor inst) {
        System.out.println("Classroom: Opening the classroom.");
        inst.prepareLesson();
        inst.startLecture();
        System.out.println("Classroom: Class in progress...");
    }
}

public class Main { // Main.java
    public static void main(String[] args) {
        Instructor instructor = new Instructor();
        Classroom room = new Classroom();

        System.out.println("Main: Student arrives.");
        attend(room, instructor);
        System.out.println("Main: Student leaves.");
    }

    static void attend(Classroom c, Instructor i) {
        System.out.println("Student: Entering classroom.");
        c.beginClass(i);
        System.out.println("Student: Taking notes.");
    }
}
```

Your tasks:

1. List the execution sequence step-by-step from the moment `main()` starts until the program terminates.
2. Indicate who calls whom at each step. For instance:
   "`main()` calls `attend()`"
   "`Classroom.beginClass()` calls `Instructor.prepareLesson()`"
3. Include the order of printed output as it appears on the console.

Tip: Follow the execution sequence example from the lecture slides. Be explicit about when control returns to the caller.

Activity 2 - Banking System Implementation & Debugging

Goal: Practice both object interaction and systematic debugging

Scenario: Implement and debug a simple banking system

Implementation Requirements:

- Create Account class with deposit/withdraw methods
- Create Transaction class to record operations
- Implement transfer between accounts

Debugging tasks:

1. Add logging to trace all transactions
2. Verify account balances after operations
3. Handle edge cases (insufficient funds, negative amounts)

Activity 3 - Student Management

Scenario: Debug a pre-written student management system with intentional errors. The given code is in the file `StudentManagementSystem.java` which is provided with this tutorial.

Tasks:

1. Use stack traces to locate errors
2. Apply print logging to understand flow
3. Fix each bug systematically

Document your debugging process in a MS Word file.

Submission

Submit a **zip** file containing all of your source codes and documents to this tutorial's submission box in the course website on FIT LMS.