Faeq Alrimawi
Computer Science & Information Systems
University of Limerick

Assignment 3: Test-Driven Development

Due[1]: Sunday Week 13 (08/12/2024, 11:59pm Irish time)
Mark: 40% (Individual)

1. Objective

The objective of this assignment is to evaluate your understanding and application of Test-Driven Development (TDD) approach, and also data flow testing technique covered in the module.

2. Submission Guidelines

Submission will be soft copy only on **Brightspace**.

- One submission per student.
- THREE files: a report (pdf) and TWO Java files. No Zip files.
- Submit by Sunday Week 13 (11:59pm Irish time).

3. Items to Submit

You need to submit your response in TWO ways:

1. **Report** documenting the assignment, including:
    - Task 1 (TDD implementation): a comprehensive description of the TDD process followed in implementing the system, including rationale behind the choice of test cases.
    - Task 2 (Data Flow Testing): a description of the DU-pairs or definitions identified, test cases created, as well as coverage levels.

   The report should be well-structured, clear and organised, with proper headings and subheadings. **You should name your report: <ID>_report4. Format: PDF.**

2. **Java files**: you need to submit TWO java files:
    - The JUnit Java file used in your TDD implementation. This refers to the file containing all the test cases that are used in implementing the system using TDD. should be named: **TDD_<your ID>.java (your class should have the same name)**
    - The Java file containing the implementation of the system.
      should be named: **BankAccountManagementSystem_<your ID>.java**

[1] Early submission of drafts is advised. The Brightspace assignment will be set up to accept unlimited submissions, however, each one will overwrite the previous one.

4. Note on Plagiarism and Cheating

Plagiarism and cheating are serious academic offenses that undermine the integrity of the learning process and the reputation of both individuals and institutions. It is crucial that all students understand and adhere to the principles of academic honesty throughout this assignment:

1. **Original Work:** All work submitted must be original and created by student. Copying, reusing, or paraphrasing content from external sources without proper attribution is strictly prohibited.
2. **Proper Citation:** If you refer to external sources, including textbooks, online materials, or classmates' work, ensure that you provide appropriate citations. Failure to do so can be considered plagiarism.
3. **Collaboration vs. Copying:** Collaboration between students is encouraged and expected, but there is a clear distinction between collaborative work and copying. Collaboration involves sharing ideas and discussing concepts to enhance understanding. Copying involves duplicating another person's work or ideas without adding your own input.

Consequences for plagiarism and cheating may include a failing grade for the assignment, academic probation, or more severe disciplinary actions as per university policies. Please take this note seriously and maintain the highest standards of academic integrity in your work. If you have any questions or uncertainties about what constitutes plagiarism or cheating, seek clarification from your instructor before submitting your assignment.

5. Assignment Specification: What You Need to Do

The bank upon reviewing their current Account Management System, introduced in the 2nd assignment, has identified serious issues with it. As a response, the bank has decided to abandon the old system and develop a new one from scratch using Test-Driven Development (TDD) principles. Your goal is to ensure a robust system is developed using a TDD approach.

5.1 Task 1: TDD Implementation (60%)

Your task is to develop a comprehensive set of unit tests using TDD principles for the new Bank Account Management System. Start by writing a failing test, then implement the functionality to make the test pass. Refactor your code as necessary and repeat the process.

New specification: the new Bank Account Management System specification can be found at the **end of this document, page 6**.

Important: please ensure that your report details the steps and iterations of your TDD approach throughout the system development process. Have clear rationale, comments and names for your test cases. The goal here is not the final system, but rather the process that you followed to reach it.

5.2 Task2: Data Flow Testing (40%)

In this task you are required to assess the program created in Task 1 by applying data flow testing technique using two coverage criteria.

5.2.1 DU-Pair Coverage (30%)

Your task is to test the **deposit** method by applying a **DU-pair coverage**. You are required to identify relevant data variables within it, establish definition-use (DU) pairs of these variables, and then create a test plan consisting of three test cases to cover all/some of these pairs. You are also required to determine the coverage level based on the test cases in your plan. You should clearly indicate what pairs each case covers.

Hint: the coverage criterion requested in this section is the “All DU-pairs” criterion. In other words, you need to identify all DU-pairs in the deposit method and based on the three tests you created, determine their coverage level. You do not need to implement the test cases using JUnit.

5.2.2 Definition Coverage (10%)

Your task is to test the **withdraw** method by applying a **Definition coverage**. Similar to the DU-Pair coverage, you are required to identify definitions of data variables within the withdraw method and then create three test cases based on these definitions. You are also required to determine the coverage level of your tests.

6. Assessment Criteria

The table provided outlines the assessment criteria and grading breakdown for evaluating your performance. Utilise these guidelines as a reference and consistently strive for excellence in your work.

Table 1 Assessment Criteria

Criteria	Excellent (80%-100%)	Proficient (60%-79%)	Satisfactory (40%-59%)	Limited (20%-39%)	Inadequate (0%-19%)
Task 1: TDD Implementation (60%)					
Task2: Data Flow Testing (40%)					

Mark: (<Individual Score> / 100) (*40%)

6.1 Scoring Scale

Excellent:

- Represents an exceptional level of understanding and execution of the assignment's requirements.
- Achieves a performance that consistently exceeds the highest standards set by the assignment's objectives.
- Characterised by exceptional attention to detail, a comprehensive approach, and minimal to no errors.

Proficient:

- Shows strong competence in understanding and executing the assignment's requirements.
- Consistently meets or slightly exceeds the assignment's objectives.
- Characterised by strong attention to detail and a good depth of coverage.

Satisfactory:

- Meets the basic expectations for the assignment but may lack some depth or precision.
- Characterised by satisfactory quality, with moderate proficiency in fulfilling the assignment's requirements.
- May have occasional errors or incomplete coverage of the assignment's expectations.

Limited:

- Displays some understanding of the assignment's requirements and objectives.
- Achieves work that is somewhat effective but may have shown inconsistencies or gaps in execution.
- Characterised by a decent quality of work that may fall short in terms of depth, precision, or comprehensive coverage.

Inadequate:

- Fails to meet the minimum expectations for the assignment.
- Demonstrates significant deficiencies, resulting in incomplete, poorly executed work.
- Characterised by multiple errors, omissions, and inadequate coverage.

Bank Account Management System: NEW Requirement Specification

The Bank Account Management System is a Java-based software application designed to provide basic banking functionalities, including the initialisation of accounts, depositing and withdrawing funds, checking account balances and recording transaction history.

1. Requirement 1: Account Creation

Ensure the system allows the creation of a new bank account with a specified initial balance.

1.1 Implement a successful account creation operation.

1.2 Validate that the initial balance is a positive numeric value.

1.3 Implement a check to prevent the creation of duplicate accounts.

2. Requirement 2: Deposit

Enable users to deposit and withdraw funds from their bank account.

2.1 Implement a successful deposit operation.

2.2 Validate that deposited amounts are positive numeric values.

3. Requirement 3: Withdrawal

Enable users to withdraw funds from their bank account.

3.1 Implement a successful withdrawal operation.

3.2 Validate that withdrawal amounts are positive numeric values.

4. Requirement 4: Overdraft Protection

Prevent users from overdrawing their accounts and reject withdrawal attempts exceeding the available balance.

4.1 Implement overdraft prevention to disallow negative balances.

4.2 Reject withdrawal attempts that exceed the available balance.

5. Requirement 5: Balance Inquiry

Allow users to check their account balance at any time.

5.1 Implement a successful balance inquiry operation.