BCO5008: Object-Oriented Systems 1
Assignment

Dr Tom Crick
`tcrick@cardiffmet.ac.uk`
`http://drtomcrick.com`

7 December 2015

Introduction

The assessment for BCO5008: *Object-Oriented Systems 1* consists of a programming assignment in Term 2 worth 50% and an examination in May/June for the remaining 50%.

This document specifies the programming assignment: to model, design and implement a simplified banking system in Java.

Learning Objectives

On the completion of this assignment you will be able to:

- Evaluate the basic theory and concepts of object-oriented programming and design, implement, test and analyse programs using this paradigm.
- Demonstrate beginner-level fluency in using an appropriate object-oriented programming language and development tools/environments.
- Outline the need for a professional approach to system design and be able to model and analyse real-world problems using appropriate techniques.

Task

Design and implement a banking system, consisting of an appropriate number of bank account types, along with appropriate functionality for a range of standard banking and account operations.

You will need to model the problem and provide an suitable design in the form of a list of key requirements and appropriate UML diagrams (for example, class diagrams) which will represent your classes, their attributes and relationships.

You will then implement your design in Java; this should consist of the following core implementation:

- An overarching **bank environment** that is able to store one or more bank accounts (of varying types);
- A generic **bank account** type;
- Extensions of this generic bank account type to implement a **current account** and a **savings account**;

- Uniquely identifiable accounts: an **account number** (consisting of eight digits, 0-9) and a **sort code** (consisting of six digits, 0-9, in the form `XX-XX-XX`);
- Implement functionality to:
  - **deposit** and **withdraw** money;
  - **check** balances;
  - have an **overdraft** on appropriate types of accounts;
  - have an **interest rate** on appropriate types of accounts;
  - return values for any other appropriate **account operations**.

You will need to demonstrate the following basic **account operations**:

1. Manipulating the available bank accounts within the bank, using an appropriate menu system.
2. Depositing and withdrawing money from different types of account.
3. Setting an overdraft limit on a current account.
4. Applying a default bank charge when a current account exceeds its overdraft limit.
5. Applying an interest rate on a savings account.
6. Generating a statement of account transactions for any type of account that can be saved to a file.

No GUI is required for this assignment; the program should allow user input/output from the console to enter certain values, such as the opening balance, the overdraft limit and transfer amounts; this input should be validated to ensure it is appropriate. The statement functionality should make use of appropriate Java I/O features to store transactions in an appropriate format/location on the local machine.

You should also implement any further functionality that you deem appropriate in a bank account system to support the requested operations; these extensions should be clearly indicated and explained in your written overview of each class. **Marks will be awarded for innovative and interesting extensions to the basic functionality described above.**

Your system should make **appropriate use of object-oriented features**, such as abstraction, a suitable inheritance hierarchy, overloaded/overridden methods, abstract classes, interfaces, data hiding, encapsulation and polymorphism. You should also provide an appropriate interface (e.g. constructors, methods and fields) to your classes.

Deliverables

You will need to submit the following:

1. A number of Java classes that implement the different bank account types as described above *N.B.* only submit your Java (`.java`) code files;
2. A clearly recognisable loader/driver class (containing a `main()` method) that creates instances of each account type and demonstrates the key operations;
3. A short README text file to explain how to run your system;

4. A document that contains:
    - A list of the key requirements;
    - Appropriate UML diagrams which describes the classes in the system, their attributes and relationships (this should include a UML class diagram as the minimum);
    - A brief written overview of each class and any key design decisions (such as why you have chosen to implement certain methods or relationships)

Assessment

Key criteria for marking your submission will be: an appropriate and justifiable design, error-free compilation, program correctness and functionality, exception handling, appropriate use of object-oriented techniques and data structures, proper program structure, usability, source code quality and adherence to standard Java coding conventions.

Marks will be allocated according to the following scheme:

- **30% for the design component:**
    - Appropriate and comprehensive UML diagrams: **15%**
    - Key requirements and concise written overview of classes and design decisions: **15%**
- **70% for the programming component**:
    - Appropriate classes and use of object-oriented programming techniques: **20%**
    - Provision of appropriate bank account functionality through fields, constructors and methods: **20%**
    - Exception handling, programme robustness, handling malformed input from user: **10%**
    - Source code quality (readability, maintainability, comments, naming conventions) and following submission criteria (including README): **10%**
    - Innovation and extension to the base specification: **10%**

Submission

The deadline for this assignment is **MIDNIGHT, SUNDAY 20 MARCH 2016**.

Submission will be online via the BCO5008 Assignment area on Moodle; your assignment should be a compressed zip archive with the name `username.zip` (e.g. `sm19056.zip`). Before you upload your assignment, make sure that the compressed archive contains all necessary files and unzips into an appropriately named directory. Ensure that you complete an assignment submission form (also available on Moodle) and include this in your compressed archive.

All submitted code should compile and run under the Java 7 JDK as installed on the Cardiff Met network. If you decide to use Java 8 API features, please draw attention to how and why you have decided to used them. You do not have to use Eclipse to complete this assignment, but it is recommended IDE for the module.

The assignment should be conducted individually. **Attention is drawn to the University rules on plagiarism and collusion**; please refer to Section 2.6 of the Cardiff Metropolitan University Student Handbook[1].

[1]See: `http://www.cardiffmet.ac.uk/studenthandbook/`

Support

Questions regarding this coursework should primarily be asked in the tutorial sessions. General questions can also be posted on the BCO5008 Moodle forum (use of the forum is strongly encouraged, but please adhere to the forum rules regarding individual specifics and posting code) or via email to me (`tcrick@cardiffmet.ac.uk`).

You should also make use of appropriate Java reference books and the Java 7 API[2] and Java 8 API[3], along with referring back to the content covered in the lectures over Terms 1 and 2.

Feedback

Individual and group feedback will be provided via Moodle during the Easter vacation to aid your exam preparation. Note that any mark you receive for the assignment is preliminary and subject to comfirmation at the June examination board.

Grading Criteria

>70% A fully working application that demonstrates an excellent understanding of the assignment problem using relevant object-oriented programming features. Able to write and debug programs in Java using a range of system libraries and APIs, with a excellent understanding of modelling of real-world problems in an OO framework.

60%–69% A fully working application that demonstrates a good understanding of the assignment problem using relevant object-oriented programming features. Able to write and debug programs in Java using a range of system libraries and APIs, with a good understanding of modelling of real-world problems in an OO framework.

50%–59% A working application that demonstrates some understanding of the assignment problem using relevant object-oriented programming features. Able to write and debug programs in Java using system libraries and APIs, with some understanding of modelling of real-world problems in an OO framework.

40%–49% A partially working application that demonstrates satisfactory understanding of the assignment problem using object-oriented programming features. Able to write and debug basic programs in Java using system libraries and APIs, with satisfactory understanding of modelling of real-world problems in an OO framework.

<40% A partially working or poor application that demonstrates little understanding of the assignment problem using object-oriented programming features. Not able to write and debug basic programs in Java using system libraries and APIs, with little understanding of modelling of real-world problems in an OO framework.

[2] `http://docs.oracle.com/javase/7/docs/api/`
[3] `http://docs.oracle.com/javase/8/docs/api/`