Department of Computer Science
Faculty of ICT
University of Malta

Dr Neville Grech
neville.grech@um.edu.mt

CPS2004 Object-Oriented Programming

Study-Unit Assignment

November 13, 2019

Preamble: This document describes the assignment for study-unit *CPS2004: Object Oriented Programming*. This assignment is worth **100%** of the total, final mark for this unit. You are expected to allocate approximately 65 hours to complete the assignment. The deadline for this assignment is **Tuesday, 14th of January, 2020 at 15:00**. Late submissions will **not** be accepted. Questions regarding the assignment should **only** be posted in the Assignment VLE forum.

Unless otherwise stated in the task description, this is an individual assignment. Under **no** circumstances are you allowed to share the design and/or code of your implementation with fellow students. You may **not** copy code from internet sources, you will be heavily penalized if you do so! The Department of Computer Science takes a very serious view on plagiarism. For more details refer to plagiarism section of the Faculty of ICT website[1].

Important Notice

The main objective of this assignment is to demonstrate you understood the OO concepts presented in class. Submitting a perfectly working solution, without the use of OO principles will result in a failing grade.

1 Deliverables

You are to regularly commit all of your code to a GitLab private repo (as shown during the first tutorial). In the documentation, please state the **last commit hash** and **repository URL** on the front page. The documentation is to be submitted on the VLE website. Failure to submit this in the required format will result in your assignment not being graded. Please give me

[1] https://www.um.edu.mt/ict/Plagiarism

(nevillegrech) read access to your GitLab repo as soon as you start working on this. Marks will also be deducted if you fail to commit regularly to this GitLab repo. Only one file is required for electronic submission. Replace `NAME` and `SURNAME` with your name and surname respectively. Replace `IDCARD` with your national id. card number (without brackets), e.g. `123401G`.

- **`2020_CPS2004_SURNAME_NAME_IDCARD_assignment_code.tar.gz` -** An archive containing your assignment code. This needs to be uploaded to VLE. Each task should be located in a top level directory in the `.tar.gz` file named `task1`, `task2`, and `task3`. It is your responsibility to make sure that this archive file has uploaded to VLE correctly (by downloading and testing it). Failure to opening the archive will result in your assignment not being graded.
- **`2020_CPS2004_SURNAME_NAME_IDCARD_assignment_doc.pdf` -** The assignment documentation in `.pdf` format. The documentation has to be uploaded to VLE, together with your code. A hard-copy should be submitted to the secretary's office (Kevin Cortis) at the Computer Science department by the stipulated deadline.

  The report should **not** be longer than 14 pages (including figures and references) and, for each task, should contain the following:
  - UML Diagram showing the classes' makeup and interactions
  - A textual description of the approach (highlighting any extras you implemented)
  - Test cases considered (and test results)
  - Critical evaluation and limitations of your solution
  - Answers to any questions asked in the task's description

  This report should **not** include any code listings.
- **Signed copy of the plagiarism form -** This should be submitted to the secretary's office at the Department of Computer Science.

2 Technical Specification

Your code will be compiled and run on Linux. C++ code will be compiled using the `g++` compiler (version 7.4.0) from the GNU Compiler Collection. Please make sure to compile C++ using the command line:

```
g++ -Wall -std=c++17 *.cpp
```

Your Java submissions will be compiled and run using the OpenJDK (version 11). Please make use of standard libraries only. While you are free to use any IDE, make sure that your code can be compiled (and run) from the command line.

Failure to do so will have a severe impact on your grade. As a requirement, add a bash script (`compile.sh`) in each task directory to compile your task.

Also make sure to add a bash script (`run.sh`) in each task directory to execute your task. In these bash scripts, include any command line arguments and test data files – if applicable. Each task should have a `Launcher` file (either `.java` or `.cpp`) which contains a `main(...)` method used to run your solutions. **Note that you should not have any absolute paths hardcoded in your programs/bash scripts.**

3 Tasks

This assignment consists of three programming tasks. You may use C++ or Java for the first task. The second task should be implemented in Java and the third task in C++.

3.1 Immutable List (in C++ or Java)

Object oriented design is not just about designing a system from scratch but also about adapting existing systems and improving their design. Take a look at the specification for lists in C++ or Java:

`https://en.cppreference.com/w/cpp/container/list`

`OR`

`https://docs.oracle.com/javase/8/docs/api/java/util/List.html`

These specifications were designed over 20 years ago when mutable shared state concurrent software was the default (and it still is in many contexts). Another way to design robust concurrent software is to avoiding shared mutable state, which requires the use of immutable data structures.

1. Find which method signatures from this interface need to change in order to support an immutable list.
2. Redesign the core parts of this interface in order to implement an immutable list. You will need to include some operations to add/remove/copy/clear items from a list. You don't have to include *all* of the functionality however, especially where there is repetition.
3. Implement an immutable list (e.g. a linked list), which implements this interface or abstract class.
4. Test immutable lists by adding and removing elements in a concurrent environment with no locking.
5. Bonus: Why does an immutable list perform worse than its mutable counterpart in some operations? Propose optimizations for this list such that the performance of it can improve.

(Total for task: 33 marks)

3.2 Exchange Platform (in Java)

Design a simple stock exchange platform, emphasizing the concepts in **boldface**. Implement some simple functionality where appropriate and write unit tests that exercise this logic.

An exchange platform is a facility where traders can buy and sell securities, such as shares of stock and bonds and other financial instruments. Users of a stock exchange have to **REGISTER** and be **APPROVED** by the **EXCHANGE PLATFORM OPERATOR** in order to use the system. There are two types of users: **TRADERS** and **LISTERS**. To be able to trade a **SECURITY** on this stock exchange, the **SECURITY** must be **LISTED**. Securities include a description, price, and total supply.

The exchange platform maintains an **ORDER BOOK**. Orders can be **BUY ORDERS** or **SELL ORDERS**. Orders are raised by a trader, and include a **QUANTITY**, **PRICE** and **TIMESTAMP**. When orders are sent to the platform, the platform places these on the order book, which is publicly visible. The system will try to fulfill orders by matching buy and sell orders together using a **MATCHING ENGINE**. Some orders will be fulfilled, some won't. Orders can also be **CANCELLED** by either the exchange, lister or trader.

Please design your system in such a way as to facilitate future changes, because due to the changing regulatory landscape, additional requirements may come up. Please justify how your system can be easily changed to implement the following requirement:

- Audit trails: every action by a user needs to produce a log (that cannot be tampered with) so that any fraudulent activity can be investigated by competent authorities and scenarios recreated.

(Total for task: 34 marks)

3.3 Expression Language (in C++)

Write C++ templates which allow you to use types to represent arithmetic expressions over the four basic operations (addition, subtraction, multiplication and division) and a single variable: (e.g. $x * x + (x - 2)3/5$). For the purposes of this question we refer to the single variable as x. The types representing the expressions should contain a function named `eval` that provides code for evaluating the expression. This function should accept a double parameter that will be used as the value for the single variable x.

```
E.g.,
  double res = ((x() * x())  + c(14)).eval(4);
  printf("%f\n",res);
```

Use the expression templates to write a C++ template which accepts arithmetic expressions and **generates code at compile time** for numerically approximating the integral of that expression with respect to the single variable. Your template should provide a function named `integrate` which accepts double values as the lower and upper limits of the range to be integrated over. This function should also accept an integer value which specifies the number of divisions of the range by which the integral is approximated. Show how to use this template to generate code for a short example program which numerically integrates the arithmetic expression:
$x^2 + 2 * x - 3$

```
E.g.,
  double res2 = integrate(((x() * x()) + c(2)*x() - c(3)),1,2,10000);
  printf("Integrating x^2+2x-3 over 1<x<2 (ans 7/3): %f\n",
         res2);
```

(Total for task: 33 marks)

4 Grading Criteria

The following criteria, described in Table 1, will be taken into consideration when grading your assignment.

Table 1: CPS2004 Assignment grading criteria. Equal consideration will be given to each.

Overall Considerations	
OO Concepts	Demonstrable and thorough understanding and application of OO concepts and design. You should make use of most of the concepts presented during lectures.
Thoughtful Design	As presented during lectures and clearly documented. Please include: design (in UML), technical approach, testing, critical evaluation and limitations.
Functionality	Completeness and adherence to tasks' specification. Correctness of solutions provided.
Quality and Robustness	Proper error handling. Proper (and demonstrable) testing of solutions. No memory leaks or wasted resources.
Programming Skill	Concise, precise, easy to understand code utilizing appropriate Java and C++ features.
Environment	Use of Linux/Unix setup, version control and reliable build environment.

Note that **not** submitting one (or more) of the tasks will severely affect your overall mark.