COMP7505 Assignment 1

This assignment is worth **15%** of the total course marks.

Administration

Due 5pm Friday 3 September 2010 (Week 6)

Assignments will be returned during tutorials. Refer to the course profile for policy regarding late submission of assignments.

School Policy on Student Misconduct

You are required to read and understand the School statement on student misconduct, available at:

http://www.itee.uq.edu.au/about_ITEE/policies/student-misconduct.html

In particular, you must familiarise yourself with UQ's definition of plagiarism, and understand the School's policies regarding student plagiarism.

Submission

The assignment must be submitted through the UQ Blackboard site:

http://blackboard.elearning.uq.edu.au/

All parts of the assignment must be in a single zip archive, created by exporting the modified project from Eclipse. To create the archive:

- Select the File → Export… menu item
- Select General → Archive File (as the export destination)
- Enter the destination file name, which must be `Student<your_student_number>.zip` with `<your_student_number>` replaced by your full eight-digit student number.

Export the project only **after** you have completed all parts of the assignment. Please ensure that your exported archive file contains all of your work before submitting it.

Coding Style

Your code must be formatted using the built-in Eclipse coding style. Open the Eclipse "Preferences" window, go to Java → Code Style → Formatter and under "Active Profile" make sure that "Eclipse [built-in]" is selected. You should reformat your code to follow this coding style before submission, by selecting Source → Format or Source → Format Element.

Resources and restrictions

You **may not** use any classes from the Java Class Library except those in the *java.lang* package. You **may not** use any code that you did not author, except for code provided to you by us.

To complete this assignment you must first download an Eclipse archive file that contains a set of ADTs, a pair of data structures, and some JUnit tests. This file is available at:

http://www.itee.uq.edu.au/~comp3506/Resources

Import the downloaded archive into a new Eclipse workspace using the "Existing Projects into Workspace" option:

1. Select the File → Import menu item
2. For the Import Source, select General → Existing Projects into Workspace and click "Next"
3. Select "Select archive file", "Browse", and select the archive file and click "OK"
4. Select the available project and click "Finish"

Introduction

Assignment 1 is divided into four parts worth 10 marks each; these 30 marks comprise 15% of the overall course mark. The goal of the assignment is to familiarise you with the process of implementing Abstract Data Types (ADTs) and evaluating strengths and weaknesses of different implementations.

Background Context

A small team in your company is working on a next-generation social-web blog-integration smart-phone client platform. As the software runs on resource-limited devices, management has decided that dependency on any external frameworks is unacceptable, and so all code, including all basic data structures, must be written from scratch.

You have been assigned to implement a variety of data structures for the software's scheduling system. The previous programmer working on the project has left you a variety of interface definitions and half-finished code. Your job is to take the existing project, bring the implementation up to standard, and flesh out the functionality of the system.

Part 1 – Testing (5 marks total)

Among other things, your predecessor left array-based queue and stack classes unfinished. While their definition and functionality are already in place, they possess five known errors. Even though these classes are unfinished, your boss has decided that fixing the known errors will make the data structures suitable as a stop-gap implementation that can be used for the scheduling mechanism.

Luckily, several test cases have been implemented to help you to track down the errors. You need to identify and fix the errors exposed by the test cases. When you have finished, all of the test cases should run successfully.

Task 1

You must find and correct the **five** errors that exist in the ArrayStack and ArrayQueue classes. For each error you **must** add a **comment** that describes the problem with the existing code and why your new code is now correct. **Each error is worth 1 mark**.

Do not change the tests or interfaces. If you wish to create more tests, do so in another package.

See "Resources and restrictions" above. The code you have imported contains both implementation and test source in the "COMP3506-A1" project, in the `au.edu.uq.itee.comp3506.part1` package.

Running the test sets will aid you in tracking down the problems, but note that there is not necessarily a one-to-one correspondence between test errors and code errors. You should read the supplied comments thoroughly so that you understand what the code is intended to do.

Part 2 - Implementation (7 marks total)

With the fixed array-based stack and queue implementations, the scheduler now runs. However, another team working on the project has encountered several situations where the performance of the array-based implementations is unsatisfactory. You have been asked to implement a pair of linked-list-based stack and queue classes.

In order to easily allow run-time decisions about which data structures to use for different scheduling requirements, you must also implement a factory class that can construct your new stack and queue classes, as well as the classes you fixed in Part 1.

Task 2

Implement the Stack (IStack.java) and Queue (IQueue.java) ADTs provided using an appropriate linked list implementation (8 marks). Do not modify the provided interfaces. Your classes must be called ListStack and ListQueue, and must correctly and efficiently implement the respective interfaces with the expected time and space complexities of a link-based structure implementation, as discussed in lectures.

You must also produce a class that implements the IADTFactory interface (IADTFactory.java) and allows choice between ListStack and ArrayStack, and ListQueue and ArrayQueue (2 marks). Your class should be called ADTFactory, and will return instances of your ADTs based upon the passed identifier. The identifier for each class must be the name of the class.

See "Resources and restrictions" above. You will use the interfaces specified in the already-imported "COMP3506-A1" project. You may also use any classes that were given to you in the imported project. All classes for this task must be implemented within a package called "**student*yourstudentnumber***". For example, if your student number is 12345678, you would declare the package as:

```
package student12345678;
```

Note: Parts of your assignment will be automatically marked. It is essential that you correctly implement the **IADTFactory** interface in order for Part 2 of your assignment to be marked. If you do not properly implement the **ADTFactory** class you risk **receiving no marks for part 2**.

Part 3 – Analysis (8 marks total)

Now that your team has access to a range of stack and queue implementations, and a factory mechanism which allows swapping out these implementations on the fly, your boss wants to formulate a policy on which implementation should be used in different situations. In order to assist with this, you need to document and explain the characteristics of each of the available algorithms. In addition, you have been asked to outline possible future directions for development.

Task 3

Write a document detailing:

- the differences in computational complexity between your list- and array-based implementations;
- examples of scheduling situations in which one implementation would provide an advantage over the other;
- a suggested policy – based on space and time complexity trade-offs – for choosing between

implementations based on the run-time scheduling requirements; and

- possibly future extensions and improvements in scheduling and the required data structures.

You document should be no more than 400 words/one A4 page, and should clearly address each of the above points in its own section. Include this document in **PDF format** with the filename "Task3.pdf" in the top-level directory of the "COMP3506-A1" project.

Part 4 – Challenge Question (10 marks total)

The stack and queue structures, while functional, do not provide the development team with enough flexibility to handle certain situations. In order to handle simple priority-based scheduling, you have been asked to provide a quick prototype priority queue implementation. This implementation is not expected to ship, but to provide a functional interim component, and need not therefore be efficiently implemented.

Task 4

Implement the IPriorityQueue (IPriorityQueue.java) ADT provided using your choice of an array or a linked list (2 marks). Your class must be called ProtoPriorityQueue and must correctly implement the interface. However, note that an efficient implementation is *not* expected.

You must ensure that your ADTFactory from Part 2 can return an instance of your priority queue class, under the same identifier rules as above.

See "Resources and restrictions" above. You will use the interfaces specified in the already-imported "COMP3506-A1" project. You may use any code in this project. All the classes for this task must be implemented within a package called "**student*yourstudentnumber***". For example if your student number is 12345678, you would use the following line to declare the package:

```
package student12345678;
```

As with Part 2, correct implementation of ADTFactory is essential.

Task 5

Finally, you are to produce a short description describing the weaknesses of your priority queue implementation in terms of computational complexity, and detail how these weaknesses could be overcome using a different underlying data structure (1 mark). Include this description as a separate section at the end of the document you produced for Task 3.