Faculty of Engineering, Environment and Computing

M19COM Advanced Software Design Development

Assignment Brief 2018/19

Family Name(s)	Forename(s)		ID Number(s) (from your student card)
Module Title: Advanced Software Design Development	Ind/Group Individual	Cohort(Sept/Jan/May): 1819OctJan	Module Code: M19COM
Coursework Title: Assignment 1			Hand out date: **17 October 2018**
Lecturer: Dr Hong Guo			Due date: **16 November 2018 at 15:00** **In-class Test in room JA143**
Estimated Time (hrs): 25 Word Limit*: Not applicable	Coursework type: Programming Assignment		% of Module Mark 50%

Submission arrangement:

Your submission must be both paper-based and electronic.

Paper-based submission: Please complete, print and include the first page of this cover sheet with your paper submission. You should submit a single document with this coversheet on paper that contains:

- ***Design***
  - A UML design class diagram for extended system design with full details.
  - Description of basic design patterns applied in the design class diagram.

- ***Implementation and source code – this should be unedited and no screenshots!***
  - Printed listings of all the classes completed: Operations, AircraftOperationsMain, Aircraft, AircraftHelper.
  - Printed Javadoc output for the Operations class.

- ***Test plan and output of run***
  - Your test plan should identify all implemented functionality, test data and expected outcomes.
  - Printed output from a final test run, complete and unedited, showing the full testing of all the methods implemented in Operations, including the full testing of pre and post conditions of the `add()`, `remove()`, `get()` and `sort()` methods.

This assignment is assessed through an In-class Test that is worth 100% of the assignment mark. You must bring this assignment to the test and hand it in with your test paper, or you will not be able to do it. The test will be 60 minutes long and will cover each of the tasks specified.
Note: the test is "closed book" and "no notes" except for your assignment.

Electronic Submission of Source Code Files:

- You must upload electronic copies of your own Java source code files (Operations.java, and AircraftOperationsMain.java) on Moodle via the Electronic Submission of Source Code Files submission link. The source-code plagiarism detection software will be used to check for source code plagiarism. You must work independently, please.

The Faculty of Engineering, Environment and Computing Policy on Assessed Coursework applies to this coursework. You are advised to read the guidelines available on the EC Student Web on CU Portal. Keep a safe copy of all coursework submitted for reference.

Mark and Feedback date: 2 week after date of submission.
Mark and Feedback method: Assessed through in-class test, feedback will be written on your test script.

Module Learning Outcomes Assessed:

- Demonstrate a practical understanding of core OO programming concepts focusing on definition, extension and reuse, that is explained though to OO design and GRASP (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns) Principles.
- An ability to understand and design-by-contract principles for correctly implementing and testing abstract data (ADT) type specifications in the OOP context.
- Select suitable data structures and use appropriate OOP techniques in designing and implementing ADT classes as exemplified through the Java Collections Framework.

Task and Mark distribution: Assessed through In-class Test

Assignment 1: Coursework Brief

An international airport's maintenance operations are required to keep track of the number of flights each aircraft they service operates per month, in order to schedule regular maintenance checks. A Java application has been developed which monitors the flight operations of each aircraft and adjusts the aircraft's total number of flights to reflect data received from its daily operations.

For this assignment we focus on the classes that represent aircraft and lists of aircraft.

A class Aircraft to represent an aircraft is provided, and this must not be modified. Each aircraft has a name (String), type (String), number of flights (int) and a unique registration (String). Comparisons and equality tests are based on registration.

An interface AircraftList is also provided, with Javadoc comments describing the expected behaviour and contracts of the methods to be implemented in a new class: Operations.

You are required to implement the Operations class, test this class and then show and discuss how the design of the system can be extended to include more specific subclasses of aircraft.

The complete set of source files for this assignment should be downloaded from Moodle.

Task 1 – Initial System Design

Show using UML diagrams the existing system classes and their associations and dependencies.

Task 2 – Implementation of Operations Class

Implement all the methods specified in AircraftList in a new class Operations, taking care to observe their contracts. The internal data structure to be used for storing aircraft objects should be an *unsorted* Java Collections data structure such as: ArrayList. You should add a suitable representation invariant and Javadoc comments for this class.

Task 3 – Testing of Operations Class

Test your implementation using the supplied AircraftHelper and AircraftOperationsMain classes, writing code in the AircraftOperationsMain class to test fully all the methods in the Operations class, including the add(), remove(), get() and sort() methods against their specifications. A single test run should clearly produce all the evidence to verify the correct operation of those methods, this should include the testing of the sort method showing a list of at least 25 aircrafts which are sorted correctly.

Task 4 – Extended System Design and Application of Basic Design Patterns

Show using UML diagrams how you can extend the system to use two subclasses: PropellerAircraft and JetAircraft that inherit from Aircraft. Note the subclasses should inherit all the methods from Aircraft except for getType() which should be overridden in the new derived classes to return the correct type of the aircraft. Describe what basic design patterns and principles this shows as being applied.

Note that the UML diagram for task 1 and 4 can be combined into a single UML diagram showing all the system design classes including the extended design classes.

Notes:

1. You are expected to use the CUHarvard referencing format. For support and advice on how this students can contact Centre for Academic Writing (CAW).
2. Please notify your registry course support team and module leader for disability support.
3. Any student requiring an extension or deferral should follow the university process as outlined here.
4. The University cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computer. Students should therefore regularly back-up any work and are advised to save it on the University system.
5. If there are technical or performance issues that prevent students submitting coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will be communicated via email and as a CUMoodle announcement.

Overall Mark Guidelines To Students

0-39	40-49	50-59	60-69	70+	80+
Work mainly incomplete and /or weaknesses in most areas	Most elements completed; weaknesses outweigh strengths	Most elements are strong, minor weaknesses	Strengths in all elements	Most work exceeds the standard expected	All work substantially exceeds the standard expected

Assessment Criteria

Assignment is assessed via In-class test that is worth 100% of this assignment mark.

General guidelines for a pass-level mark (40%):

- A basic description with UML diagrams of the existing system classes.
- You will need to produce a working and partially tested implementation of the required classes.
- Basic description with UML diagrams showing how the design can be extended as required, describing the application of one basic GRASP design pattern introduced so far in the module.

General guidelines for a distinction-level mark (70% or above):

- A detailed description with UML diagrams of the existing system classes.
- Your implementation of the required classes should be accurate, the testing should be adequate.
- Detailed descriptions with UML diagrams showing how the design can be extended as required, describing the application of two basic GRASP design patterns introduced so far in the module.

Note: *Marks will be lost for failing to follow the instructions above and in comments within the supplied files, and for incomplete submissions, e.g. no UML diagrams, Javadoc, limited testing of required class methods.*

Code for The AircraftList Interface

```
/**
 * Interface for a class representing a list of aircraft *** DO NOT MODIFY ***
 * @author Hong Guo
 * @version 1.3 - 1/10/18
 */
public interface AircraftList
{
    /**
     * Add a new aircraft to the list, if not already there
     * @param aircraft  the aircraft to add
     * @return  true if successful, else false
     * throws NullPointerException if aircraft is null
     */
    public boolean add(Aircraft aircraft) throws NullPointerException;
    // POST: if aircraft is null the exception is thrown; else
    // the aircraft is added to our list and true is returned,
    // or false if the aircraft was not added for some reason


    /**
     * Remove an aircraft from the list
     * @param registration  the registration of the aircraft to remove
     * @return  true if successful, else false
     * @throws NullPointerException if registration is null
     */
    public boolean remove(String registration) throws NullPointerException;
    // POST: if string is null the exception is thrown; othewise
    // if the aircraft is not in our list false is returned, else
    // matching instance of the aircraft is removed and true is returned


    /**
     * Get an aircraft given the aircraft's registration
     * @param registration  the aircraft's registration string
     * @return  the aircraft if found, else null
     * @throws NullPointerException if registration is null
     */
    public Aircraft get(String registration) throws NullPointerException;
    // POST: if string is null the exception is thrown; otherwise
    // if an aircraft is found matching the specified registration
    // then that aircraft is returned, else null


    /**
     * Get the size of the list (number of aircraft)
     * @return  the list size
     */
    public int size();
    // POST: the list size (= number of aircraft) is returned


    /**
     * Convert the list to a single string "S0\nS1\nS2\n..."
     * @return a printable string formatted as above
     */
    public String toString();
    // POST: a string is returned of the form "S0\nS1\nS2\n..."
    // where each Si is the string representation of ith aircraft


    /**
     * Sort the list of aircraft into ascending order by registration
     */
    public void sort();
    // POST: the list of aircraft are sorted in ascending order by registration
}
```

See also the classes Aircraft, AircraftHelper and AircraftOperationsMain. The complete set of source files can be downloaded from Moodle as an archive **M19COM_Ass1_SourceCode.zip** and extracted for use in JGRASP.