

AY: 2019/20

CIS6003 – Advanced Programming - 20 credits

Term 1

Module Leader: Dr. Ambikesh Jayal

Assessment Brief

Part 1: An interactive two-player battleship game using client-server architecture - 50%

Part 2: Worksheets Portfolio - 50%

WRIT1: 100%

HAND-OUT DATE: 13-01-2019

HAND-IN DATE: Refer to Moodle

Contents

[Learning Outcomes 3](#_Toc18668405)

[EDGE 3](#_Toc18668406)

[Assessment Requirements / Tasks 3](#_Toc18668407)

[Assessment Criteria 7](#_Toc18668408)

[Submission Details 7](#_Toc18668409)

[Feedback 7](#_Toc18668410)

[Marking Criteria 9](#_Toc18668411)

[Additional Information 11](#_Toc18668412)

[Referencing Requirements (Harvard) 11](#_Toc18668413)

[Mitigating Circumstances 11](#_Toc18668414)

[Unfair Practice 11](#_Toc18668415)

# Learning Outcomes

This assessment is designed to demonstrate a student’s completion of the following Learning Outcomes:

* Demonstrate fluency in contemporary programming languages, development tools and environments.
* Evaluate and demonstrate the theory and concepts of contemporary/industry standard programming and design in the software development life cycle.
* Demonstrate awareness of industry standards of professional and ethical software development, software carpentry and foremanship

# EDGE

The Cardiff Met EDGE supports students in graduating with the knowledge, skills, and attributes that allow them to contribute positively and effectively to the communities in which they live and work.

This module assessment provides opportunities for students to demonstrate development of the following EDGE Competencies:

|  |  |
| --- | --- |
| ETHICAL | Usability, and user experience are two important factors when designing and implementing a software application that is interacted by the users. The information, messages generated by the system, and information collected by the client program (e.g., personal data) should be handled, saved and processed with proper care adhering to international and regional laws and regulations. |
| DIGITAL | Software engineering and programming skills accumulated during the course and through the assignment will enhance student’s hands on digital skills, use of IDEs, build tools, inter-component/process communication skills, and industry standard programming and design best practices. |
| GLOBAL | Software Engineering principles design patterns discussed in this class universally accepted. Skills gained through this are location and language agnostic and are applicable across the globe. |
| ENTREPRENEURIAL | Problem solving skills, knowledge acquired on client-server architecture, programming and system designing skills are pivotal for anyone who expects to aspire to be an entrepreneur in a technical domain. |

# Assessment Requirements / Tasks

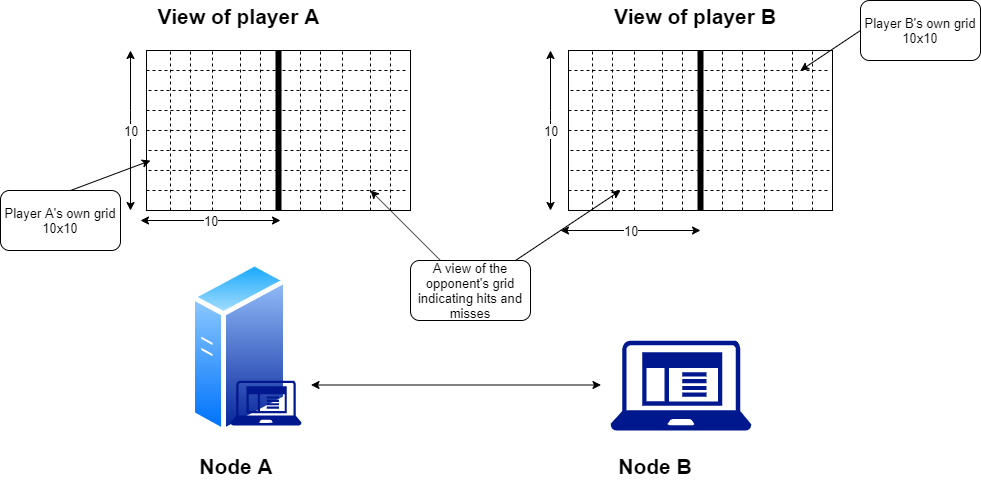
**Part 1: An interactive two-player battleship game using client-server architecture**

You are required to implement an interactive application using a ***client-server architecture*** ***simulating*** a two-player battleship game.

A description of how the actual game works is given in the following link.

<https://www.youtube.com/watch?v=4gHJlYLomrs>

The system consists of software simulating server component and a client component (that can be executed as two separated programs to simulate 2 players).



C

D

E

F

Figure 1: A high level overview explaining the client-server architecture and the overall concept

The implementation should handle the following two main use cases.

**Use case 1:** The system should facilitate two players to play against each other. In this case, Node A (that represents a server) should allow another player to manually place ships, attach opponent’s ships using the user interface.

**Use case 2:** The system should facilitate a player to play against the PC. In this case, node A that represents a server should act as the 2nd player. In this case, the placement of ships, attacking opponent’s ships should take place as random events. You may use a simple random number generator to achieve this.

The overall system should be based on client-server architecture and in one of the nodes (either node A or B) should represent a server component and other represents a client component.

The following steps should occur until all ships of a player is destroyed by an opponent.

***Loop: until all ships of a player is destroyed by the opponent****;*

*Each node should generate a user interface to facilitate for the players to mark locations of the ships as preferred (Views* ***C*** *and* ***F****). The user interface should facilitate player to guess the locations of opponent’s ships (views* ***D*** *and* ***E****).*

*When a player guesses (sender) an opponent’s ship’s location using the views D or E, this location is communicated to the other node (receiver) and that node will update the players’ main view (C or F) to indicate the outcome. If it is a hit, corresponding location of the opponent’s ship is marked appropriately (e.g., a red box).*

*The outcome (hit or a miss) is communicated back to the sender and will be reflected in sender’s opponent view pane (D or E) appropriately.*

*E.g., Red marker for a hit, white marker for a miss.*

*Update the status of the ships on each players view E.g., indicate how many ships destroyed so far.*

***End***

General functions of the server and client components in a client-server architecture are described below.

***Server***:

* The server is initiated by establishing a server socket on a given port number.
  + The port number can be given as a command line argument
* Server should wait until a client is connected before starting the game.
* Once a client is connected, communication initiates, and both entities should be able to communicate with each other.
* *Note: Ensure all relevant threads required to a server component are implemented.*

***Client:***

* Once the server is initiated, client can connect to the server using the server’s host IP address and port number.
  + These may be provided as command line arguments when initiating a client.
* Once connected to the server, client can start communicating with the server component.
* *Note: Ensure that all relevant threads required for a client component are implemented.*

***Design and Implementation requirements***

You are required to submit the software components (source code), and relevant documents that contain the following

* **UML use case diagram(s):** explaining the use cases. Key features such as generalization, <<include>>, <<extension>> relationships should be captured. The diagrams can be further explained using use case model
* **UML class diagram:** describing the entities and their associations. Key features such as inheritance hierarchies, associations with multiplicities, aggregation/compositions (if applicable), interfaces (if applicable) should be identified
* UML Notes should be used where applicable to explain the design approaches
* **Source files** corresponding to the software component simulating a **server** implementation
* **Source files** corresponding to the software component simulating a **client** implementation
* **‘Ant’ build.xml files** (one for server, one for client) that can be used to build the java source files.
* **Documentation (Word/PDF document)** explaining the steps required to run your **server** component, and the steps required to run **client** component that simulate players.
  + The documentation should include the UML diagrams
  + Discuss the use of different design patterns within the implementation.
* **A link to the git remote repository:** The documentation should **include a link** to the git remote repository (bitbucket preferred) with proper access being granted to the tutors (not publicly shared).

A zip file containing these should be uploaded to Moodle as explained in “*submission details*” below.

# Assessment Criteria

|  |  |
| --- | --- |
| Completeness of the submission package | 5% |
| UML use-case diagrams | 5% |
| UML class diagrams | 10% |
| Documentation explaining the implementation, execution details | 10% |
| Implementation of classes identified in the class diagram | 10% |
| Use of OOP concepts within the design and implementation | 20% |
| Proper use of threads | 5% |
| Appropriate Exception and Error handling | 5% |
| Use of suitable design patterns within the implementation | 10% |
| Coding skills and best practices | 5% |
| Implementation of the client-server architecture | 5% |
| Overall execution of the system | 10% |
| Total | 100% |

# Submission Details

Please see Moodle for confirmation of the Assessment submission date.

Submission will be by **4:00pm** on the deadline day.

**Any assessments submitted after the deadline will not be marked and will be recorded as a Non-Attempt.**

The assessment must be submitted as a .**zip** file through the submission point in [Moodle](https://learn.cardiffmet.ac.uk/)

The ***zip*** file **MUST** contain all the project elements defined in **Sec. *Design and Implementation requirements***.

NOTE: submission formats other than ***zip*** are NOT accepted and will not be graded during the marking process.

Your assessment should be titled with your Student ID Number, module code and assessment id, e.g. st12345678CIS5003WRIT1.zip

**Part II: Worksheets Portfolio (50%)**

**Worksheets portfolio:** For the assignment you need to work on worksheets about various topics on industry standard programming and design in the software development life cycle. All students need to submit **worksheets and related code in a zip folder**. The purposes of these weekly worksheets is to explain the key issues in modelling and analysing real world problems using appropriate techniques. Further details can be found from Moodle.

# Feedback

Feedback for the assessment will be provided electronically via [Moodle](https://learn.cardiffmet.ac.uk/), and will normally be available 4 working weeks after initial submission. The feedback return date will be confirmed on Moodle.

Feedback will be provided in the form of a rubric and supported with comments on your strengths and the areas which you improve.

All marks are preliminary and are subject to quality assurance processes and confirmation at the Examination Board.

Further information on the Academic and Feedback Policy in available in the Academic Handbook ([Vol 1, Section 4.0](http://www.cardiffmet.ac.uk/registry/academichandbook/Documents/AH1_04_00.pdf))

# Marking Criteria: part 1

|  |  |
| --- | --- |
| **70 – 100%**  **(1st)** | An excellent design is given that covers all the requirements of the assignment. An excellent set of UML diagrams are given capturing the systems design and behaviours, demonstrating an exceptional understanding on the underlying OO design principles.  An excellent implementation is given covering all of the server-side and client-side functions. Outstanding coding practices are followed, git repository is well maintained with continuous commits and comments. Exceptional use of multithreading, exception handling (including custom exceptions), and excellent demonstration of Java OO programming concepts (nested classes, interfaces, collections etc) and design patterns.  Additional elements have been considered to improve usability and user experience of the system.  Excellent, detailed documentation is provided explaining the build, and execution of the server and client components, implementation and results |
| **60-69%**  **(2:1)** | A good design is given that covers most of the system requirements. A good set of UML diagrams are given capturing systems design and behaviours, demonstrating a good understanding on the underlying OO design principles.  A good implementation is given covering most of the server-side and client-side functions. Good coding practices are followed, git repository is maintained with some commits and comments. Good use of multithreading, exception handling, good demonstration of Java OO programming concepts, and knowledge in design patterns.  A good documentation is provided explaining the build, and execution of the server and client components, implementation and results |
| **50-59%**  **(2:2)** | An average design is given covering only some of the system requirements. UML diagrams capture some of the system’s design aspects and behaviours. However, use of OO design principles, correct UML notations, UML notes needs improvements.  An average implementation is given covering some of the server-side and client-side functions. Demonstrate average coding skills and git repository is not well maintained. Some of use of threads, exception handling and design patterns is visible.  An average documentation is provided explaining the build, and execution of the server and client components. |
| **40-49%**  **(3rd)** | A marginal design is given covering only few system requirements.  UML diagrams capture basic system’s design aspects and behaviours. However, use of OO design principles, correct UML notations, UML notes needs a lot of improvements.  A basic implementation is given covering only few of the server-side and client-side functions. Demonstrate basic coding skills and git repository is not well maintained. Basic (or no) use of threads and exception handling is visible and no use of design patterns. Basic documentation is provided explaining the build, and execution of the server and client components. |
| **35-39%**  **(Narrow Fail)** | A poor design is given covering only very basic system requirements.  UML diagrams capture very basic system’s design aspects and behaviours. However, use of OO design principles, correct UML notations, UML notes needs a lot of improvements (poor use of notations, syntax and concepts)  A very basic implementation is given covering minimal server-side and client-side functions. Demonstrate poor coding skills and git repository is not well maintained (or not provided). Poor (or no) use of threads and exception handling is visible. Very basic documentation is provided explaining the build, and execution of the server and client components |
| **<35%**  **(Fail)** | No meaningful design has been given and little to no code has been implemented. There is no evidence of any meaningful testing or analysis of the results obtained. Little to no understanding of the problem and the techniques required to create a solution are evident |

# Additional Information

## Referencing Requirements (Harvard)

The Harvard (or author-date) format should be used for all references (including images).

Further information on Referencing can be found at Cardiff Met’s [Academic Skills](https://study.cardiffmet.ac.uk/AcSkills/Pages/Referencing.aspx) website.

## Mitigating Circumstances

If you have experienced changes or events which have adversely affected your academic performance on the assessment, you may be eligible for Mitigating Circumstances (MCs). You should contact your Module Leader, Personal Tutor or Year Tutor in the first instance.

An application for MCs, along with appropriate supporting evidence, can be submitted via the following link to the [MCs Dashboard](https://cis.cardiffmet.ac.uk/MitCircs/MitCircs/StudentDashboard)

Applications for MCs should ideally be submitted as soon as possible after circumstances occur & at the time of the assessment. **Applications must be submitted before the relevant** [**Examination Board**](http://www.cardiffmet.ac.uk/registry/exams/Pages/Examination-Boards.aspx)**.**

Further information on the Mitigating Circumstances procedure is available in the Academic Handbook ([Volume 1, Section 5](http://www.cardiffmet.ac.uk/registry/academichandbook/Pages/Ah1_05.aspx))

## Unfair Practice

Cardiff Metropolitan University takes issues of unfair practice **extremely seriously**. The University has distinct procedures and penalties for dealing with unfair practice in examination or non-examination conditions. These are explained in full in the University's Unfair Practice Procedure (Academic Handbook: [Vol 1, Section 8](http://www.cardiffmet.ac.uk/registry/academichandbook/Pages/Ah1_08.aspx))

Types of Unfair Practice, include:

**Plagiarism,** which can be defined as using without acknowledgement another person’s words or ideas and submitting them for assessment as though it were one’s own work, for instance by copying, translating from one language to another or unacknowledged paraphrasing. Further examples include:

* Use of any quotation(s) from the published or unpublished work of other persons, whether published in textbooks, articles, the Web, or in any other format, which quotations have not been clearly identified as such by being placed in quotation marks and acknowledged.
* Use of another person’s words or ideas that have been slightly changed or paraphrased to make it look different from the original.
* Summarising another person’s ideas, judgments, diagrams, figures, or computer programmes without reference to that person in the text and the source in a bibliography or reference list.
* Use of services of essay banks and/or any other agencies.
* Use of unacknowledged material downloaded from the Internet.
* Re-use of one’s own material except as authorised by the department.

**Collusion**, which can be defined as when work that that has been undertaken with others is submitted and passed off as solely the work of one person. An example of this would be where several students work together on an assessment and individually submit work which contains sections which are the same. Assessments briefs will clearly identify where joint preparation and joint submission is specifically permitted, in all other cases it is not.

**Fabrication of data**, making false claims to have carried out experiments, observations, interviews or other forms of data collection and analysis, or acting dishonestly in any other way.