Programming 04 – Advanced Programming Assignment Brief

Contents

[Module Details 1](#_Toc472072871)

[Assignment Description 2](#_Toc1394001069)

[Summary 2](#_Toc1005299359)

[Deliverables 2](#_Toc863228761)

[Submission 3](#_Toc1676560024)

[Marking 4](#_Toc1642736131)

[Generative AI and Academic Misconduct 7](#_Toc948659252)

[Learning Outcomes 7](#_Toc820226427)

[Computer Science 7](#_Toc1600269383)

[Games Development 7](#_Toc2105623079)

[Software Engineering 8](#_Toc1225508154)

[Advice and Guidance 8](#_Toc2118296418)

[How is this assessment marked? 8](#_Toc719261896)

[Assessment Marking Criteria 9](#_Toc905005299)

| Module Details | | | |
| --- | --- | --- | --- |
| **Module code:** | COM5015M | **Level of Study:** | 5 |
| **Module Leader(s):** | David Gundry | **Credits:** | 20 |
| **Assessment format:** | Portfolio | **Method of submission:** | Moodle |
| **Deadline or Assessment Period:** | 2 June 2025 | **Feedback date and place:** | 24 June 2025 |
| **Assessment limits:** length, load, word count, etc. | N/A | **Component number:** | 1 |
| **Is this exempt from anonymous marking under the policy?:** | No | **Component weighting:** | 100 |

| Assignment Description |
| --- |
| Summary In this assessment you will create a basic chat application, including a server. There are specific functional requirements for this application that are the focus of the marking. These are given in the 'Required Functionality' section below. You should then try to address one or more of the additional requests from users in the ‘Extended Functionality’ section. However, briefly, the assessment involves creating two Java programs:   1. **Server:** The server application should allow the client to connect on a specified port. When connected, the server can receive chat messages from the client. The server should receive and store chat messages from clients. The server should send chat messages received to connected clients. The server should also serve a webpage with statistics and information about the chat. 2. **Client:** The client application should connect to a server at a given address and port. When connected to the server, it should be able to send and receive messages. Note that it is not a requirement that the client has a graphical user interface, however you may create a graphical user interface if you choose.  Deliverables The deliverables for this assessment are the following:   1. Generative AI and Third-Party Content Declaration 2. Video demonstration 3. Source code and related project files 4. Compiled Java files  1. Generative AI and Third-Party Content Declaration You must summarise your inclusion of AI-generated or AI-modified content (if any) and third-party assets or libraries across your project. This will take the form of a written declaration. This may be very brief or very long, depending on your use of third-party and AI-generated content.  This declaration should be brief and factual. It should indicate how you have recorded the source or licenses of content, assets, or code. Note this includes recording which assets were created using AI.  If it is not clear what was your own work and what was done by a third-party or generative AI, this may reduce your mark or in serious cases constitute academic misconduct.  **Generative AI**: You may use generative AI in writing this section. 2. Video Demonstration You will create a video to demonstrate your application. The video should thoroughly demonstrate the application and its functionality, including demonstrating client-server interactions. This video may optionally include audio or video commentary, annotations or captions for clarity however these are not a formal requirement. If you have addressed some of the extended functionality, the video should make clear what specific functionality you have implemented. If you considered specific legal, safety, or security issues when implementing a feature, you should mention this.  You should expect your video to be around 7-10 minutes long. If the video is shorter than this it is likely it does not demonstrate the project in enough depth.  **Generative AI**: You may use generative AI in creating this section (e.g. for automated transcription), however your video must accurately reflect your Java project. 3. Source code and related project files You will provide a `.zip` file of all source code and other project files needed to build the application. A `README.md` file (Markdown or text format) should be included in the root directory of the project giving necessary instructions and requirements to build and run the project. This may be very brief but will necessarily be longer if the tools or build process are non-standard.  Note that functionality that is only present in the source code and not evidenced in the demonstration video will generally not be considered in marking.  **Generative AI**: You may not include code that has been created by generative AI. You may use generative AI in creating documentation or asset files such as images. 4. Compiled Java files All files required to execute the Java application should be supplied in the form of JAR archives. Instructions for launching these (e.g. any particular commands or command-line options required) must be specified in your `README.md` file. Any additional assets or libraries required to run the applications - if any - must be supplied with instructions as necessary in your `README.md` file. Your Java applications must run on the computers in CC/106.  **Generative AI**: You may use generative AI in creating this section, but it must correspond to the Java source code supplied. Submission There are several parts of this project that must be delivered. These should be submitted as a zip file to the Moodle submission point. If your submission is larger than the Moodle limit for a single file of 400MB, you can submit a split zip. It is possible to split zip archives using the tool 7Zip, which is available through AppsAnywhere (<https://appsanywhere.yorksj.ac.uk/>). Marking A mark will be awarded to your project based on the degree to which the video demonstration of the developed software and the provided source code, satisfies the following:   1. Analyse, evaluate and develop programs written in Java, 2. Apply object-oriented principles to the design of software, 3. Apply common software design patterns to Java, 4. Develop Java relating to core Computer Science subject areas such as Databases and Networks; or games and games technologies 5. Identify and address legal, safety, and security issues when developing software for a given context of use   In your video demonstration you should demonstrate all of the functionality that has been implemented, including additional functionality that you have chosen to implement that is not listed below. This demonstration is of the software in use and does not need to show the source code. Core Functionality The following defines a set of core functionality that is referenced in the marking criteria below. These are written as user stories. Client  1. As a user, I can specify an address and port to connect to or allow it to connect to the default 2. As a user, I can specify a username, and have others see my messages associated with that username 3. As a user, I can connect to the server 4. As a user, I can send a message to other active clients on the server 5. As a user, I can see messages sent by other clients on the server  Server  1. As an administrator, I can specify a port on which to run the server 2. As an administrator, I can start and stop the server. 3. As an administrator, I can access a log file of server activity including all chat messages sent associated with usernames 4. As an administrator, I can access a web page with basic information about the status of server 5. As an administrator, I can access a web page with a list of the current connected users.  Extended Functionality A selection of optional requirements are stated below in the format of feedback from users. These are referred to in Extended Functionality in the marking criteria below. This user feedback may correspond to a number of changes of different complexity. You need to identify what improvements are possible to effectively address the user feedback within the constraints of the project (time, expertise, etc.). You then need to implement these changes.  You are not expected to attempt all of these. Here quality is more important than quantity. It is better to think through and implement fewer of these in depth than address the bare minimum of several. You have flexibility about what functionality to implement; however, the functionality that you implement should be relevant for a chat application. Moreover, pay particular attention to the legal, safety, and security requirements for the application as these are explicitly addressed in the marking critiera.  You should highlight what relevant functionality you have implemented in your demonstration video. Moderation Mary, Administrator: “We need to ensure all of our users are abiding by our terms of use, otherwise it opens us up to potential legal issues, especially if they are using our server for illegal activity. We also occasionally have problems with users posting offensive messages. We need effective moderation tools to enable us and/or our users to take action to address these issues.” File upload Jack, User: “We often want to share files with each other. At the moment we have to upload the files to somewhere like Dropbox and then post the link. It would be much more convenient if we could share files directly over the server; like, I could post the file and then others could choose to download it. If I could post an image and have it appear in the chat that would be amazing. It would be even nicer if this could be as efficient as possible, e.g., by compressing data to send.”  Mary, Administrator: “If we were to host files, we need to ensure that we are not held legally responsible in the UK for their contents. We may need a way to flag inappropriate content or comply with DMCA takedown requests from copyright holders. We also need to ensure the file hosting doesn’t threaten the security or stability of the server.”   User interface Linda, User: “Just using commands in a terminal is hard to use. I would like to have a nicer user interface. I don’t mind if this is a text-based user interface [e.g., using Lanterna], or a graphical user interface [e.g., Swing, JavaFX]. It should make it easy to recognize commands or options that are available, rather than having to recall the command to type in. I want it to be really easy to use and fully-featured. It should be accessible.” Security Improvements Phil, Administrator: “Our obligations under the GDPR (data protection legislation) mean that we need to ensure the security of our client data. In this day and age, sending data unencrypted over the internet is no longer acceptable. We also need to ensure the data is secure when it is on our servers. Of course, our servers also need to be secure against unauthorized access.  We may also be required to provide data under the Investigatory Powers Act in the UK and similar laws in other countries. However, we want to protect the privacy of our users as much as possible without getting on the wrong side of the law.” Interactivity Kyle, User: “Our server wants to add some interactive or fun elements to the chat, such as:   * A simple multiplayer game like as Noughts and Crosses/Tic-Tac-Toe, or the ability to roll dice. * Being able to create and vote in polls and report the results in the chat and/or via the webserver   There would need to be no way to cheat, of course!” Webserver improvements Jane, User: “I want to be able to join the chat when I am not sitting at my own computer. At Uni I can’t install the client software. I would like to be able to connect to a webpage where I can participate as much as possible in the chat.”  John, Administrator: “It would be convenient to be able to control the server through the webpage. While I might be able to see the list of connected users, if I want to reboot the server or kick a client, I would need to SSH into my server account.” Generative AI and Academic Misconduct All code you submit (excluding libraries) must be your own. While you can use resources such as Stack Overflow and generative AI such as ChatGPT to aid your understanding, all code you submit must have been written by you and not copied from a third party. You may make use of third party libraries (e.g. Lanterna, mentioned above) so long as you make clear where you have done so in your Generative AI and Third-Party Content statement |
|  |

| Learning Outcomes |
| --- |
| You must successfully achieve the following Learning Outcomes to pass this assessment: Computer Science 5.2 Analyse, evaluate, and develop organizational computer systems and standards in core Computer Science subject areas such as Databases and Networks, that can be implemented across an organisation to reduce information and systems risk, identify, and mitigate vulnerability, and ensure organisational compliance.  5.3 Recognise any risks or safety aspects that may be involved in the operation of computing and information systems within a given context.  5.4 Demonstrate a critical understanding of the personal, organisational and legal/regulatory context in which computer and information systems could be used, the risks of such use and the constraints that may affect how computer systems are implemented. Games Development 5.2 Recognise any risks or safety aspects that may be involved in the operation of computing and information systems within a given context. Be aware of the growing ethical issues around games and the games industry.  5.3 Apply appropriate theory, practices and tools for the specification, design, development and evaluation of intermediate computing systems including programming in a high-level language; apply this knowledge to the development of games using game technologies.  5.4 Apply the principles, methods, and tools of systems design to develop information systems that meet business needs Software Engineering 5.2 Develop professional and analytical skills and standards in software engineering subject areas such as advanced programming, user experience, software design pattern, for a professional software engineering career;  5.3 Recognise any risks or safety aspects that may be involved in the operation of computing and information systems within a given context;  5.4 Demonstrate a critical understanding of the personal, organisational and legal/regulatory context in which computer and information systems could be used, the risks of such use and the constraints that may affect how computer systems are implemented; |

| Advice and Guidance |
| --- |
| * It is recommended that you make use of Git for version control from the start of your project. Ensure you always have a working version ready to roll back to, especially toward the end of the project. You may wish to use the feature-branch workflow or Git commit tags to make this version easy to find. * When submitting your project, remember to include the hidden .git folder inside your project directory. Without this, the repository information will not be included. |

| How is this assessment marked? |
| --- |
| Your work will be marked according to the assessment instructions provided within this document and the selected Learning Outcomes’ (LOs) (see above).  Furthermore, this assessment is marked using the assessment marking criteria or a similar rubric that aligns with the University’s Generic Assessment Descriptors (see below). This is to ensure all assessment decisions are comparable regardless of the discipline or mode of assessment.  Please note, that you must meet the required baseline standards (40 – 49%) which will include the LOs and minimum expectations of the assessment. Further still, you must ensure you meet the requirements of each grade boundary to progress to the next i.e. you should demonstrate your learning through the standards of a 3rd, 2:2, and 2:1 to achieve a 2:1 (60 – 69%). These standards are designed to scaffold and build your learning to achieve your fullest potential in each criterion being assessed. |

| Assessment Marking Criteria |
| --- |
| **Pass Grade Bands (100 – 40)** (Learning Outcomes must be met)  **Fail Grade Bands (39 – 0)** (Learning Outcomes are not met)   1. Analyse, evaluate and develop programmes written in Java  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **3rd**  (40 - 49) | **2:2**  (50 - 59) | **2:1**  (60 - 69) | **First**  (70 - 84) | **High First**  (85 – 100) | **Borderline Fail**  (30 - 39)  **(Credits may be compensated)** | **Fail**  (0 – 29)  **(Credits may not be compensated)** | | Program has some of the core functionality, or some implementation may be unreliable or error-prone. | Most core functionality has been implemented with minimal errors. | All core functionality has been implemented effectively with no obvious errors; some extended functionality has been implemented | Some extended functionality has been implemented effectively and appropriately, the implementation draws on knowledge from independent study | The requirements for the program have been considered in depth and the implementation is thorough. | While an attempt has been made to implement some functionality, the program is not functional or functionality is very limited | An attempt to implement relevant functionality has not been demonstrated |  1. Apply object-oriented principles to the design of software  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **3rd**  (40 - 49) | **2:2**  (50 - 59) | **2:1**  (60 - 69) | **First**  (70 - 84) | **High First**  (85 – 100) | **Borderline Fail**  (30 - 39)  **(Credits may be compensated)** | **Fail**  (0 – 29)  **(Credits may not be compensated)** | | Object-oriented programming techniques have had limited use | Object-oriented programming techniques have been used throughout | Object-oriented programming techniques have been used effectively and appropriately | Object-oriented programming techniques have been used cleanly, expressively and efficiently, following best practices. Object-oriented design has been used to simplify the problem solution or effectively support code-reuse | Use of object-oriented principles is nearing professional standard. | Program does not use object-oriented programming techniques | There is no evidence of an attempt to design software. |  1. Apply common software design patterns or best practices to Java  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **3rd**  (40 - 49) | **2:2**  (50 - 59) | **2:1**  (60 - 69) | **First**  (70 - 84) | **High First**  (85 – 100) | **Borderline Fail**  (30 - 39)  **(Credits may be compensated)** | **Fail**  (0 – 29)  **(Credits may not be compensated)** | | Source code is reasonably clear; some effort has been made to make it understandable | Source code is well structured and easy to understand | Some software design patterns have been used. | Software design patterns have been used appropriately and effectively. Code is clean, expressive, and well structured. | Advanced software design pattens from independent study have been used. | Source code is supplied, but is counter-intuitive, unclear, or incorrect. | Source code is missing or unreadable |  1. Develop Java relating to core Computer Science subject areas such as Databases and Networks; or games and games technologies  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **3rd**  (40 - 49) | **2:2**  (50 - 59) | **2:1**  (60 - 69) | **First**  (70 - 84) | **High First**  (85 – 100) | **Borderline Fail**  (30 - 39)  **(Credits may be compensated)** | **Fail**  (0 – 29)  **(Credits may not be compensated)** | | Minimal functionality related to core Computer Science subject areas, in particular databases and networks | Functionality related to core Computer Science subject areas, in particular databases and networks has been implemented | Implemented functionally competently applies core Computer Science subject areas, in particular databases and networks | Implemented functionally demonstrates a high level of understanding in effectively applying core Computer Science subject areas, in particular databases and networks; some functionality may relate to multiplayer games or interaction | Implemented functionality integrates knowledge from independent study that goes significantly beyond that taught content. | Implemented functionality is unrelated to core Computer Science subject areas | Minimal functionality has been implemented |  1. Identify and address legal, safety, and security issues when developing software for a given context of use  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **3rd**  (40 - 49) | **2:2**  (50 - 59) | **2:1**  (60 - 69) | **First**  (70 - 84) | **High First**  (85 – 100) | **Borderline Fail**  (30 - 39)  **(Credits may be compensated)** | **Fail**  (0 – 29)  **(Credits may not be compensated)** | | An attempt to address legal, safety and security issues in the design or development of the software has been implemented | A minimal but sufficient implementation of functionality relating to multiple legal, safety and security issues has been made | The implementation of functionality relating to legal, safety and security issues demonstrates consideration beyond the suggestions in the module brief | The software reflects thorough consideration of the context and risks of use, or relevant implemented features reflect thorough understanding from independent study (for example an effective content flagging or moderation system) | The implementation of some of the legal, safety, or security features demonstrates an exceptional level of understanding from independent study (for example effective encryption, two-factor authentication, or automated content moderation using an API) | There may have been some consideration of legal, safety, or security issues, but this is not reflected in the implemented functionality. | There is no evidence that legal, safety or security issues have been considered | |