DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

University of Strathclyde Science

ADVANCED PROGRAMMING SEMESTER 2 COURSEWORK 2025/26 – CRYPTOGRAMS PROJECT

Submission	Submission Details	Deadline	Weighting	Marking and Feedback
User Stories	MyPlace	9/2/2026 **9am**	10%	Will take place in labs week commencing 9/2/26, marks returned shortly thereafter
High Level Design (Class Diagram)	MyPlace	23/2/2026 **9am**	10%	Will take place in labs week commencing 23/2/26, marks returned shortly thereafter
Iteration 1	MyPlace	8/3/2026 **9am**	12%	Will take place in labs week commencing 8/3/26, marks returned shortly thereafter
Iteration 2	MyPlace	22/3/2026 **9am**	12%	Will take place in labs week commencing 22/3/26, marks returned shortly thereafter
Iteration 3 (final implementation)	MyPlace	29/3/2026 **9am**	6%	Will take place in labs week commencing 29/3/26, marks returned shortly thereafter

Table 1

Note that the weightings total 50% of your CS207 class mark, reflecting the weighting for this semester.

Failure to comply with the instructions detailed in this specification will result in a minimum 10% mark reduction penalty for the appropriate submission. Late submissions: as this is continuous assessment for which sample solutions will be released shortly after the first two deadlines, it will not be possible to accommodate late submissions. Where an individual within a team has extenuating circumstances they should contact the lecturer as soon as possible and an appropriate course of action will be determined.

There is a lot of information in this document. Please take the time to read it carefully.

Assessment Overview

The purpose of this assignment is to provide the opportunity to follow the software development lifecycle within a team. This is an important aspect of being a professional in the field as it provides structure and consistency, resulting in projects which are more likely to be successful.

An important challenge here is to explore how the software development lifecycle can be applied in a given context. In the assessment for this semester of Advanced Programming you will be asked to follow a software development lifecycle process in order to develop a Java program to allow users to generate and solve cryptogram puzzles. This will be completed in teams of five which can be chosen by students so long as it is within their allocated lab group. No requests to change lab group can be accommodated as personal circumstances will have already been taken into account.

Background

Encipherment is a process in which a message is encoded in such a way that it is not readable by someone who does not have the mapping from the original message to the encoded message. In particular, in classical cryptography a substitution cipher maps each letter in the alphabet for alternative letters or numbers. Each letter is mapped to a single cipher value. When mapping letters to letters, letters cannot be substituted for themselves.

A cryptogram is a word puzzle where players are presented with an enciphered phrase. The aim of the puzzle is to identify the original message. Players can employ a number of approaches to achieving this. For example, looking for short words such as 1 letter words, which are 'I' or 'A' as these are the only two 1 letter words in English. Players can also use the frequency each enciphered letter appears in the puzzle – e.g. the most frequent letter is probably 'E' for messages in English. There may also be common groups of letters such as 'TH' or 'ION' which may be repeated in the message.

Example Cryptogram

Here is an example where letters are mapped to numbers with a key at the top which you can use to track your progress. One letter has been completed for you.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
																			7						

T _ _ _ _ _ _ _ T _

7 12 13 25 24 4 12 7 24 21 16 10 26 17 10 25 16 22 20 8 8

__ __ T __ __ __ __ __ __

5 22 7 13 26 26 5 26 21

Tasks

The aim is to develop software in Java which allows the user to play cryptogram puzzles. These puzzles should map letters to either letters or numbers. The functionality will be similar to http://cryptograms.org/play.php but not identical. The software should also maintain the state of completion of cryptograms which the user plays, and details about the players including a scoreboard showing player statistics such as number of games successfully completed.

The assignment comprises multiple tasks to be completed in teams of 5. The assessment is designed to be completed by a minimum of 4. Due to class size it may be necessary to have some teams of four. Students will be able to select their own team of five from those in their lab group. The tasks involved are as follows:

Task 1 – Iteration 0 User Stories

The first task is to identify the requirements of the project through creation of user stories and consideration of acceptance criteria. The submission for this stage is the user stories you write. Whilst you are not expected to write and submit all your acceptance criteria, you should spend time considering at least two appropriate acceptance criteria for each user story as you will be asked about this in the marking lab for this submission. Your submission should contain no more than 14 user stories.

To write the user stories, it will be necessary **first** to **elicit** the **requirements** through conversation with the lecturer who will act as the product owner and customer. Requirements elicitation will take the form of questions posed to a Myplace forum. Each team will be permitted to ask one question but will be also be able to see all other teams' questions. The deadline for submitting your question is Thursday 5th February at 1pm.

Note you will be best placed to perform well if you commence thinking about the requirements prior to asking a question, as general requirements can be gathered from the details provided in the background. The answers provided serve to provide further clarification where required. Questions similar to “what are the user stories” will not be answered, but will still count as the question for your team.

Submission of the list of user stories should be to the appropriate slot on MyPlace by the deadline shown in Table 1. The submission should be a single page PDF, and should only be submitted by one member of the team though it is the whole team’s responsibility to ensure that the submission made is agreed upon and on time. The submission slot will be configured as a team submission, meaning all team members will be able to see the submission and hence check it.

This submission will be marked and feedback provided in person in the labs, see the dates in Table 1. In the lab you will be asked to show the user stories you submitted on MyPlace, and asked to suggest acceptance criteria for a subset of user stories. The combination of your responses and the submission will be assessed for your mark in this aspect. See the related marking scheme later in this document.

Task 2– Iteration 0 High Level Design: Class Diagram

This task involves creating a **class diagram** which should represent your initial high-level design of the cryptogram system. This should be created based on the initial user stories which will be released after the requirements stage has been completed.

The class diagram should identify the appropriate classes, associations, and an initial attempt at identifying appropriate attributes and operations. It should not include any user interface classes at this stage. You should also include a 1 paragraph rationale for any specific design decisions made (i.e., the "why" of those decisions). This can be included as a note in the diagram or as additional text beneath the diagram.

Submission should be via the appropriate slot on MyPlace by the deadline shown in Table 1. The submission should be a **single page PDF**, and should only be submitted by one member of the team though it is the whole team's responsibility to ensure that the submission made is agreed upon and on time. The submission slot will be configured as a team submission, meaning all team members will be able to see the submission and hence check it.

The class diagram will be marked and feedback provided in person in the labs, see the date in Table 1.

Task 3–Implementation, Testing, and Final Report Iterations 1-3 (2 weeks each)

After class diagrams have been submitted, a sample solution will be released. This **must be used** as the basis for your **implementation**. All .java files, including JUnit java files, for each iteration should be **submitted to MyPlace in addition** to the commits to the **GitLab** working repository. Submission to MyPlace is to ensure a timestamped copy of code at the point of submission for each iteration.

Code should be **pushed** to the team GitLab repository at a **minimum once a week** from approximately week 6, when implementation iterations will begin. One member of the team should **create the GitLab repository** called 'CS207 Cryptograms Team ? 2026 ' where ? should be replaced with the team number and **share it with the other team members, as well as the lecturer**. Note that the lecturer must be assigned the role of reporter or above to allow them access to the code. This can be achieved through the web interface by accessing a specific repository settings, then selecting 'members' and searching for the person you wish to add and identifying their role. Each team member must contribute to the implementation, and hence be pushing code to the GitLab repository.

Also note that code should not make use of external libraries or frameworks with the exception of JUnit for testing.

Task 3 is split into three iterations. Iteration 1 and 2 will last 2 weeks, while iteration 3 will last 1 week.

The iterations will cover a number of user stories which will be identified from the product backlog by the lecturer (product owner) at the start of each iteration.

Code for iterations 1, 2, and 3 will be marked in the labs in week 8, 10 and 11 by the tutors and lecturer. See Table 1 for dates.

Peer Assessment

The aggregated and weighted coursework mark for the Cryptogram project may be adjusted proportionately to represent individual contributions to the work. This means that you will be provided "interim" marks at each stage, but these will not be collated, adjusted for individual contribution and finalised until the end of the class.

Adjustments based on individual contribution will be established using a peer evaluation form and review of commits to GitLab. If a team member has no commits on GitLab it will be assumed they didn't contribute to the implementation and may receive a result of 0.

Team members are required to complete their own personal assessment of their team mates. This will be established using a peer assessment activity on MyPlace. Each member of the team must complete the peer assessment activity on MyPlace by the final deadline detailed in Table 1.

This involves allocating a score of 1-5 to each team member, including yourself, to reflect the individual's contribution to the project. A score of 1 means there was little or no contribution, a score of 5 means they made a significant contribution to the project. If you do not to submit a peer assessment, then it will be assumed all team members made a significant contribution to the project.

Marking Schemes

User Stories

A maximum of 24 marks is available for the initial user stories and acceptance criteria. Marks will be distributed as follows:

Criteria	Maximum Marks Available
Appropriate identification of appropriate role	1
Appropriate identification of an epic user story	1
Appropriate identification of user stories at a suitable level of granularity using correct structure	14
Acceptance criteria questions: appropriate acceptance criteria are presented in the marking lab for the user stories. They follow the suggested structure, demonstrate it shows the user story has been achieved, and are testable. Teams will be asked to identify four acceptance tests, two tests for two user stories which	8

will be randomly selected by the markers. Thus, you should attempt to create at least two acceptance tests per user story you identify.	

Class Diagram

A maximum of 18 marks is available for the high level design. Marks will be distributed as follows:

Criteria	0 points	1 point	2 points	3 points	4 points
Appropriate identification of classes, with clear, narrow responsibilities	No submission	Classes and properties identified are mostly inappropriate for the cryptogram system. Perhaps multiple classes where the responsibilities aren't clear.	Classes and properties appropriately identified are somewhat appropriate, clear areas for improvement such as properties not representing clear responsibilities and unnecessary classes	Classes and properties appropriately identified are mostly appropriate, with perhaps one class which is unnecessary, or some properties do not represent clear responsibilities	Classes and properties identified are appropriate with clear and narrow responsibilities.
Does the design include all the classes needed to provide all the features	No submission	Most features cannot be achieved using this design	Many features can be achieved using this design. This could be e.g. a model class such as something to manage the players is missing	The features can be achieved using this design, perhaps with only a minor issue such as an attribute which is missing	The features can be clearly achieved using this design
Is the design appropriately cohesive	No submission	Some elements of a cohesive overall design, but mostly minimal cohesion. Many instances of attributes and operations in the wrong class or are missing.	Overall good cohesion in some areas, but instances of attributes and operations in the wrong class or are missing.	Overall a design which is very cohesive, but perhaps one or two instances where cohesion could have been improved.	Overall design is highly cohesive with attributes and associations in a single class related to a single purpose.
Is the design appropriately loosely coupled	No submission	Design is very highly coupled	There is some instances of inappropriate coupling	Inappropriate coupling is kept to a minimum	
Syntax is correct	No submission	Syntax is mostly incorrect, with many clear errors	Syntax is mostly correct, with one or two errors such as incorrect cardinality notation	Syntax is correct	

Implementation

Marked in 3 iterations. At the start of each iteration the product owner will select the user stories from the product backlog which should be implemented in that iteration. Each iteration will be marked according to the following criteria and weighted according to the weighting provided in Table 1:

Criterion	0 points	1 point	2 points	3 points
Functionality (per user story)	No functionality achieved	Some functionality of the user story is achieved, but there are significant areas of deviation from the expected functionality as described by the user stories and acceptance tests sample solution	The user story functionality is achieved mostly as expected, perhaps one or two minor cases where the functionality deviates from expected as described by the user stories and acceptance tests sample solution	The user story functionality is achieved completely as expected as described by the user stories and acceptance tests sample solution
Testing (per user story)	No testing	Some attempt at automated testing, but it is significantly limited compared to the criteria prescribed in the sample solution	Testing is mostly as expected, with perhaps one or two scenarios missed	Testing is as expected with all scenarios and acceptance criteria described in the sample solution explored through automated testing.
Code Quality (per iteration)	No submission	Code quality is poor, demonstrated by long unfocussed methods, large code reuse, and inappropriate naming of variables	Code quality is good, but with definite room for improvement. Perhaps a few overly long or unfocussed methods, or some poor variable name choices such as String s1	Code quality is very good, small focussed methods, reuse is apparent, meaningful variable name choice
User Interface (per iteration)	UI is not at all clear, the user wouldn't be able to figure out the appropriate commands	UI is somewhat clear, with a little time the user could figure out the commands but may make one or two mistakes	UI is very clear, the user can easily figure out what the commands are and how to achieve the functionality with no mistakes	

Supplementary Material

A nice video demonstrating the approaches to solving a cryptogram is shown here https://www.youtube.com/watch?v=5y9THLG94SU

You can see and try some free examples on http://cryptograms.org/play.php though there are many other such websites.

One possible tool for creating your class diagram is VioletUML, available from http://alexdp.free.fr/violetumleditor/page.php . It is free to download, and can be run from the JAR file thus doesn't need to be installed. You are free to use other software should you so desire.