|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **AT2** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR Task 3** | | |
| **Name:** | | **Yoseph Campbell** | | |
| **Student ID:** | | **20085059** | | |
|  | |  | | |
| Assessment type (): | | | | |
|  | Questioning (Oral/Written) | |  | Portfolio |
|  | Practical Demonstration | |  | Project |
|  | 3rd Party Report | |  | Other – Please Specify |
|  |  | |  |  |
| Assessment Resources | | | | |

|  |
| --- |
| The base requirements this assessment task include:   * Python interpreter * IDE or editor for developing Python programs (only PyCharm supported by the college) * Git tools (GitHub tools) for version control * Drawing tool (online) for UML (or use pen, paper) * Access to the internet * Access to Office 365 or Microsoft Word * Access to Blackboard   Use of some of these items may not occur in this part of the assessment task. |
| Assessment Due This assessment is split into components that have several due dates:   |  |  |  | | --- | --- | --- | | **Task** | **Week** | **Date** | | Task 3 | 10 | 17:00 (5pm) on the day of the scheduled lecture |   Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances.  We also will endeavour to update these document(s) at the same time. |
| Instructions Follow the steps listed in this assessment item.  Submission of the documentation, code, and associated items is at the end of each part of the portfolio.  Each part of the portfolio has a deadline for submission.  It is advantageous to you to attempt to meet the deadline provided. |
| Important If you are using a different configuration of tools and equipment for this assessment item, then assistance in this and subsequent parts of the portfolio to ensure the systems work correctly will be limited.  Make sure you put your name and student ID on the front page. Failure to do so will mean your submission will not be marked. |
| Answering Questions When a step includes a question, you must attempt to answer it.  There is a minimum and maximum number of words to use for each answer.  If a step has more than one question, these maxima and minima are a total for all the questions in that specific step.  All answers must be in complete sentences unless indicated.  If required, make sure to add any code you’ve written in a separate file to your submission. DO NOT put code in a Word document. |
| Sources of Information In industry, it is good practice to keep track of where information was obtained. This is especially true if it is a written document, or even code.  If you answer any questions using information from web sites, please include the site name and URL (Web site address) after the answer. Likewise, include the title and author for books and magazine articles. For example:   * RS Electronics Ltd: <https://au.rs-online.com/> * Slack API Documentation, Users List Method: <https://api.slack.com/methods/users.list> |
| Code Storage We advise that you create a GIT repository on GitHub and use this to store a copy of your work.  You may also use OneDrive within your college Office365 to store a backup of your code or keep a copy on a USB thumb drive.  Please note that it is the student’s responsibility to keep backups of their work. This includes any documents and code. |
| Scenario (updated) You have just started working for a boutique Perth-based software development company.  You will be helping within a team that is creating a way to visualise messaging and encryption as a way of teaching teenagers who are interested in coding. For this task, you do not have to worry about the visualisation part, as that will be handled by someone else in your team.  In this task, you will be using the following requirements to update the model which you created in Task 2. Requirements The story of Alice and Bob continues…  Bob has moved to another place. This means that they can no longer drop letters off at each other’s houses, simply because of the distance. They will need to use the Postal Service for mail delivery.  When Alice writes a letter to Bob, she will drop it off at the Post Office. The Post Office holds the letter until the Postie, called Charli, picks it up for delivery. They will put the letter in Bob’s letterbox on their daily round of mail delivery.  After they noticed someone had read one of their letters, Alice and Bob decided to add a level of security to their letters by only sending encrypted messages. By sending encrypted letters, no one can read their messages when in transit. (They can still *try*, though.)  For this assessment, you may assume no other people send letters to each other.  At any point in time, you may ask your lecturer for feedback about your work. In fact, it is encouraged that you do so.  Read all descriptions carefully, as they may look like but are different from those in Task 2.  Please note that you may find it easier to answer the questions and perform the tasks in the steps following in a different order. Execute the tasks in any order you like, but please don’t change the order of the questions/steps in this document. |

|  |  |  |
| --- | --- | --- |
| **STEP** | **Task to perform** | Words Min/Max |
| 00 | Example question This is an example step. It may contain instructions or a question.  If a minimum and maximum word count are provided, you must take those into account. |  |
|  | Your answer or evidence (e.g., screenshots) go in this box.  This box will grow automatically as you put in text and images. |  |
| 01 | Find the new classes Read the story from the scenario carefully.  You should now create Base classes, but only if useful; Base classes should be considered when you have three or more objects that share common traits. This does **not** mean that you must use a Base class in this scenario, in fact, there may be other solutions.  Questions to ask yourself:   * how many people exist in the new scenario? Do these people have the same responsibilities and capabilities? * How does an encrypted letter relate to a “normal” letter? Are they really that different? * When (and why) do you create a new (or a Base) class and when do you just update an existing class? * How does the Post Office (and Charli) know where to send letters?   Answer the following questions:   1. How many new classes can you define? 2. What are those classes? 3. What old classes do you need to update? | 20-50 words |
|  | 1. How many **new** classes can you define?  * Encryptor * Person (Base class) * PenPal * Postie * Post Office  1. What are those classes  * Encryptor is a static class, contains a method to encrypt * PenPal inherits from person base class * Postie inherits from person base class * Person becomes a Parent class, PenPal/Postie are children classes  1. What **old** classes do you need to update?   Will need to update Person class to fit as a general class for PenPal and Postie |  |
| 02 | Refining the Class Diagram Using the classes that you found in the previous step, update the Class Diagram from Task 2. Keep it as simple as possible.  Define the relationships between the classes. Only define a relationship when the classes should know about each other.  Reminder: relationships in UML are associations, aggregations, and compositions, and you must apply aggregation and composition principles where needed.  Put the multiplicities in the Class Diagram too.  You may also provide a textual description if you think that’s relevant, for example, to explain how you got to the solution. | n/a |
|  |  |  |
| 03 | Create a detailed Sequence Diagram As an example, for the Sequence Diagram you may assume Alice is sending an encrypted letter to Bob.  Think about the interactions that take place between the various classes:   * Alice writes a new letter addressed to Bob * She encrypts the content * She drops it off at the Post Office’s mailbox * Charli picks up the letter from the Post * Charli gets the addressee from the letter * Charli delivers the letter at Bob’s by putting it in his letterbox * Bob gets the letter from the letterbox * He decrypts the letter and reads it | n/a |
|  |  |  |
| 04 | Create a detailed Communication Diagram Using the Sequence Diagram from Step 3, create a Communication Diagram.  Remember that there is a very strong relationship between Sequence and Communication Diagrams. | n/a |
|  |  |  |
| 05 | Create a detailed Activity Diagram For this step, you will use the following example to create the Activity Diagram. Activity: Mail Delivery Postie Charli does their daily round of mail delivery. They pick up any letters at the Post Office. They deliver the message at recipient (well, their letterbox).  Your activity diagram must consider the possibility that there are no letters at the Post Office. You may assume that if there *is* mail at the Post Office, it is a single letter. In other words: keep it simple. | n/a |
|  |  |  |
| 06 | Create a detailed State Diagram Based on the new insights you gained from modelling using various UML diagrams, update the State Diagram from the previous Task 2.  Explain what has changed from the original State Diagram in one or two sentences.  If you feel no changes are needed to the State Diagrams from Task 2, at least add a State Diagram for any new state that may have arisen in the system (e.g., in the Post Office). | n/a |
|  |  |  |
| 07 | Update the refined Class Diagram Based on the new insights you gained from modelling using various UML diagrams, you should again update the classes (new and old) you defined in the first step. For example, you may have discovered certain additional methods or attributes that classes need.  Update the Class Diagram so it reflects those changes. In effect, any methods or attributes that can be found in the other diagrams, should be added to the classes in the Class Diagram. | n/a |
|  |  |  |
| 08 | Writing Python code Using the model that you created in the previous steps, write some Python code that reflects that. You should have a couple of classes, each with methods and attributes.  Follow these steps:   1. You should use the existing project and repository from Task 2. 2. Create a new branch to work in. 3. Add the necessary files (one class per file) and write the code. You may write a few unit tests to test that your code makes sense, or an example (like the activity from Step 5) to show that your code works. 4. *You do not have to code an* *encryption algorithm,* *but you must somehow indicate encryption takes place. However, if you can come up with a very simple encryption algorithm, feel free to implement it.* 5. Add and commit the files and push to your remote repository. 6. Create a new pull request (PR) for this branch. You can keep making changes in the branch, pushing to the remote repository, until you are done. The pull request will simply keep track of the changes. 7. After you are satisfied your code is finished, merge the pull request into your main branch. 8. Update your local repository so it is synchronised with the remote repository on GitHub.   Your code must reflect the model that is described with the UML diagrams.  Any boilerplate code (e.g., in a **main()** function) or unit tests do not need to be added to the UML model. Just focus on the core classes.  Add the project (as a zip file) to your submission in Blackboard. | n/a |
|  |  |  |
|  | Submission of Portfolio Work To submit the portfolio, do the following:   * Ensure you have put your name and student ID on the front page of this document. Your submission will not be accepted if name or student ID is missing. * Save the document with your answers as a MS Word file (.docx). * Do NOT zip your Word document! * Create a zip file with your Python project (please remove the virtual environment and project folder from the zip file before submitting). * Open Blackboard, and locate the AT2 Portfolio Task 2 assessment * Open the assessment and upload the original document and the zip file as separate files in one submission. * Click submit.   Whilst there is no need to use any other word processing software as you have access to Office 365 for free as a student, if you use Apple Pages, or Open Office, we will then require you to upload the original file **AND** a PDF version. |  |

# Appendix A: Code Style Guidelines

Your code will follow the PEP 8 standard.

Readability Counts  
- Zen of Python

Explicit is better than implicit.  
- Zen of Python

Other code standards available in the Presentation, “Python Coding Standards for North Metropolitan TAFE”.