|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **AT2** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR Task 2** | | |
| **Name:** | | **Jessica Turner** | | |
| **Student ID:** | | **20088290** | | |
|  | |  | | |
| 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 2 | 7 | 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 You have just started working for a boutique Perth-based software development company.  You will be working in a team that is creating a way to visualise messaging and encryption as a way of teaching young 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 requirements to create a simple model of the story provided below. The model includes Class Diagrams, Sequence and Communication Diagrams, and State and Activity Diagrams. You will write a little bit of Python as well to test your model and to learn about the relationship between the model and the code. Requirements The story goes like this.  Alice and Bob live in the same neighbourhood in a tiny town. They like to write each other old-fashioned letters. For example, when Alice writes a letter to Bob, she puts it in Bob’s letterbox, and vice versa.  The letterbox has a little indicator (a flag) that shows whether it contains a letter.  Alice and Bob both check their letterboxes at most once a day and only one letter is in transit at any given moment. For example, after Alice sent a letter to Bob, she would not send another one until she’s had a response from Bob.  A letter will mark itself as “read” whenever someone reads the message in the letter. This is a hidden state. Alice and Bob use this to check whether no one else has read their messages.  Your task is to model this behaviour in UML diagrams. You can create the diagrams in one of the many online tools that exist. You may also use pen and paper and take a photo if that is easier. Make sure the diagrams are clearly readable.  Step 8 requires you to write some Python code based on the models. If you want, you can write the Python code as you go along. This will help you to learn the relationship between the UML model and the code. Make sure to create a repository to put your code in (a requirement for Step 8).  At any point in time, you may ask for feedback about your work. In fact, it is encouraged that you do so. |

|  |  |  |
| --- | --- | --- |
| **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. | n/a |
|  | 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 classes After reading the story from the scenario carefully, answer the following questions.   1. How many classes can you define?  Hint: look at the nouns. A noun is a word that describes a thing, for example, a bike; ‘bike’ is a noun in this context. A person (by any name) could be a class too. 2. What are those classes?  Note: you do not have to think about creating super classes or base classes at this stage. | 20-50 words |
|  | 1. I can define four classes. 2. The classes are:    1. Alice    2. Bob    3. Letterbox    4. Letter |  |
| 02 | Create the Class Diagram Using the classes that you found in the previous step, create an initial Class Diagram. 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.)  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 Sequence Diagram To create this Sequence Diagram, you may assume Bob is sending a letter to Alice, which she then reads.  Think about the interactions that take place between the various classes when Bob writes a letter and sends it (or rather: drops it off) at Alice’s. What happens next when she reads the letter?  Again, keep it simple. | n/a |
|  | Diagram  Description automatically generated |  |
| 04 | Create a Communication Diagram Using the Sequence Diagram from Step 4, create a Communication Diagram.  Remember that there is a very strong relationship between Sequence and Communication Diagrams. | n/a |
|  |  |  |
| 05 | Create a State Diagram If you read the scenario and the story carefully, you will have noticed that the letterbox has state. If it contains a letter, the indicator (a little flag) is put up and if the letter is removed, the indicator is put down.  Letters also have state, since a letter updates a flag that indicates when it has been read (think of it as an old-fashioned wax seal).  You can capture state changes in a State Diagram.  Pick one of the following options to create a State Diagram for:   1. Create a State Diagram that shows how the letterbox’s indicator, or flag, changes its position (“state”) based on the presence of a letter.   Or: 2. Create a State Diagram that shows how the message’s “read” flag is changed based on actions by the actors (Alice, Bob, etc.).   If you make any assumptions about the Class or the State changes, please describe them as well. | n/a |
|  |  |  |
| 06 | Create an Activity Diagram For this step, you will use the following simple example for which to create the Activity Diagram. Activity: Write and Deliver a Letter Alice writes a letter and delivers it in Bob’s letterbox. | n/a |
|  |  |  |
| 07 | Update the Class Diagram Based on the new insights you gained from modelling using various UML diagrams, you should update the classes you defined in the first step. For example, you may have discovered certain 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 the 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. Create a new project, for example, in PyCharm, and give it a descriptive name. 2. Add the necessary files (one class per file) and write the code. 3. You must write a few unit tests to test that your code makes sense, or an example (like the activity from Step 6) to show that your code works. 4. Create a repository on GitHub in your personal account and link it to your local project. (You may change the order of the steps if you feel that makes life a bit easier.) 5. Add and commit the files and push to your remote repository. 6. Communicate any down time, service interruptions and installation of service (VCS) to your lecturer. 7. Inform your lecturer of the final outcomes, e.g., through email. Make sure to record that communication (e.g., provide a screenshot of the email).   Your code must reflect the model that is described with the UML diagrams.  However, if you write unit tests or a **main()** method to run an example, you do not have to add those to the UML diagram. 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”.