## **ACIT 2515 – Object Oriented Programming – Assignment 2**

**You must complete this assignment in a team.**

**Goals**

* To implement the code that supports your UML design.
* To exercise the four pillars of object oriented programming (OOP) in your design: abstraction, encapsulation, inheritance and polymorphism.
* To apply best practices in Python development.

**Implementation (16 marks)**

Implement the four classes in Python as per your UML design: AbstractEntity, Entity1, Entity2, EntityManager. Make sure you follow all the OOP and Python best practices we covered in the course so far. This includes DocString, naming, constants and method parameter validation.

The manager class must implement:

* add – Takes in an entity object, assigns it a unique ID and adds it to the list of entities.
* get – Takes in an ID and returns that entity object from the list of entities, if it exists. Returns None if it does not exist.
* get\_all – Returns a list of all entity objects in the list of entities. Returns an empty list if there are none.
* get\_all\_by\_type – Takes in a type and returns a list of all entity objects in the list of entities of the given type. Returns an empty list if there are none.
* delete – Takes in an ID and removes that entity from the list of entities. Raises an exception if an entity with the given ID does not exist in the list of entities.

**Unit Testing**

Create unit tests for each of the EntityManager, SpecificEntity1 and SpecificEntity2 classes. You do not need a unit test for the AbstractEntity class.

For the EntityManager unit tests, make sure you exercise each method with both types of entities.

Make sure to follow all the best practices for unit testing. Include test fixtures where needed. There should be test coverage for all the public methods in your classes (including those inherited from the abstract parent class).

**Analysis**

Answer the following two questions:

* How does your design implement the four pillars of OOP (abstraction, encapsulation, inheritance and composition, and polymorphism)?
* Why are your classes good abstractions (i.e., models) of the real-world entities they represent?

**Grading Summary**

|  |  |
| --- | --- |
| Implementation   * Entity Manager (8 marks) * Abstract Entity (4 marks) * Specific Entity 1 and 2 (3 marks) | 15 marks |
| Unit Tests   * Entity Manager (2 marks) * Specific Entity 1 and 2 (3 marks) | 5 marks |
| OOP Analysis | 5 marks |
| **Total** | **25 marks** |

Marks will be subtracted for violations of the best practices covered so far in the course (including naming conventions, reST DocString, constants for magic numbers, venv folder in submissions, etc).

**Submission**

Push the following files to Git:

* Entity Manager Class (i.e., employee\_manager.py)
* Abstract Entity Class (i.e., abstract\_employee.py)
* Specific Entity 1 Class (i.e., manager.py)
* Specific Entity 2 Class (i.e., engineer.py)
* Your test folder with the unit tests:
  + Entity Manager Unit Test (i.e., test\_employee\_manager.py)
  + Specific Entity 1 Unit Test (i.e., test\_manager.py)
  + Specific Entity 2 Unit Test (i.e., test\_engineer.py)
* The answers to the OOP questions in PDF format

Then submit your Git URL and submission tag to D2L. One submission per group.