ACIT 2515 – Object Oriented Programming – Assignment 1

Instructor	Mike Mulder (mmulder10@bcit.ca)
Total Marks	30
Due Dates	Friday, Feb. 15, 2019 by midnight

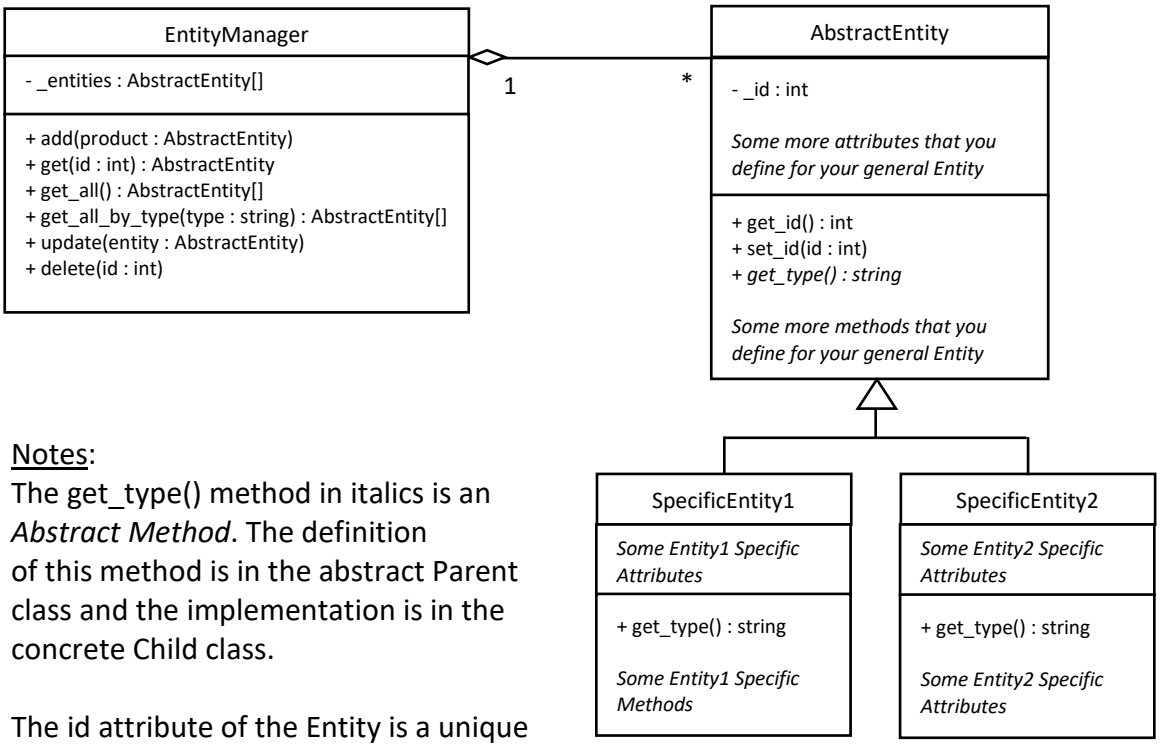
You must do this assignment (and subsequent assignments) with a partner.

Goals

- To exercise the four pillars of object oriented programming (OOP): abstraction, encapsulation, inheritance and polymorphism.
- To setup the core objects for a CRUD (create, read, update, delete) full-stack application that you will be building out (extending and refactoring) in Assignments 2 and 3.

Overview

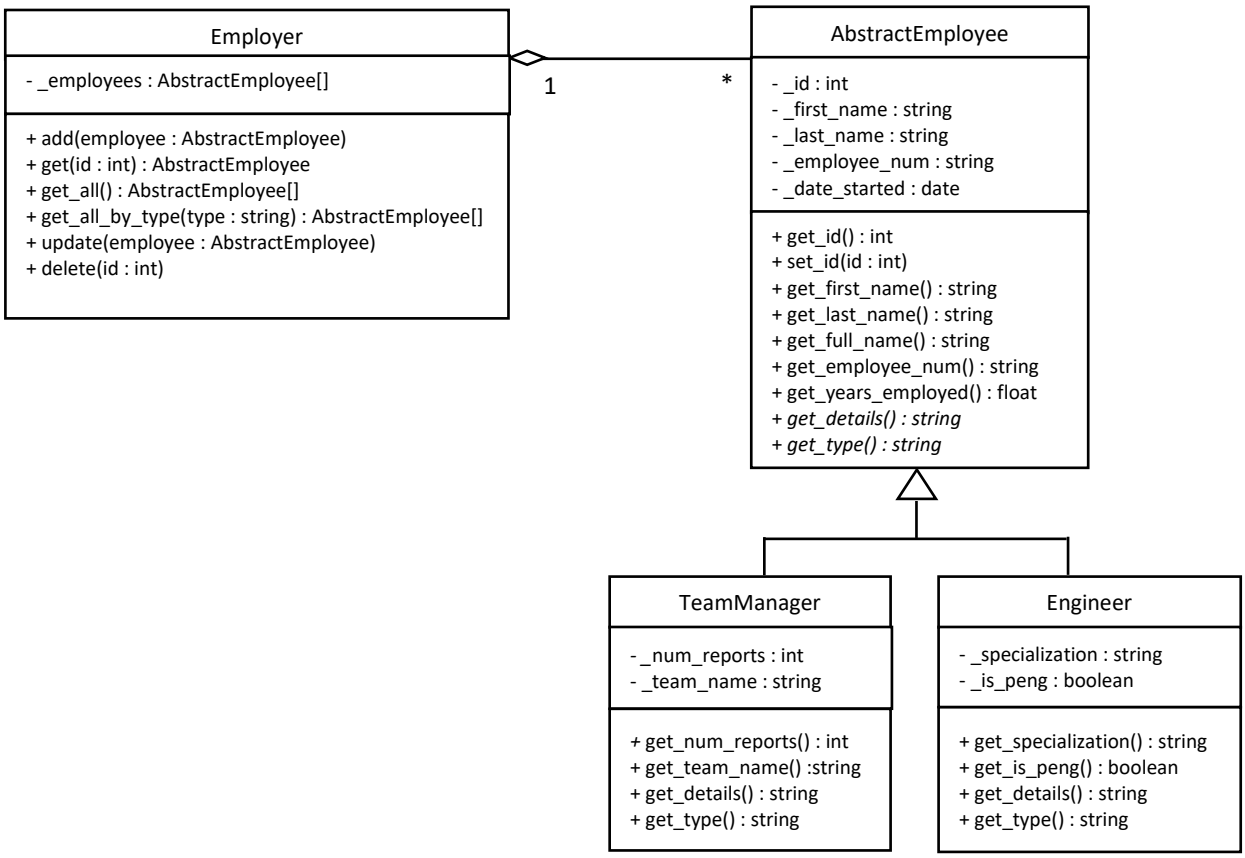
This is the general pattern for the design you will be implementing for this assignment. You need to choose your "Entity" and update the design below.



Notes:
The get_type() method in italics is an *Abstract Method*. The definition of this method is in the abstract Parent class and the implementation is in the concrete Child class.

The id attribute of the Entity is a unique integer assigned to instance when it is created by the EntityManager.

Here is a representative example of what your design might look like (**cannot use this for the assignment – choose your own "Entity"**).



Design (4 marks)

Choose the "Entity" for your assignment and create the UML diagram (as above) with the specific details for your entity.

- EntityManager – Update to reflect the name of your entity manager and entity. It should have the same attributes and methods (with naming updates as appropriate, i.e., AbstractEntity to AbstractEmployee).
- AbstractEntity – Update with the name of your entity and add the attributes and methods for this abstract entity. It should have at least 4 attributes plus those for id and type. It must have get_id/set_id methods, an abstract get_type method, getters for each attribute and at least one additional abstract method
- SpecificEntity1 – Update with the name of your specific entity and add the specific attributes and methods for this type of entity. It should have at least 2 attributes specific to this entity (with getter methods). It must implement the abstract methods from the parent class.
- SpecificEntity2 - Update with the name of your specific entity and add the specific attributes and methods for this type of entity. It should have at least 2 attributes specific to this entity (with getter methods). It must implement the abstract methods from the parent class.

Choose class, attribute and method names that follow Python naming conventions.

Make sure the design of your "Entity" classes are good abstractions of the real-world entities you are modeling. *If you want early feedback, you may have your instructor review your design via Slack, e-mail or during office hours.*

Implementation (16 marks)

Implement the four classes in Python as per your UML design. Make sure you follow all the OOP and Python best practices we covered in the lectures and labs over the first 5 weeks of the course. This includes DocString, naming, constants and method parameter validation.

Some details of the methods in the manager class:

- 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.

- Update – Takes in an entity object and replaces the existing entity in the list of entities based on the ID. Raises an exception if an entity with the same ID does not exist in the list of entities.
- 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 (6 marks)

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 we covered in the lectures and labs over the first 5 weeks. Include test fixtures (setUp and tearDown) and logPoint in each of your unit test classes. There must be test coverage for all the public methods in each of your classes (including those inherited from the abstract parent class).

Analysis (4 marks)

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 entity classes good abstractions (i.e., models) of the real-world entities?

Grading Summary

UML Design	4 marks
Implementation • Entity Manager (8 marks) • Abstract Entity (4 marks) • Specific Entity 1 (2 marks) • Specific Entity 2 (2 marks)	16 marks
Unit Tests • Entity Manager (2 marks) • Specific Entity 1 (2 marks) • Specific Entity 2 (2 marks)	6 marks
OOP Analysis	4 marks
Total	**30 marks**

Marks will be subtracted for violations of best practices covered so far in this course (i.e., naming, DocString, constants for magic numbers, etc).

Submission

Submit the following in a zipfile called **Assignment1.zip** and upload to the dropbox in D2L (Activities -> Assignments -> Assignment 1):

- UML Design in PDF format (**uml_design.pdf**)
- Entity Manager Class (i.e., employee_manager.py)
- Entity Manager Unit Test (i.e., test_employee_manager.py)
- Abstract Entity Class (i.e., abstract_employee.py)
- Specific Entity 1 Class (i.e., manager.py)
- Specific Entity 1 Unit Test (i.e., test_manager.py)
- Specific Entity 2 Class (i.e., engineer.py)
- Specific Entity 2 Unit Test (i.e., test_engineer.py)
- OOP Analysis Answers in PDF Format (**oop_analysis.pdf**)

The submission is due Friday, Feb. 15 at midnight for all sets. No late submissions will be accepted.