UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 9

Objects and Object Oriented Programming

Date Due: August 2, 2019, 11pm **Total Marks: 65**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- **Assignments are being checked for plagiarism.** We are using state-of-the-art software to compare every pair of student submissions.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M.
- Make sure your name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you.
- Do not submit folders, or zip files, even if you think it will help.
- Programs must be written in Python 3.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Read the purpose of each question. Read the Evaluation section of each question.

Question 0 (5 points):

Purpose: To force the use of Version Control in Assignment 9

Degree of Difficulty: Easy

You are expected to practice using Version Control for Assignment 9. This is a tool that you need to be required to use, even when you don't need it, so that when you do need it, you are already familiar with it. Do the following steps.

1. Create a new PyCharm project for Assignment 9.
2. Use `Enable Version Control Integration...` to initialize Git for your project.
3. Download the Python and text files provided for you with the Assignment, and add them to your project.
4. Before you do any coding or start any other questions, make an initial commit.
5. As you work on each question, use Version Control frequently at various times when you have implemented an initial design, fixed a bug, completed a question, or want to try something different. Make your most professional attempt to use the software appropriately.
6. When you are finished your assignment, open the terminal in your Assignment 9 project folder, and enter the command: `git --no-pager log` (double dash before the word 'no'). The easiest way to do this is to use PyCharm, locate PyCharm's `Terminal` panel at the bottom of the PyCharm window, and type your command-line work there.

   Note: You might have trouble with this if you are using Windows. Hopefully you are using the department's network filesystem to store your files. If so, you can log into a non-Windows computer (Linux or Mac) and do this. Just open a command-line, `cd` to your A6 folder, and run `git --no-pager log` there. If you did all your work in this folder, git will be able to see it even if you did your work on Windows. Git's information is out of sight, but in your folder.

   Note: If you are working at home on Windows, Google for how to make git available on your command-line window. You basically have to tell the command-line app where the git app is.

You may need to work in the lab for this; Git is installed there.

What to Hand In

After completing and submitting your work, open a command-line window in your Assignment 9 project folder. Run the following command in the terminal: `git --no-pager log` (double dash before the word 'no'). Git will output the full contents of your interactions with Git in the console. Copy/paste this into a text file named `a9-git.log`.

If you are working on several different computers, you may copy/paste output from all of them, and submit them as a single file. It's not the way to use git, but it is the way students work on assignments.

Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

Evaluation

- 5 marks: The log file shows that you used Git as part of your work for Assignment 9. For full marks, your log file contains
  - Meaningful commit messages.
  - At least three commits per question for a minimum number of 12 commits

Question 1 (25 points):

Purpose: To adapt existing ADTs into the object-oriented paradigm.

Degree of Difficulty: Moderate

In class, we discussed how many of the ADTs we previously implemented could be equivalently represented using object-oriented programming. For this question you are going to implement object-oriented versions of the node-based Stack and Queue ADTs (provided). You have been provided the following files on moodle:

- `NStack.py`: A node-based Stack ADT.
- `NQueue.py`: A node-based Queue ADT.
- `Node.py`: A object-oriented node
- `test_stack.py`: A test script that can be used for testing your object-oriented Stack implementation.
- `test_queue.py`: A test script that can be used for testing your object-oriented Queue implementation.

To succeed in this question you will need to complete the following tasks:

(a) (10 points) Transcribe the node-based Stack ADT (`NStack.py`) into OO-style. Follow the object-oriented guidelines described in Chapters 20 and 21 of the textbook. You are expected to write proper doc-strings (interfaces) for each method. Use the provided test script to check the correctness of your solution. Put your solution in a file called `a9q1_Stack.py`.

(b) (10 points) Transcribe the node-based Queue ADT (`NQueue.py`) into OO-style. Follow the object-oriented guidelines described in Chapters 20 and 21 of the textbook. You are expected to write proper doc-strings (interfaces) for each method. Use the provided test script to check the correctness of your solution. Put your solution in a file called `a9q1_Queue.py`.

(c) (5 points) In a file call `a9q1_reflection.txt`, Answer the following questions about your experience implementing these classes. You may use point form, and informal language. Just comment on your perceptions; you do not have to give really deep answers. Be brief. These are not deep questions; a couple of sentences or so ought to do it.

1. List the similarities between the two classes in terms of attributes and methods.
2. List the differences between the two classes in terms of attributes and methods.
3. Briefly discuss some of the difficulties and/or errors you encountered when implementing and debugging your classes. Did the similarity between the two classes help or hinder?

What to Hand In

- A file titled `a9q1_Stack.py` with your implemented OO-Stack class.
- A file titled `a9q1_Queue.py` with your implemented OO-Queue class.
- A file titled `a9q1_reflection.txt`.

Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

Evaluation

- Stack Class
  - 8 marks for implementation
  - 2 marks for documentation

- Queue Class
  - 8 marks for implementation
  - 2 marks for documentation
- Reflection
  - 2 marks for correctly identifying similar attributes and methods
  - 2 marks for correctly identifying the distinct attributes and methods
  - 1 mark for a discussion on encountered difficulties

Question 2 (20 points):

Purpose: To practise inheritance.

Degree of Difficulty: Moderate

For this question, rewrite the object-oriented Stack and Queue code from the previous question using inheritance to encapsulate common attributes and methods. You are expected to write proper doc-strings (interfaces) for each method.

To succeed in this question you will need to complete the following tasks:

(a) (10 points) Implement a super-class called `Container`. This class should contain all the attributes and methods common to the object-oriented Stack and Queue classes you identified in the previous question. Put your solution in a file called `a9q2_Container.py`. Some of your original attributes and methods may have to be changed to make them more generalizable.

(b) (5 points) Rewrite the object-oriented Stack class so that it inherits all the Container attributes and methods. Put your solution in a file called `a9q2_Stack.py`. You can use the test script provided in question 1 to make sure your new Stack class works correctly.

(c) (5 points) Rewrite the object-oriented Queue class so that it inherits all the Container attributes and methods. Put your solution in a file called `a9q2_Queue.py`. You can use the test script provided in question 1 to make sure your new Queue class works correctly.

What to Hand In

- A file titled `a9q2_Container.py`
- A file titled `a9q2_Stack.py`
- A file titled `a9q2_Queue.py`.

Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

Evaluation

- Container Class
  - 7 marks for implementation
  - 3 marks for documentation
- Stack Class
  - 1 marks for correctly setting up inheritance
  - 3 marks for `__init__()`
  - 1 mark for documentation
- Queue Class
  - 1 marks for correctly setting up inheritance
  - 3 marks for `__init__()`
  - 1 mark for documentation

Question 3 (15 points):

Purpose: To do a bit of Object Oriented Programming.

Degree of Difficulty: Moderate. The only problem is the time it will take to study and find your way around the given code base.

In this question we will be working with 2 object-oriented classes, and objects instantiated from them, to build a tiny mock-up of an application that a teacher might use to store information about student grades.

Obtain the file `a9q3.py` from Moodle, and read it carefully. It defines two classes:

- `GradeItem`: A grade item is anything a course uses in a grading scheme, like a test or an assignment. It has a score, which is assessed by an instructor, and a maximum value, set by the instructor, and a weight, which defines how much the item counts towards a final grade.
- `StudentRecord`: A record of a student's identity, and includes the student's grade items.

At the end of the file is a script that reads a text-file, and displays some information to the console. Right now, since the program is incomplete, the script gives very low course grades for the students. That's because the assignment grades and final exam grades are not being used in the calculations.

Complete each of the following tasks:

1. Add an attribute called `final_exam`. Its initial value should be `None`.
2. Add an attribute called `assignments`. Its initial value should be an empty list.
3. Change the method `StudentRecord.display()` so that it displays all the grade items, including the assignments and the final exam, which you added.
4. Change the method `StudentRecord.calculate()` so that it includes all the grade items, including the assignments and the final exam, which you added.
5. Add a method called `drop_lowest(grades)` to `StudentRecord` that will search through the given list of grade items, and sets its the weight to zero (not the score). The argument to this method, `grades`, is any list of grade items, but should work for either the studnet's lab marks or the student's assignment marks. The script at the end of the `a9q3.py` shows how it is applied to the students lab marks. Right now, it does nothing!
6. Add some Python code to the end of the script to calculate a class average.

Additional information

The text-file `students.txt` is also given on Moodle. It has a very specific order for the data. The first two lines of the file are information about the course. The first line indicates what each grade item is out of (the maximum possible score), and the second line indicates the weights of each grade item (according to a grading scheme). The weights should sum to 100. After the first two lines, there are a number of lines, one for each student in the class. Each line shows the student's record during the course: 10 lab marks, 10 assignment marks, followed by the midterm mark and the final exam mark. Note that the marks on a student line are scores out of the maximum possible for that item; they are not percentages!

The function `read_student_record_file(filename)` reads `students.txt` and creates a list of `StudentRecords`. You will have to add a few lines to this function to get the data into the student records. You will not have to modify the part of the code that reads the file.

List of files on Moodle for this question

- `a9q3.py` — partially completed
- `students.txt`

What to Hand In

- The completed file `a9q3.py`.

Be sure to include your name, NSID, student number, course number and laboratory section at the top of the file.

Evaluation

- 1 mark: You correctly added the attribute `final_exam` to `StudentRecord`
- 1 mark: You correctly added the attribute `assignments` to `StudentRecord`
- 3 marks: You correctly modified `StudentRecord.display()` correctly to display the new grade items.
- 3 marks: You correctly modified `StudentRecord.calculate()` correctly to calculate the course grade using the new grade items.
- 5 marks: You correctly added a method called `drop_lowest()` to `StudentRecord` class. Your function is correct, and efficient, and demonstrates good design.
- 2 marks: You added some code to the script that calculates the class average.