Practical-Exam-05: Hogwarts Magic Queue

Due 25 Oct by 12:00 **Points** 100 **Available** 25 Oct at 10:15 - 25 Oct at 12:00 about 2 hours

This assignment was locked 25 Oct at 12:00.

Introduction

This session ends 12 pm

Make sure to complete your submission in time.

Submit to SVN & Websubmission (for every problem solved)

Submission

- Create the repository on the SVN server
- You must checkout only the folder practical-exam-05 from your server
- No other folders from your SVN will be allowed during the practical exam.
- Check if your repository is being track by WebSubmission before start solving the exam.

```
svn mkdir -m "first assessment commit" --parents https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-11/practical-exam-05

```

Assessment

- **This is a practical exam - your work must be entirely your own.**
- Marks for this practical exam contribute 2 marks.
- Marks will be awarded later by your workshop supervisor (40%) and websubmission marker (60%).
- **Due date: the end of this section (12 pm).**
- **Do Not Forget** To Submit on **WebSubmission**
- **Late penalties:** Only work submitted during your enrolled practical session from a Linux system in the practical lab will be marked.

Regarding functional marks, please consider:

```
(1) only codes that can compile will be marked;
(2) only codes that are in the suggested directory will be marked;
(3) only codes submitted to SVN and WebSubmission before the deadline will be marked;
(4) only codes containing your signature on the top of the file will be marked by tutors;
(5) you will have your markers decreased in 3 points if *.class file present in your folders;
```

Note

- To acquire full marks **(1)** all your functionalities must be working perfectly, **(2)** your code has to be well and proportionally commented, and **(3)** your code must follow correct indentation (4 spaces for every new indentation level) and **(4)** you have to use all the content from latest lectures.
- We argue that you are not just asked to solve a problem but use the more sophisticated way to solve it. For instance, you can solve a problem using ten variables, but it will always be better to solve the same problem with an array.

```
/*==================================
Foundations of Computer Science
Student: you name
id: your id
Semester:
Year:
Practical Exam Number:
==================================*/
```

Practical Exam 05: Hogwarts Magic Queue

Harry Potter is a series of fantasy novels written by British author J. K. Rowling. The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry.

As you may know, the fees for Hogwarts School are magically expensive. Worrying about their students future, the school of Hogwarts has a system that provides a magical scholarship for their students. However, they lack a computer system that organizes the queue of students that need the scholarship.

In this Practical Exam, you are required to implement Hogwarts Scholarship Queue system. This Queue requires 3 components (basic class student, node class, and Queue class). The development of this system will happen in (3) stages. They are:

Problem (1) implementing the basic class for a student;
Problem (2) implementing basic node class containing objects from the class student as information into the node; and,
Problem (3) implementing basic Queue structure using all the aforementioned components;

Problem 01: Basic Class for Student (30 % - functional marks)

```
'''

Signatures:

On this practical exam, you are not asked to design any signature other than accessors and mutators;

class Student
properties: name (String), age (int), period (int)
(1) Basic Constructor: Student()-> set everything to 0 or "unknown";
(2) Constructor with all parameters: Student(String tmpName, int tmpAge, int tmpPeriod);

Requirements:
1. Set accessors and mutator for all properties;
2. Set Basic and Special Constructor
3. Comment file to acquire Code Style Marks
```

Problem 02: Basic Class Node (20 % - functional marks)

'''

Signatures:

class **Node**
properties: next (Node), info (Student)
(1) Basic constructor: Node()
(2) Constructor with parameter for student: Node(Student tmpStudent)

Requirements:

1. Set accessors and mutator for all properties;
2. Set Basic and Special Constructor
3. Comment file to acquire **Code Style Marks**

Problem 03: Basic Class Queue (50% - functional marks)

'''

Signatures:

class Queue
properties: back - the back of the queue is where the last element is enqueue;

methods: enqueue(Student tmpStudent), dequeue(), displayQueue()

The method **displayQueue** consists of traversal and print. The output format should follow:

#1 Cedric Diggory, 14 years old, 1st year in Hogwarts;
#2 Harry Potter, 14 years old, 2nd year in Hogwarts;
#3 Draco Malfoy, 14 years old, 3rd year in Hogwarts;

Requirements:

1. Make methods described above;
2. Make accessors and mutator for all properties;
3. Make constructor
4. Must compile and work properly;

'''

Help and Test Cases

Actions:

1. Set up Queue:

```
#1 Cedric Diggory, 14 years old, 1st year in Hogwarts;
#2 Harry Potter, 14 years old, 1st year in Hogwarts;
#3 Hermione Granger, 12 years old, 1st year in Hogwarts;
#4 Ron Weasley, 15 years old, 1st year in Hogwarts;
#5 Fred Weasley, 16 years old, 3rd year in Hogwarts;
#6 George Weasley, 16 years old, 3rd year in Hogwarts;
#7 Draco Malfoy, 15 years old, 2nd year in Hogwarts;
```

2. Set up the behaviors:

```
1. Recently, Cedric Diggory died in a battle. He no longer needs a scholarship. He can be dequeued;
2. Mr. Potter was granted a scholarship. He can be dequeued;
3. Recently, Mr. Malfoy had family financial issues. He needs to be added to queue;
```

3. Display Queue

```
   howgarts.displayQueue(): this method should display

   Student Name, Age years old, Period year in Hogwards;
```

```
public class Test{

    public static void main(String [] args){
        Queue howgarts = new Queue();

        howgarts.enqueue(new Student("Cedric Diggory", 14, 1));
        howgarts.enqueue(new Student("Harry Potter", 14, 1));
        howgarts.enqueue(new Student("Hermione Granger", 12,1));
        howgarts.enqueue(new Student("Ron Weasley", 15, 1));
        howgarts.enqueue(new Student("Fred Weasley", 16, 3));
        howgarts.enqueue(new Student("George Weasley", 16, 3));

        howgarts.dequeue();
        howgarts.dequeue();

        howgarts.enqueue(new Student("Draco Malfoy", 15, 2));

        howgarts.displayQueue();

    }
}
```

Practical Exam 06

Criteria	Ratings				Pts
Code Style	**30.0 Pts** **Excellent** Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read, (5) lines no more than 80 characters including comments.	**18.0 Pts** **Good** Your code (1) has useful comments and place line and block comments correctly, (2) follow indentation, (3) has good variable naming.	**0.0 Pts** **No marks** You code (1) don't have comments.		30.0 pts
Basic class for student	**21.0 Pts** **Excellent** Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem, (4) your code implements accessors and mutators;	**17.0 Pts** **Good** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem, , (4) your code has the implementation of accessors and mutators.	**11.0 Pts** **Fair** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem.	**0.0 Pts** **No marks** Your code does not compile	21.0 pts
Basic class for Node	**14.0 Pts** **Excellent** Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem, (4) your code implements accessors and mutators;	**11.0 Pts** **Good** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem, , (4) your code has the implementation of accessors and mutators.	**7.0 Pts** **Fair** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem.	**0.0 Pts** **No marks** Your code does not compile	14.0 pts
Basic class for Queue	**35.0 Pts** **Excellent** Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem, (4) your code implements accessors and mutators;	**28.0 Pts** **Good** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem, , (4) your code has the implementation of accessors and mutators.	**18.0 Pts** **Fair** Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem.	**0.0 Pts** **No marks** Your code does not compile	35.0 pts
					Total points: 100.0