# Practical-Exam-05: Hogwarts Magic Queue

**Due** 3 Jun by 16:55 **Points** 100 **Available** 3 Jun at 15:00 - 3 Jun at 17:00 2 hours

This assignment was locked 3 Jun at 17:00.

## Introduction

**This session ends at 16:55 (online session)**

**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>/2022/s1/fcs/week-12/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 (30%) and websubmission marker (70%).
- **Due date: 5 minutes before the end of your session (16:55 pm).**
- **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 folder
   s;
```

## 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)** complete the basic class for a student;  A draft file is provided **here (https://myuni.adelaide.edu.au/courses/74996/files/11019311?wrap=1)** **(https://myuni.adelaide.edu.au/courses/74996/files/11019311/download?download_frd=1)**
**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;

These are all to be implemented in a sub-directory called **hogwarts**

## Problem 01: Basic Class for Student (20 % - functional marks)

### Signatures:

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);
(3) Write a method  printStudent()  to print its values.

### Requirements:

1. Set Basic Constructor
2. Set Special Constructor
3. Write printStudent().

The format of the output for a given student is shown below

```
Printing student record
Name:   Harry Potter
Age:    14
Period: 2
```

## Problem 02: Basic Class Node (15 % - 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 (25% - functional marks)

## Problem 03: Basic Class Queue (65% - functional marks)

```
Signatures:

class Queue
properties:
  front - the front of the queue is where the first Node was enqueued;
  back - the back of the queue is where the last Node is enqueued;

methods:
  void enqueue(Student tmpStudent)     // add student to back of queue
  Student dequeue()                      // remove student at front of queue
  String  peek()                        // return name of student at front of queue
  void displayQueue()                  // print queue list
```

```
The method displayQueue consists of traversal and print. The output format is as follows:

#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;

If the queue is empty, the method displayQueue should print
     "There are no students waiting for a scholarship".
```

The method **peek** does return the name of the first student, but does not remove it form the queue. For example, for the queue printed above, it will return "Cedric Diggory".
If the queue is empty it should return the empty string.

The method **enqueue**(std) should create a new Node with the student content, and then add that node to the back of the queue.

The method **dequeue**( ) should return the student removed from the queue. If the queue is empty, it returns null.

### Requirements:

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

## Optional: Your Notes: Can Help Make up Some

## Marks if your Programs Don't Pass the Tests

You should be documenting what you are trying to do as you write your program. This can help reveal your software development process. As you work should be updating a file called **notes.txt** in your svn directory for this exam in the hogwarts directory.  If you commit this file we can give you marks for a detailed and informed description of your software development process that can make up to 50% of the marks for functionality **if your marks for functionality fall short of 50%** (that is 35/70).

This file is a narrative so we would expect to see things such as:

- your own words describing what you are trying next.
- very specific ideas about your approach to solving each part of the problem and the test cases you will use to identify that component is working.
- notes on the bugs you encounter, what you think they are and the tests that you do to isolate them.
- notes on your design.
- anything that relates to process you are following to develop the software.

These notes are a narrative so they must be readable sequentially. They don't have to be well formatted or divided into themes they have to describe in detail what you are doing as you do it.

## Help and Test Cases (look at the driver code at end!)

### 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. One scholarship is granted to the first student in the queue.
2. Recently, Mr. Malfoy had family financial issues. He needs to be added to queue;
```

#### 3. Display Queue

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

    Position  Student Name, Age years old, Period year in Hogwarts;
```

#### 4. Empty queue

```
Remember to deal with the special case of an empty queue for each method that access the queue
.
```

```
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();
        Student next = howgarts.dequeue();
        next.printStudent();



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

        howgarts.displayQueue();

    }
}
```

**Practical Exam 06**

| Criteria | Ratings | | | | Pts |
|---|---|---|---|---|---|
| Code Style | **30 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 Pts** **Good** Your code (1) has useful comments and place line and block comments correctly, (2) follow indentation, (3) has good variable naming. | **0 Pts** **No marks** You code (1) don't have comments. | | 30 pts |
| Basic class for student | **21 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 | **17 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) | **11 Pts** **Fair** Your code (1) perform almost all the functions correctly, (2) use concepts | **0 Pts** **No marks** Your code does not compile | 21 pts |

| | | | | | |
|---|---|---|---|---|---|
| | solving the problem, (4) your code implements accessors and mutators; | your code has the implementation of accessors and mutators. | learned in class, (3) has a way of solving the problem. | | |
| Basic class for Node | **14 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 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 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 Pts** **No marks** Your code does not compile | 14 pts |
| Basic class for Queue | **35 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 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 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 Pts** **No marks** Your code does not compile | 35 pts |
| | | | | | Total points: 100 |