CS310 Data Structures Fall 2019

Programming Assignment 1

PART 1: Array Implementation of Stacks with Java Files (40 Points)

Author original Java code to do the following:

Create a StackInterface.java (Week4 Slide 7), StackArray.java implements the interface. Main Program uses StackArray.java to analyze the input text file OR Main Program can directly implement STackInterface.java (Refer to Submission Versions at the end of this document)

1. The Java program must be able to read a text file called “Input.txt”
2. The contents of the text file must be a Java program for which the main program must check for **Balanced Parentheses**
3. The main program must read the Java program text file line by line and must push / pop elements from a stack every time it encounters a parenthesis.
4. The stack must be implemented using an Array. The data type of the array is irrelevant, as long as there is a way to count the parentheses.
5. The result of the analysis, if the java text file has balanced parentheses or not must be indicated to the user on the console.
6. Assume you are only checking for one kind of parenthesis i.e. {}

Rubrics:

(10) Java Files Implemented Successfully
(10) Stack is Implemented as an Array, all stack methods are implemented
(10) Program give expected output
(10) Big O analysis for time complexity is done for the solution

Sample Input file:

```
public class HelloWorld {

    public static void main(String[]
args)  {
        System.out.println("Hello,
World");
    }

}
```

Output:

```
Parentheses are balanced
```

Sample Input file:

```
public class HelloWorld {

    public static void main(String[]
args)  {
        System.out.println("Hello,
World");

}
```

Output:

```
Parentheses are not balanced
```

PART 2: Array Implementation of Queues with Java Files: Rudimentary Student enrollment program (60 Points)

Author original Java code to do the following:

Create a QueueInterface.java (Week4 Slide 19), QueueArray.java implements the interface.

StudentInfo.java defines the data type of the Queue.

Main Program uses QueueArray.java and StudentInfo.java to simulate student registration OR Main program can directly implement the QueueInterface. (Refer to Submission Versions at the end of this document)

1. The Java program must be able to read a text file called "Records.txt"
2. The contents of the text file must be a collection of records of type: Name<space>REDID (One record per line). Assume there are never any errors in the records file.
3. The data class must be Student with Name and REDID as variables
4. The main java program must be able to read the text file and create an array of objects of type Student.
5. The main java program does the following after reading the data file:
    a. Ask the user to enter the number of seats open for their class.
    b. Create one Enrolled Student List (Size: Number of Seats open, implemented as an array) and one Wait Queue (implemented as a circular array)
    c. Add the names and ids of the students in the Enrolled Student List in order in which it was read from the file (top to bottom)
    d. Give the user an option to delete an item from the Enrolled Students List by REDID (You would implement a Linear search algorithm for the list)
    e. Once a student is deleted from the Enrolled Student List, dequeue the front of the Wait List Queue and add the student Name and REDID to the Enrolled Student List.
    f. Only the Wait List Queue uses the Queue data structure with its appropriate methods.
    g. The Enrolled Student List is just an array on which a Linear Search Algorithm can be run.
    h. The user must also be able to display the entire Enrolled Student List.

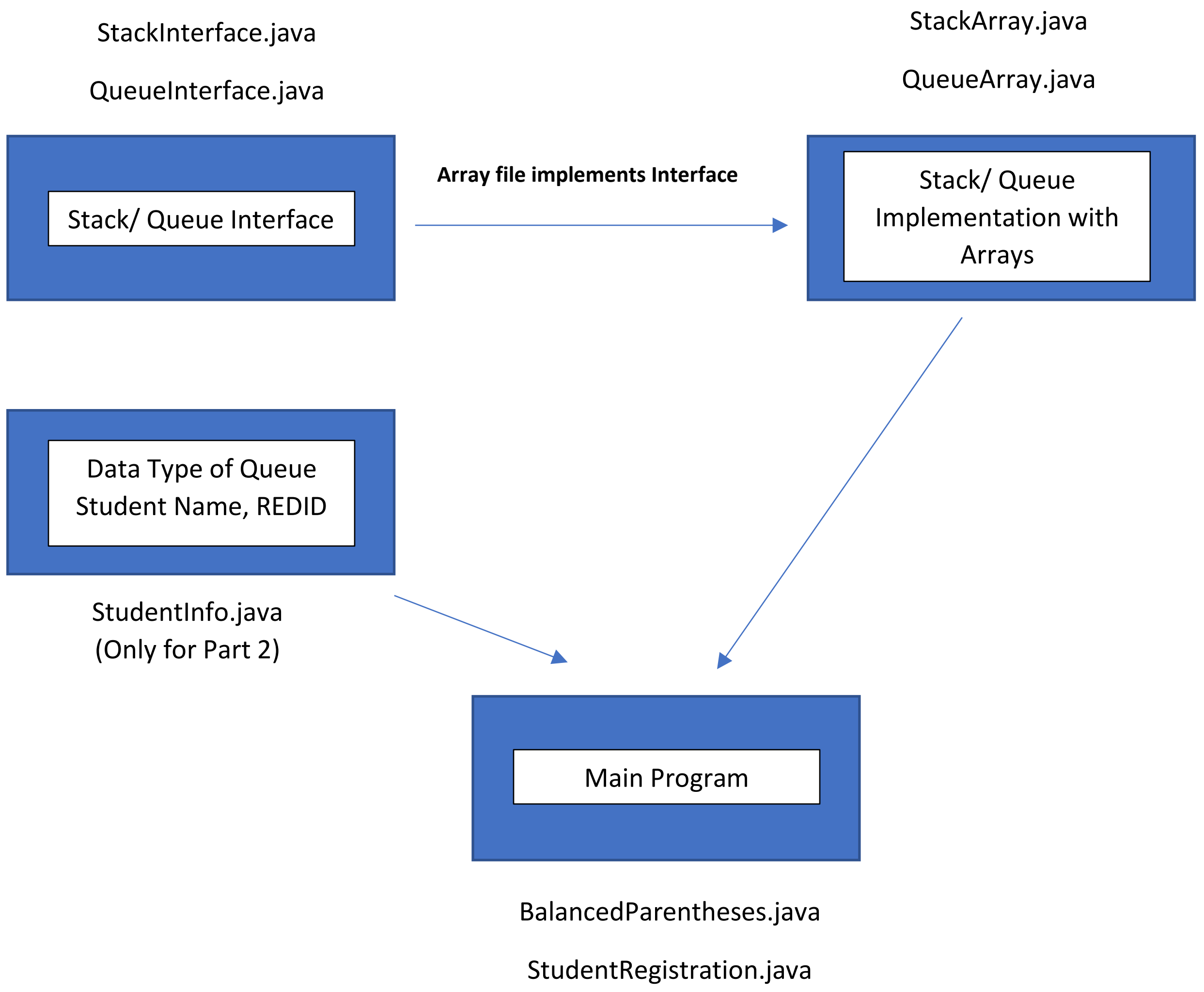
Rubrics:

(10) Java Files Implemented Successfully
(10) Wait Queue is Implemented as a Circular Array, all queue methods are implemented
(10) Linear Search is correctly implemented on the Enrolled Student List
(10) Data Class and the main program are separated and the Data class is reusable.
(10) Program give expected output
(10) Big O analysis for time complexity is done for the solution

Both programs can be written inside the same Java Project

SUBMIT ONLY ONE OF THE FOLLOWING VERSIONS OF THE PROJECT

SUBMISSION VERSION 1



Files to be submitted:

Submit an appropriately named ZIP file containing the following:

1. ReadMe File:
   A technical document explaining all the classes used.
   Write an analysis of your solution and find time complexity of both your programs using the Big O Notation.
2. JAR file of the following classes:

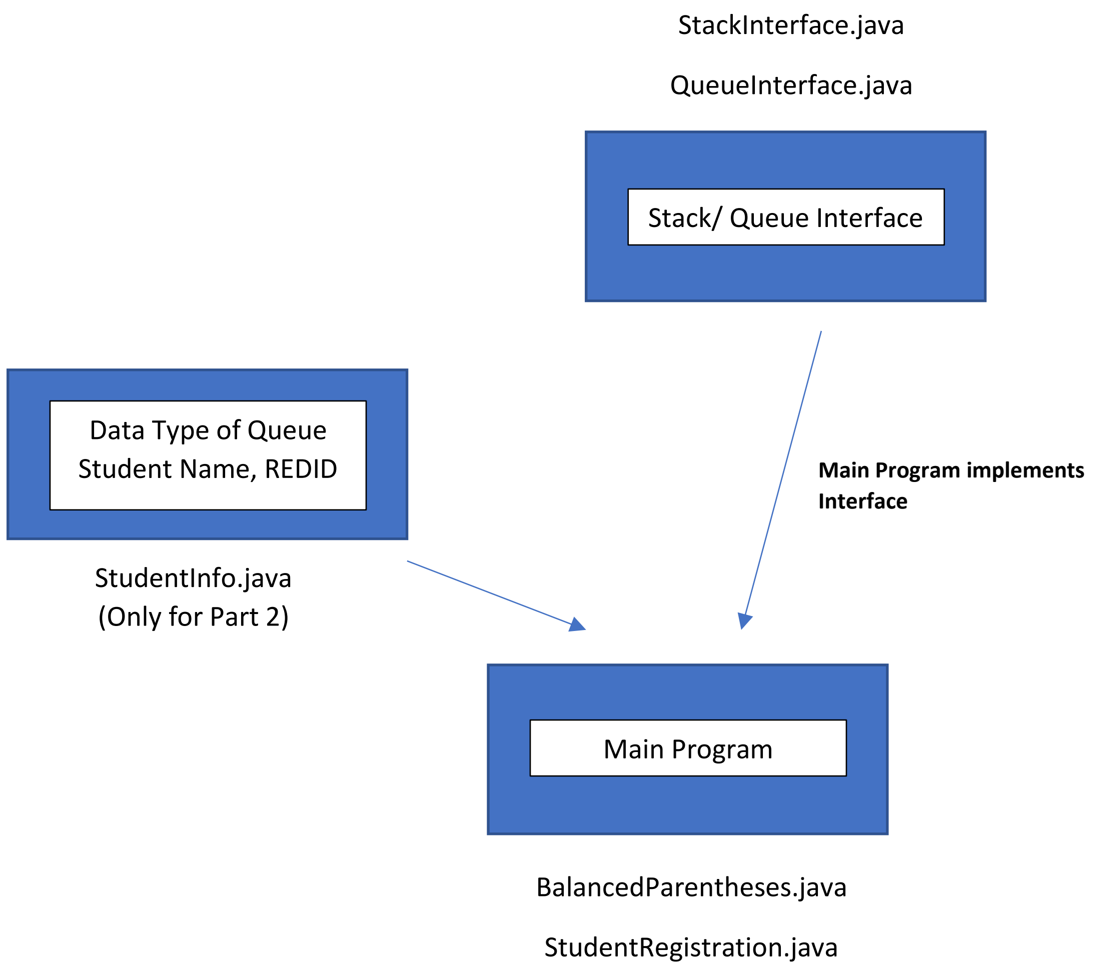
Part 1:

a. StackInterface.java
b. StackArray.java
c. BalancedParentheses.java

Part 2:

a. QueueInterface.java
b. QueueArray.java
c. StudentRegistration.java
d. StudentInfo.java

SUBMISSION VERSION 2



Files to be submitted:

Submit an appropriately named ZIP file containing the following:

3. ReadMe File:
   A technical document explaining all the classes used.
   Write an analysis of your solution and find time complexity of both your programs using the Big O Notation.
4. JAR file of the following classes:

Part 1:
a. StackInterface.java
b. BalancedParentheses.java

Part 2:
a. QueueInterface.java
b. StudentRegistration.java
c. StudentInfo.java