CLEMSON
School of Computing

Clemson University – School of Computing

CPSC2150: Software Development Foundations

Lab 4

Written by:

Damion Anderson
Jeremy Holloway

Academic Year: 2018 - 2019

Table of Contents

1. Requirement Analysis

User Story:

- As a user of MyQueue, I will be able to choose the data structure type to use the most beneficial type for my system.

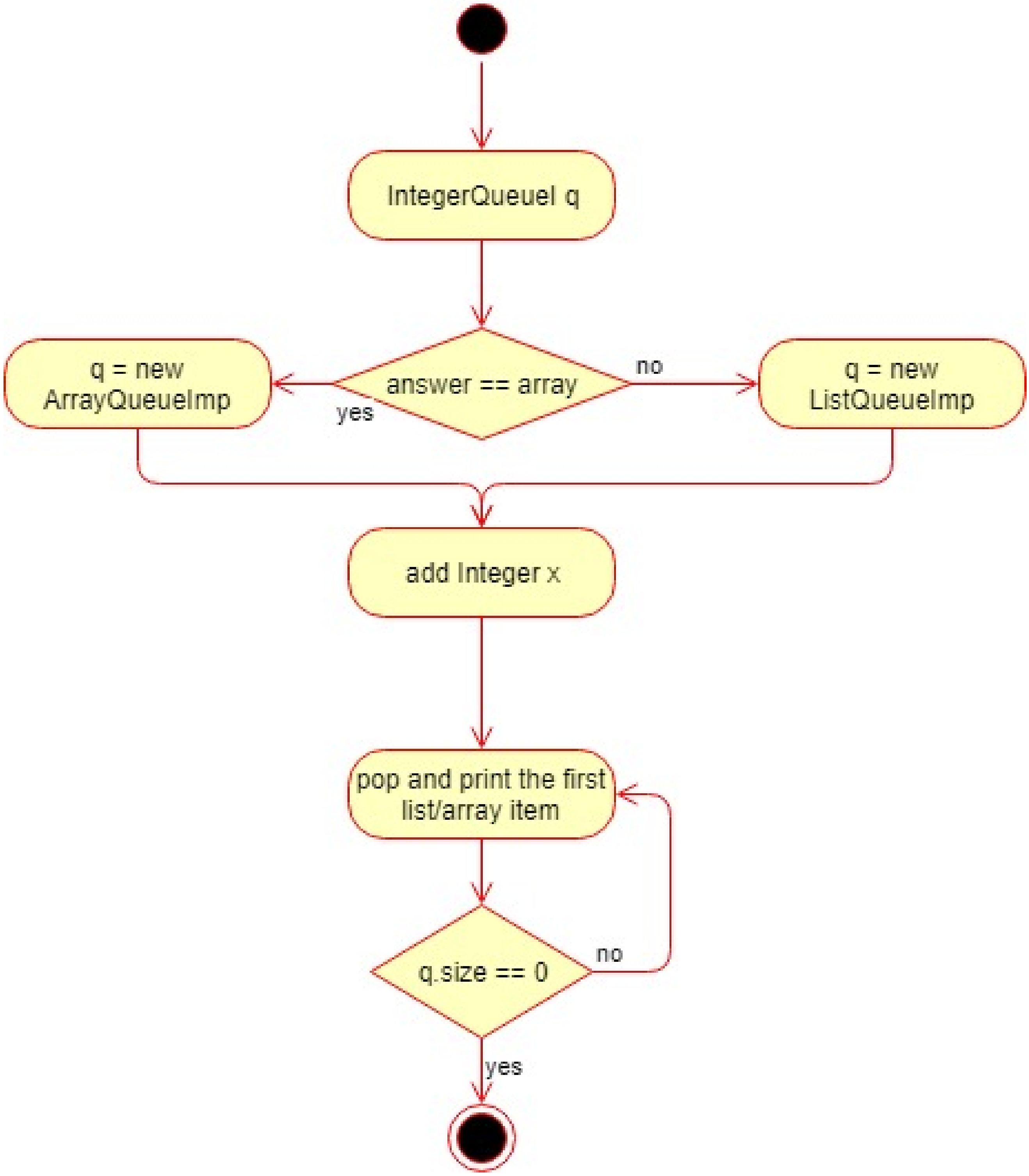
Functional Requirements:

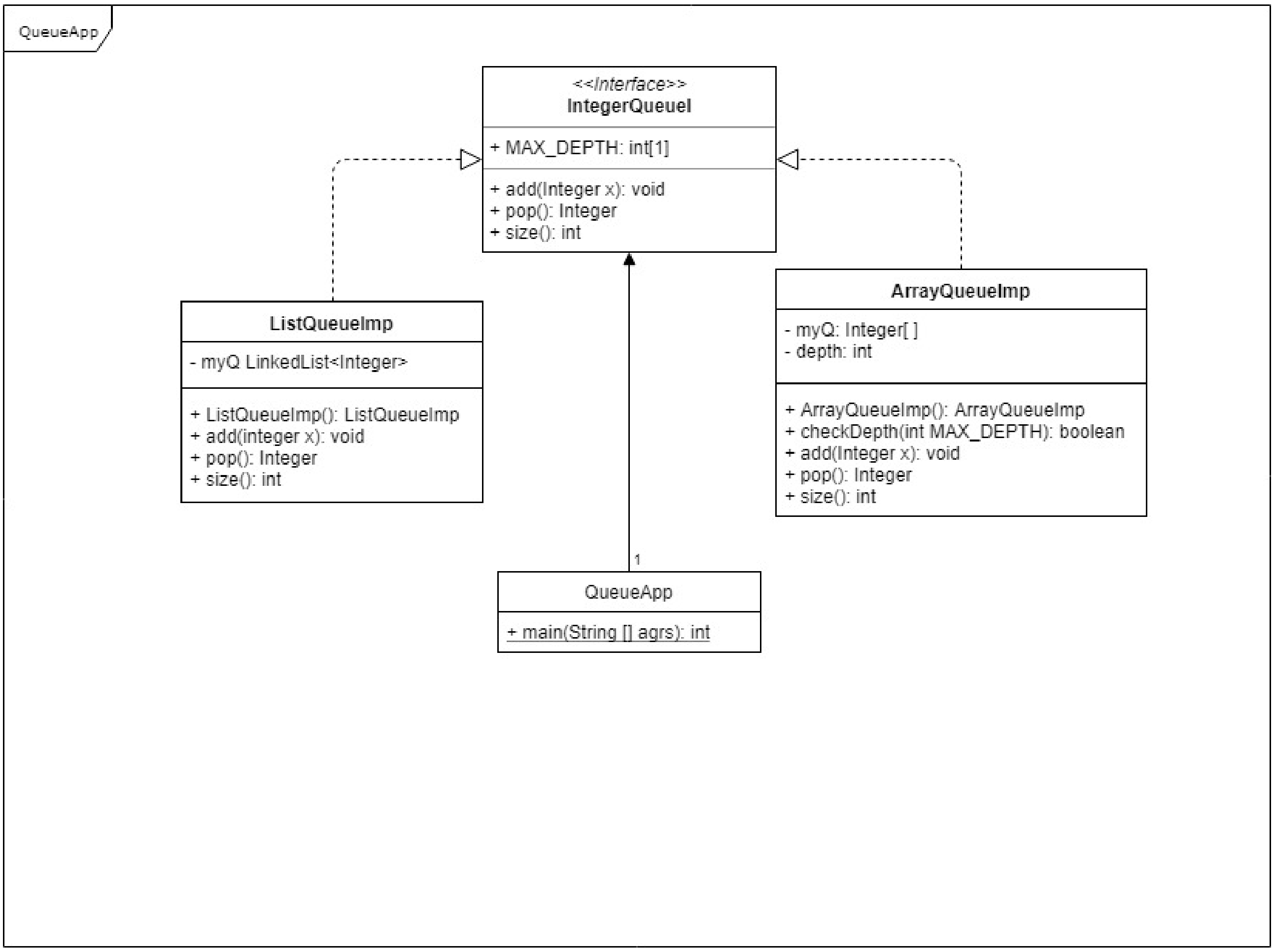
- A user will be able to choose a preferred method implementing a queue.
- A user will be able to add and remove items to or from their preferred storage method.
- A user will be able to display the number of items in their storage structure

Non-Functional Requirements:

- MyQueue is designed to take advantage of the list and array structures and allow the user to select whether to have a high or low memory overhead.
- The user will have an instant response from the app.
- The program is modular. If the client desires to remove certain functions, they may be commented out. These methods will work on any system with JDK8 installed.
- The program is written in Java. The system must have JDK8 installed. The program was designed for a Unix based environment.
- All interfaces between the user and the app will be displayed via the terminal window.
- The app is delivered via one main.java file. The user must compile via the javac Main.java command and run the app via the java main command.

2. Design


IntegerQueueI q
answer == array
yes
no
q = new ArrayQueueImp
q = new ListQueueImp
add Integer x
pop and print the first list/array item
q.size == 0
no
yes


QueueApp
<<Interface>>
IntegerQueueI
+ MAX_DEPTH: int[1]
+ add(Integer x): void
+ pop(): Integer
+ size(): int
ListQueueImp
- myQ LinkedList<Integer>
+ ListQueueImp(): ListQueueImp
+ add(integer x): void
+ pop(): Integer
+ size(): int
ArrayQueueImp
- myQ: Integer[ ]
- depth: int
+ ArrayQueueImp(): ArrayQueueImp
+ checkDepth(int MAX_DEPTH): boolean
+ add(Integer x): void
+ pop(): Integer
+ size(): int
1
QueueApp
+ main(String [] agrs): int

3. Testing Methods

Testing will be performed by calling the application to use a list structure and an array structure

LIST
Expected Output:

```
Would you like to use an array or a list?
list
42
17
37
36
12

Process finished with exit code 0
```

ARRAY
Expect Output:

```
Would you like to use an array or a list?
Array
42
17
37
36
12

Process finished with exit code 0
```

4. Deployment

This program will include a make file which can be used for three operations make, make run, make clean. The makefile must be in the same folder as cpsc2150 and not inside of it or any other subfolder.

make:
Expected output:

```
javac cpsc2150/QueueApp/QueueApp.java
```

make run:
Expected output:

```
java cpsc2150.QueueApp.QueueApp
Would you like to use an array or a list?
```

make clean:
Expected output:

```
rm -f *.class
```