Queue implementation

```
import java.util.Iterator;
import java.util.LinkedList;

public class QueueImpl<T> implements Iterable<T> {


   @Override
   public Iterator<T> iterator() {
      return null;
   }
   private LinkedList<T> queueList = new LinkedList<T>();

   public QueueImpl() {}

   public  QueueImpl(T firstElement){
      addElement(firstElement);
   }

   public int howMany(){
      return queueList.size();
   }

   public boolean isEmpty(){
      return howMany() == 0;
   }

   public T checkFirst(){
      if (isEmpty()){
         throw new RuntimeException("Empty Queue of Names!");
      }
      return queueList.peekFirst();
   }

   public T removeFirstElement(){
      if (isEmpty()){
         throw new RuntimeException("Cannot remove element from an empty queue!");
      }
      return queueList.removeFirst();
   }


   public void addElement(T element){
      if (element == null){
         throw new RuntimeException("Cannot add null elements to the list!");
      }
      queueList.addLast(element);
   }

}
```

Queue Usage:

```
import java.util.Objects;

public class Assignment1 {


    QueueImpl newQueue = createQueue("str");

    public QueueImpl createQueue(String type){
        if (type.equals("int")){
            QueueImpl<Integer> newQueue = new QueueImpl<>();
            return newQueue;
        }
        else {
            QueueImpl<String> newQueue = new QueueImpl<>();
            return newQueue;
        }
    }


    public String addInt(int number){
        try {
            newQueue.addElement(number);
            return("Success");
        }
        catch (RuntimeException exception){
            System.out.println("Error while adding element to the queue " + number);
            return("Error");
        }
    }

    public String peekFirst(){
        try {
            newQueue.checkFirst();
            return("Success");
        }
        catch (RuntimeException exception){
            System.out.println("Error while checking the first element of the queue");
            return("Error");
        }
    }


    public String removeElement(){
        try {
            newQueue.removeFirstElement();
            return("Removed");
        }
        catch (RuntimeException exception){
            System.out.println("Error while removing first element of the queue.");
```

```
            return("Error");
        }
    }


    public int checkSize(){
        try {
            return newQueue.howMany();
        }
        catch (RuntimeException exception){
            System.out.println("Error while adding element to the queue");
            return(-1);
        }
    }

    public String checkEmpty(){
        try {
            if (newQueue.isEmpty()) {
                return("Success");
            }
            else{
                return("Not Empty");
            }

        }
        catch (RuntimeException exception){
            System.out.println("Error while checking if empty.");
            return("Error");
        }
    }



    public static void main(String[] args) {

        Assignment1 assignment1 = new Assignment1();

//      for (int i = 0; i < 100; i+=2){
//          assignment1.addInt(i*i);
//      }
        assignment1.checkSize();

        assignment1.checkEmpty();

        assignment1.peekFirst();

        assignment1.addInt(1500);

        assignment1.checkEmpty();

        assignment1.checkSize();
```

```
        assignment1.removeElement();
    }


}
```

JUnit 5 Test Cases Implementation

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;



class Assignment1Test {
    Assignment1 assignment1 = new Assignment1();
    QueueImpl newQueue = assignment1.createQueue("int");


    @BeforeEach
    void setUp() {
        assignment1.addInt(10);
        assignment1.addInt(11);
        assignment1.addInt(12);
        assignment1.addInt(15);
    }

    @AfterEach
    void tearDown() {
        for (int i = 0; i < assignment1.checkSize(); i++){
            assignment1.removeElement();
        }
    }


    @Test
    void createQueue() {
        RuntimeException thrown = Assertions.assertThrows(
                RuntimeException.class,
                () -> assignment1.createQueue("bool"),
                "Expected assignment1.createQueue() to throw RuntimeException, but it didn't"
        );
        Assertions.assertTrue(thrown.getMessage().contains("string Or integer"));
        System.out.println("Boolean test for createQueue() successfull!");

        RuntimeException thrown2 = Assertions.assertThrows(
```

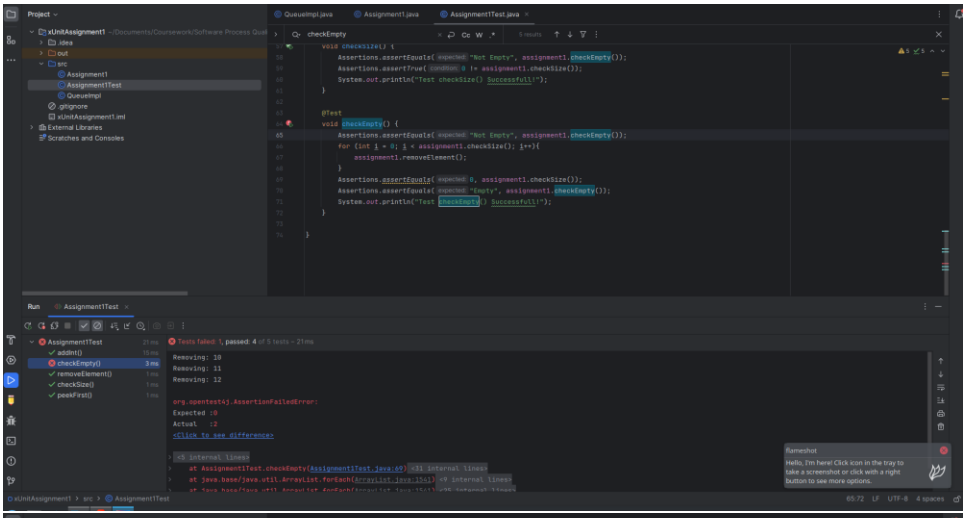
```
            RuntimeException.class,
            () -> assignment1.createQueue("short"),
            "Expected assignment1.createQueue() to throw RuntimeException, but it didn't"
    );
    Assertions.assertTrue(thrown2.getMessage().contains("string Or integer"));
    System.out.println("Short test for createQueue() successfull!");

    RuntimeException thrown3 = Assertions.assertThrows(
            RuntimeException.class,
            () -> assignment1.createQueue("array"),
            "Expected assignment1.createQueue() to throw RuntimeException, but it didn't"
    );
    Assertions.assertTrue(thrown3.getMessage().contains("string Or integer"));
    System.out.println("Array test for createQueue() successfull!");

    RuntimeException thrown4 = Assertions.assertThrows(
            RuntimeException.class,
            () -> assignment1.createQueue("long"),
            "Expected assignment1.createQueue() to throw RuntimeException, but it didn't"
    );
    Assertions.assertTrue(thrown4.getMessage().contains("string Or integer"));
    System.out.println("Long test for createQueue() successfull!");

    QueueImpl tempQueue = assignment1.createQueue("str");
    Assertions.assertEquals(tempQueue.getClass(), QueueImpl.class);
    System.out.println("Class check test for createQueue() successfull!");
}


@Test
void addInt() {
    int oldSize = assignment1.checkSize();
    assignment1.addInt(1000);
    int newSize = assignment1.checkSize();
    Assertions.assertTrue(newSize > oldSize);
    System.out.println("Test addInt() Successfull!");
}

@Test
void peekFirst()
{
    Assertions.assertEquals("Success", assignment1.peekFirst());
    System.out.println("Test peekFirst() Successfull!");
}

@Test
void removeElement() {
    int oldSize = assignment1.checkSize();
    Assertions.assertEquals("Removed", assignment1.removeElement());
    int newSize = assignment1.checkSize();
    Assertions.assertTrue(newSize < oldSize);
    System.out.println("Test removeElement() Successfull!");
```

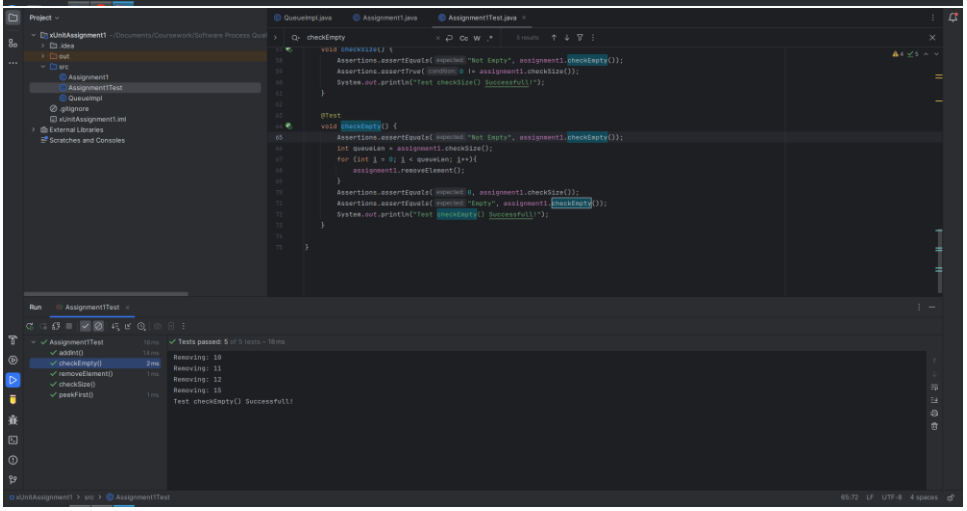
```
    }

    @Test
    void checkSize() {
        Assertions.assertEquals("Not Empty", assignment1.checkEmpty());
        Assertions.assertTrue(0 != assignment1.checkSize());
        System.out.println("Test checkSize() Successfull!");
    }

    @Test
    void checkEmpty() {
        Assertions.assertEquals("Not Empty", assignment1.checkEmpty());
        int queueLen = assignment1.checkSize();
        for (int i = 0; i < queueLen; i++){
            assignment1.removeElement();
        }
        Assertions.assertEquals(0, assignment1.checkSize());
        Assertions.assertEquals("Empty", assignment1.checkEmpty());
        System.out.println("Test checkEmpty() Successfull!");
    }

}
```

Errors, faults and failures:

- The implementation of queue in this above example is a generic self-implemented queue with methods such as create, peekFirst element, removeFirst element, add new element.
- The Queue supports only two types, string and integer queues for the sake of simplicity in implementation.
- The class Assignment1 makes use of methods of the queue and has functions that can be used to populate the queue, remove elements, check length / size of the queue.
- The implementation of test cases is done using Junit version 5 which is the latest version.
- The Assignment1 class is utilized to create a queue to test the functions present in the class.
- The Test class consists of a setup and a teardown class that makes use of the created queue to repopulate and remove all elements from the queue at the start of every test function.
- Due to the mistake in implementation of test functions, the test cases failed with a runtime exception that is invoked on the QueueImpl class.
- In the checkEmpty test case, the queue was not empty and hence the test case was failing due to improper handling of removal of elements.
- The checkEmpty method failed the next time because of a mistake in the loop definition using assignment.checkSize() method in the for loop definition instead of saving the

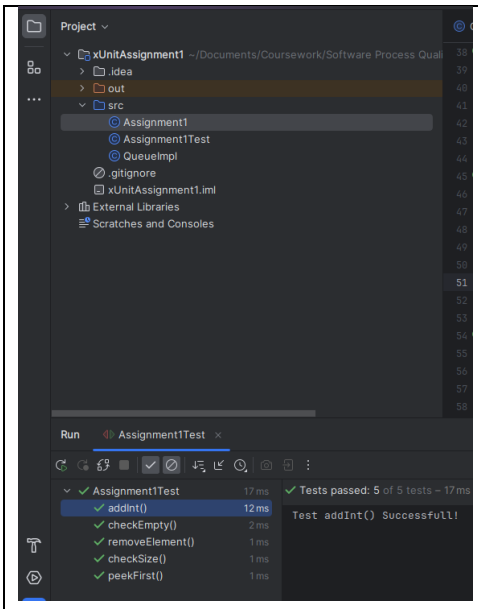
lenght of the queue in an integer variable causing the loop to run n / 2 times instead of n times for removing the elements in the queue.

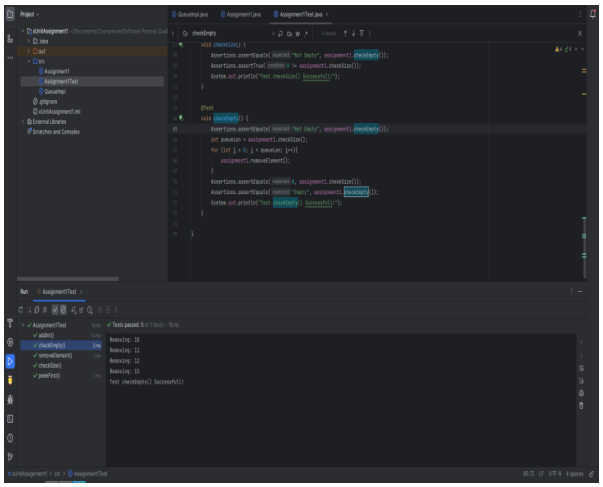


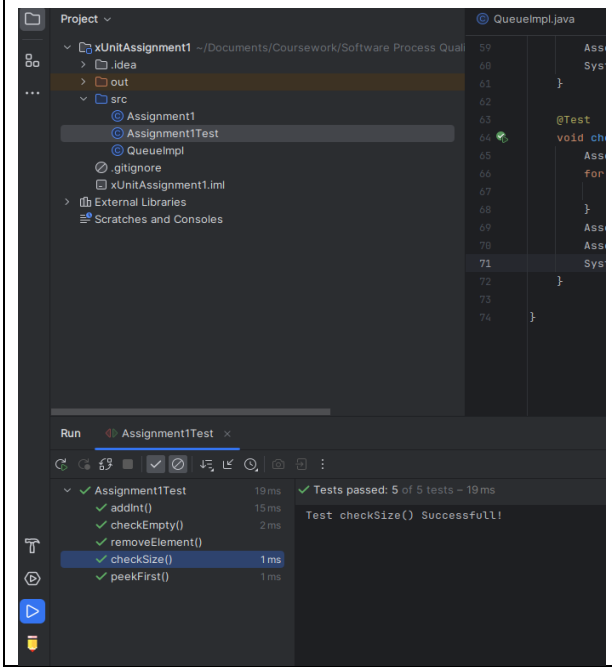


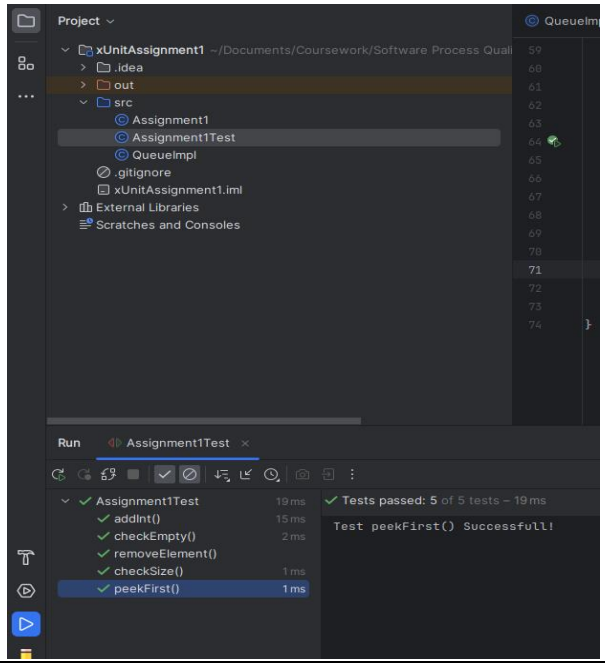
- Difficulty with implementing test for createQueue method for exceptions and other types of queues such as boolean or long.
- Checking the class of the created queue to check whether the class of the created queue is of valid QueueImpl class.

SAMPLE RUNS


Project
xUnitAssignment1 ~/Documents/Coursework/Software Process Quali
.idea
out
src
Assignment1
Assignment1Test
QueueImpl
.gitignore
xUnitAssignment1.iml
External Libraries
Scratches and Consoles
Run
Assignment1Test
Assignment1Test 17 ms
addInt() 12 ms
checkEmpty() 2 ms
removeElement() 1 ms
checkSize() 1 ms
peekFirst() 1 ms
Tests passed: 5 of 5 tests – 17 ms
Test addInt() Successfull!




Project
xUnitAssignment1 ~/Documents/Coursework/Software Process Quali
.idea
out
src
Assignment1
Assignment1Test
QueueImpl
.gitignore
xUnitAssignment1.iml
External Libraries
Scratches and Consoles
QueueImpl.java
Run
Assignment1Test
Assignment1Test 19 ms
addInt() 15 ms
checkEmpty() 2 ms
removeElement()
checkSize() 1 ms
peekFirst() 1 ms
Tests passed: 5 of 5 tests – 19 ms
Test checkSize() Successfull!


Project
xUnitAssignment1 ~/Documents/Coursework/Software Process Quali
.idea
out
src
Assignment1
Assignment1Test
QueueImpl
.gitignore
xUnitAssignment1.iml
External Libraries
Scratches and Consoles
Run
Assignment1Test
Assignment1Test 19 ms
addInt() 15 ms
checkEmpty() 2 ms
removeElement()
checkSize() 1 ms
peekFirst() 1 ms
Tests passed: 5 of 5 tests – 19 ms
Test peekFirst() Successfull!