Report on s1769931 Inf1OP 2018 PE1 Programming Exam

Generated by Automarker on April 14, 2019

Question 1

Submitted Files: OK

Compiling RingBuffer.java alone: OK

Compiling RingBuffer.java with basic tests provided in the exam: OK

Running Basic Tests: All passed

Compiling RingBuffer.java with main tests: OK

Running Main Tests: Passed 39 of 45 main tests

Failed Tests:

Question	Comments	Impact
1.2	**test1AConstructorElementCountInit**: RingBuffer one-arg constructor is expected to initialise elementCount member to 0. `java.lang.AssertionError: elementCount not initialised correctly. expected:<0> but was:<40>`	−1.5
1.2	**testNoAConstructorInit**: RingBuffer no-arg constructor is expected to call RingBuffer one-arg constructor with a default buffer capacity of 10. `java.lang.AssertionError: elementCount not initialised correctly. expected:<0> but was:<10>`	−3
1.3	**testIsFullOnInit**: newly created buffer is not expected to be full. `java.lang.AssertionError: Newly created single element buffer is not expected to be full.`	−2
1.4	**testIsEmptyOnInit**: newly created buffer is expected to be empty. `java.lang.AssertionError: Newly created single element buffer is expected to be empty.`	−2
1.7	**testGetFromBufferRegularStartEnd**: When getting an element from RingBuffer, end is not expected to move and start is expected to point at the next index. `java.lang.AssertionError: Start is expected to move forward by one. expected:<1> but was:<39>`	−2

1.7	**testGetFromBufferWrapAround**: When getting an element from RingBuffer which is the last one in the buffer array and there are more elements in the beginning, start is expected to wrap around to zero, end is not expected to move, elementCount is epxected to decrease by one, the returned element is expected to be the one previously pointed at by start and the position in the buffer array previously pointed at by start is expected to be set to minus one. 1 `java.lang.AssertionError: Start is expected to be zero after wrap around. expected:<0> but was:<38>`	−1.5

Q1: 38.0 / 50

Question 2

Submitted Files: OK

Compiling BattleShips.java alone: OK

Compiling BattleShips.java with basic tests provided in the exam: OK

Running Basic Tests: All passed

Compiling BattleShips.java with main tests: OK

Running Main Tests: Passed 36 of 38 main tests

Failed Tests:

Question	Comments	Impact
2.1	**testFireShotGridChanged**: WARNING: The given game grid is not expected to be changed. See comments for impact. 1 `Grid is not expected to be changed.: arrays first differed at element [2]; expected:<B> but was:<#>`	0
2.2	**testFindShipsIsHashtable**: The expected return type of findShips is Hashtable. 1 `java.lang.AssertionError: Returned type is expected to be a Hashtable.`	−1

Q2: 49 / 50

Total Marks: 87.0/100

Submitted Files

Submitted: RingBuffer.java

```
/*
Buffer is used to temporarily store data while it is being moved from one
 place to the other

 RingBuffer:
  - is filled with data by adding to the back and emptied by removing from the
  front
  - wraps saved data entries around the end of the underlying array
  - does that by keeping track of the location of the initial data entry in
 the buffer and the location of the last data entry


 When the buffer is empty, both Start and End markers point to the same array
 index.
 The same is true if the buffer contains a single element
 */



public class RingBuffer extends Buffer {
 private int start;
 private int end;
 private int elementCount;//Amount of valid data currently stored
 public RingBuffer(int capacity) {
  super(capacity);
  this.elementCount = capacity;
 }
 public RingBuffer() {
  super(10); //Default capcity of 10
  this.elementCount = 10;
 }

 public boolean isFull() {
  return (this.elementCount == super.buffer.length);
 }

 public boolean isEmpty() {
  return (this.elementCount == 0);
 }

 public void clear() {
  super.clear();
  this.start = 0;
  this.end = 0;
  this.elementCount = 0;
 }

 public void addToBuffer(int newNum) {
  if (this.isFull()) System.out.println("Buffer is full.\n");
  else if (this.end == super.buffer.length-1 || this.isEmpty()) {
   //if the end-point is at the end or if the buffer is empty, put the number
 at the start
   end = 0;
```

```
    super.buffer[0] = newNum;
    this.elementCount++;
   } else {
    end++;
    super.buffer[end] = newNum;
    this.elementCount++;
   }

  }

  public int getFromBuffer() {
   if (this.isEmpty()) {
    System.out.println("Buffer is empty.\n");
    return -1;
   } else if (this.start == this.end) {
    int temp = super.buffer[start];
    super.buffer[start] = -1;
    this.elementCount--;
    return temp;

   } else if (this.start == 0) {
    int temp =  super.buffer[0];
    super.buffer[0] = -1;
    this.start = super.buffer.length-1;
    this.elementCount--;
    return temp;
   } else {
    int temp = super.buffer[this.start];
    super.buffer[this.start] = -1;
    this.start--;
    this.elementCount--;
    return temp;
   }
  }

  public static void main(String[] args) {

  }

}
```

Submitted: BattleShips.java

```
import java.util.Map;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

class BattleShips {

    public static char fireShot(char[][] grid, Position shot) {
     try {
      int x = shot.getX();
      int y = shot.getY();
      char temp = grid[y][x];
      if (temp == '#') {
       System.out.println(shot + ": Miss");
```

```
      }
      else {
       grid[y][x] = '#';
       System.out.println(shot + ": Hit " + temp);
      }
      return temp;
     } catch (IndexOutOfBoundsException e) {
      System.out.println(shot + ": " + "Out of Bounds");
      return '#';
     }
    }

    public static Map<Character, Integer> findShips(char[][] grid) {
     HashMap<Character, Integer> outputMap = new HashMap<>();
     for (char[] row : grid) {
      for (char c : row) {
       if (c!='#') outputMap.put(c,outputMap.getOrDefault(c, 0)+1);
      }
     }
     return outputMap;
    }


    public static List<Character> fireShots(char[][] grid, List<Position>
shots) {
     ArrayList<Character> outputList = new ArrayList<>();
     Map<Character, Integer> originalShips = findShips(grid);

     for (Position p: shots) {
      char firedLetter = fireShot(grid, p);
      if (firedLetter != '#') {
       if (originalShips.get(firedLetter) == 1) outputList.add(firedLetter);
       else originalShips.put(firedLetter, originalShips.get(firedLetter)-1);
      }
     }
     return outputList;
    }

    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Please execute this program by"
                    + " providing the path to a game file,"
                    + " e.g. java BattleShips ./game01.txt");
            System.exit(1);
        }

        char[][] grid = BattleShipsUtils.parseGrid(args[0]);

        System.out.println("Playing with grid: ");
        BattleShipsUtils.printGrid(grid);

        Position centreShot = new Position(grid.length / 2,
                                           grid.length / 2);
        System.out.println("\nFiring Shot at centre " +
                           centreShot.toString() + " ...");
        fireShot(grid, centreShot);
```

```
70
71          System.out.println("\nGenerating␣shots␣...");
72          List<Position> shots = BattleShipsUtils.generateShots(grid.length,
73                                          grid.length * 2);
74          System.out.println("Shots:␣" + shots);
75
76          System.out.println("\nFinding␣Ships␣...");
77          Map<Character, Integer> ships = findShips(grid);
78       if(ships != null) {
79             for (Character ship : ships.keySet())
80               System.out.println(ship + "␣has␣size␣" + ships.get(ship));
81       }
82
83          System.out.println("\nFiring␣shots␣...");
84       if(shots != null) {
85              List<Character> destroyedShips = fireShots(grid, shots);
86           if (destroyedShips != null && !destroyedShips.isEmpty()) {
87               System.out.println("Destroyed␣ships:␣");
88               for(char ship : destroyedShips)
89                   System.out.println(ship);
90           }
91           System.out.println(destroyedShips);
92       }
93      }
94  }
```