951428 Liz Simpson 960689 Scott Simmons

CS-135 Coding Style

Task 7.1

```
1 import java.util.Scanner;
public class Sphinx {
49
        public static int unknown (int x, int y){if (x < y)return x;
 5
            else return y;
 6
80
     public static void main(String args[]) {
            Scanner input = new Scanner(System.in);
            System.out.print("x: ");
10
            int x = input.nextInt();
11
           System.out.print("y: ");
int y = input.nextInt();
12
13
14
            System.out.println();
System.out.println("unknown = " + unknown(x,y));
15
16
            System.out.println();
17
        }
18 }
19
```

Task 7.2

Main

```
    Main.java 
    □ Queue.java

▶ 🧬 CS-135 Coding Sytle ▶ 🕮 src ▶ 🔠 (default package) ▶ 😭 Main ▶ 💣 main(String[]) : void
 10/**
 2 \, * This class holds the main method that tests the Queue class.
  3
  4 public class Main {
  60
         * This constructor is not used. It is provided to prevent Checkstyle
         * reporting an error.
  8
         */
  9
 100
         private Main() {
 11
 12
13@
         * The main method that tests the Queue class.
 14
 15
         * @param args
 16
                      The command line arguments are not used.
 17
 18
         public static void main(String args[]) {
 19⊖
20
            int test;
             Queue queue = new Queue();
 21
 22
 23
            System.out.println(" --- Begin Experiment 1 ---");
 24
            System.out.println(" --- Empty ---");
 25
 26
            queue.empty();
 27
 28
            System.out.println("Build up a queue of one entry:");
             queue.enqueue(1);
 29
            System.out.println("
 30
                                   enqueue 1");
 31
 32
            System.out.println("Dequeue to make queue empty:");
             test = queue.dequeue();
 33
                                   dequeue: " + test);
 34
            System.out.println("
 35
            System.out.println("Test how 'dequeue' works on the empty queue:");
 36
 37
            test = queue.dequeue();
 38
            if (queue.isErrorFree()) {
 39
                 System.out.println(" dequeue: " + test);
 40
 41
             } else {
 42
                 System.out.println(" An error has occured.");
 43
 44
 45
            System.out.println(" --- End Experiment 1 ---");
 46
 47
            System.out.println(" --- Begin Experiment 2 ---");
 48
             System.out.println(" --- Empty ---");
 49
 50
             queue.empty();
 51
 52
            System.out.println("Build up a queue of five entries:");
 53
 54
            queue.enqueue(1);
             System.out.println("
 55
                                   enqueue 1");
 56
             queue.enqueue(2);
             System.out.println("
 57
                                   enqueue 2");
58
             queue.enqueue(3);
             System.out.println("
59
                                   enqueue 3");
             queue.enqueue(4);
 60
```

```
61
            System.out.println(" enqueue 4");
62
63
            queue.enqueue(5);
            System.out.println(" enqueue 5");
64
65
66
67
68
            System.out.println("enqueue another entry and"
                     + " check if 'out of memory'-protection works:");
            queue.enqueue(6);
69
            if (queue.isErrorFree()) {
    System.out.println(" enqueue 6");
70
71
            } else {
72
73
                System.out.println(" An error has occured.");
74
75
            System.out.println(" --- End Experiment 2 ---");
76
77
78
            System.out.println(" --- Begin Experiment 3 ---");
79
            queue.empty();
            System.out.println(" --- Empty ---");
80
81
82
            System.out.println("Build up a queue of three entries:");
83
84
            queue.enqueue(1);
85
            System.out.println("
                                    enqueue 1");
86
            queue.enqueue(2);
87
            System.out.println("
                                    enqueue 2");
88
            queue.enqueue(3);
89
            System.out.println(" enqueue 3");
90
91
            System.out.println("Take these three entries away.");
92
            while (!queue.isEmpty()) {
                test = queue.dequeue();
93
94
                                        dequeue: " + test);
                System.out.println("
95
96
            System.out.println(" --- End Experiment 3 ---");
97
98 }
```

Queue

```
Main.java ⋈ D Queue.java ⋈
▶ 🧬 CS-135 Coding Sytle ▶ 🤔 src ▶ 🔠 (default package) ▶ 😉 Qr
 2 * This class holds the Queue.
3 * @author 951428 and 960689
 5 */
  6
 7 public class Queue {
 8
         private static final int QUEUE_SIZE = 5;
 9
 10
 11
        private int front = QUEUE_SIZE - 1;
        private int back = QUEUE_SIZE - 1;
 12
        private int length = 0;
 13
 14
        private int[] queue = new int[QUEUE_SIZE];
15
16
        private boolean errorFree = true;
17
 180
        * Constructor for the queue.
*/
19
 20
         public Queue() {
210
 22
 23
         }
 24
 25@
         /**
         * Method for length.
 26
         * @return length of the queue
 27
 28
 29⊜
         public boolean isEmpty() {
 30
           return length == 0;
 31
 32
 33⊖
         * Method to check if the queue is full.
 34
         * @return the size of the queue
35
36
 37⊖
         public boolean isFull() {
         return length == QUEUE_SIZE;
32
39
40
         /**
410
42
         * method to check if the queue is error free.
         * @return bolean value
43
44
45⊜
         public boolean isErrorFree() {
46
           return errorFree;
         }
47
48
490
         * empties the queue.
50
51
52⊕
         public void empty() {
             front = QUEUE_SIZE - 1;
53
             back = QUEUE_SIZE - 1;
             length = 0;
55
 56
             errorFree = true;
57
         }
```

```
58
59⊜
        * takes te first item off the list.
60
61
         * @return the first item of the list
62
63⊜
64
        public int dequeue() {
            errorFree = !(isEmpty()) & errorFree;
65
66
67
68
            if (errorFree) {
                length--;
                if (front == QUEUE_SIZE - 1) {
                     front = 0;
69
70
71
                } else {
                    front++;
72
73
74
                return queue[front];
            } else {
                return 0;
75
76
77
78⊝
       }
        * Adds an item to the front of the list.
79
80
        * @param value the new value to go at the end of the queue.
81
        public void enqueue(int value) {
82⊖
83
            errorFree = !((isFull())) & errorFree;
84
            if (errorFree) {
85
                length++;
86
                if (back == QUEUE_SIZE - 1) {
87
                    back = 0;
88
                } else {
89
                    back++;
90
91
                queue[back] = value;
92
93
        }
94 }
95
```