

951428 Liz Simpson
960689 Scott Simmons

CS-135 Coding Style

Task 7.1

```
1 import java.util.Scanner;
2 public class Sphinx {
3
4     public static int unknown (int x, int y){if (x < y)return x;
5         else return y;
6     }
7
8     public static void main(String args[]) {
9         Scanner input = new Scanner(System.in);
10        System.out.print("x: ");
11        int x = input.nextInt();
12        System.out.print("y: ");
13        int y = input.nextInt();
14        System.out.println();
15        System.out.println("unknown = " + unknown(x,y));
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

1 /**
2  * This class holds the main method that tests the Queue class.
3  */
4 public class Main {
5
6     /**
7      * This constructor is not used. It is provided to prevent Checkstyle
8      * reporting an error.
9      */
10    private Main() {
11    }
12
13    /**
14     * The main method that tests the Queue class.
15     *
16     * @param args
17     *     The command line arguments are not used.
18     */
19    public static void main(String args[]) {
20        int test;
21        Queue queue = new Queue();
22
23        System.out.println(" --- Begin Experiment 1 ---");
24
25        System.out.println(" --- Empty ---");
26        queue.empty();
27
28        System.out.println("Build up a queue of one entry:");
29        queue.enqueue(1);
30        System.out.println("    enqueue 1");
31
32        System.out.println("Dequeue to make queue empty:");
33        test = queue.dequeue();
34        System.out.println("    dequeue: " + test);
35
36        System.out.println("Test how 'dequeue' works on the empty queue:");
37        test = queue.dequeue();
38
39        if (queue.isErrorFree()) {
40            System.out.println("    dequeue: " + test);
41        } else {
42            System.out.println("    An error has occurred.");
43        }
44
45        System.out.println(" --- End Experiment 1 ---");
46
47        System.out.println(" --- Begin Experiment 2 ---");
48
49        System.out.println(" --- Empty ---");
50        queue.empty();
51
52        System.out.println("Build up a queue of five entries:");
53
54        queue.enqueue(1);
55        System.out.println("    enqueue 1");
56        queue.enqueue(2);
57        System.out.println("    enqueue 2");
58        queue.enqueue(3);
59        System.out.println("    enqueue 3");
60        queue.enqueue(4);
61        queue.enqueue(5);
62    }
```

```

61     System.out.println("    enqueue 4");
62     queue.enqueue(5);
63     System.out.println("    enqueue 5");
64
65     System.out.println("enqueue another entry and"
66         + " check if 'out of memory'-protection works:");
67
68     queue.enqueue(6);
69     if (queue.isErrorFree()) {
70         System.out.println("    enqueue 6");
71     } else {
72         System.out.println("    An error has occurred.");
73     }
74
75     System.out.println(" --- End Experiment 2 ---");
76
77     System.out.println(" --- Begin Experiment 3 ---");
78
79     queue.empty();
80     System.out.println(" --- Empty ---");
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()) {
93         test = queue.dequeue();
94         System.out.println("    dequeue: " + test);
95     }
96     System.out.println(" --- End Experiment 3 ---");
97 }
98 }

```

Queue

```
Main.java Queue.java
CS-135 Coding Sytle src (default package) Q

1 /**
2  * This class holds the Queue.
3  * @author 951428 and 960689
4  *
5  */
6
7 public class Queue {
8
9     private static final int QUEUE_SIZE = 5;
10
11     private int front = QUEUE_SIZE - 1;
12     private int back = QUEUE_SIZE - 1;
13     private int length = 0;
14
15     private int[] queue = new int[QUEUE_SIZE];
16     private boolean errorFree = true;
17
18     /**
19      * Constructor for the queue.
20      */
21     public Queue() {
22     }
23
24
25     /**
26      * Method for length.
27      * @return length of the queue
28      */
29     public boolean isEmpty() {
30         return length == 0;
31     }
32
33     /**
34      * Method to check if the queue is full.
35      * @return the size of the queue
36      */
37     public boolean isFull() {
38         return length == QUEUE_SIZE;
39     }
40
41     /**
42      * method to check if the queue is error free.
43      * @return boolean value
44      */
45     public boolean isErrorFree() {
46         return errorFree;
47     }
48
49     /**
50      * empties the queue.
51      */
52     public void empty() {
53         front = QUEUE_SIZE - 1;
54         back = QUEUE_SIZE - 1;
55         length = 0;
56         errorFree = true;
57     }
58 }
```

```

58
59- /**
60  * takes te first item off the list.
61  * @return the first item of the list
62  */
63- public int dequeue() {
64     errorFree = !(isEmpty()) & errorFree;
65     if (errorFree) {
66         length--;
67         if (front == QUEUE_SIZE - 1) {
68             front = 0;
69         } else {
70             front++;
71         }
72         return queue[front];
73     } else {
74         return 0;
75     }
76 }
77
78- /**
79  * Adds an item to the front of the list.
80  * @param value the new value to go at the end of the queue.
81  */
82- public void enqueue(int value) {
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

```