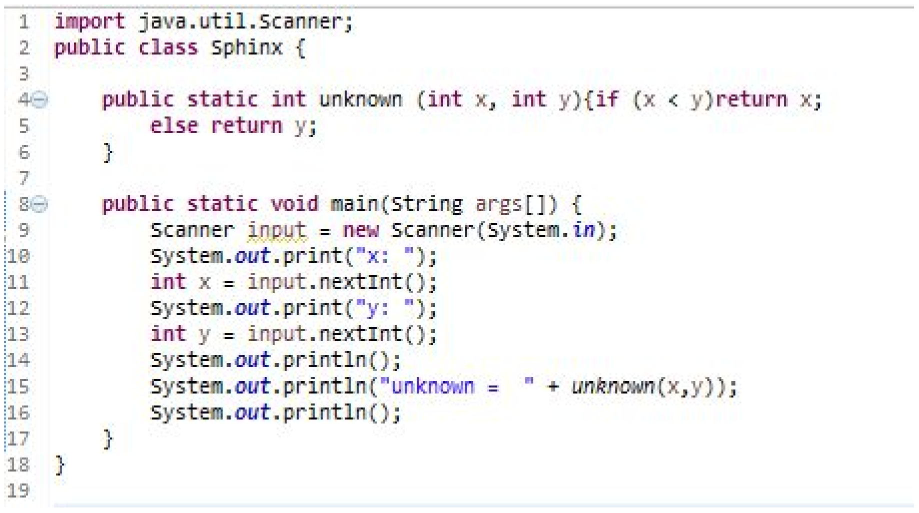
951428 Liz Simpson
960689 Scott Simmons

CS-135 Coding Style

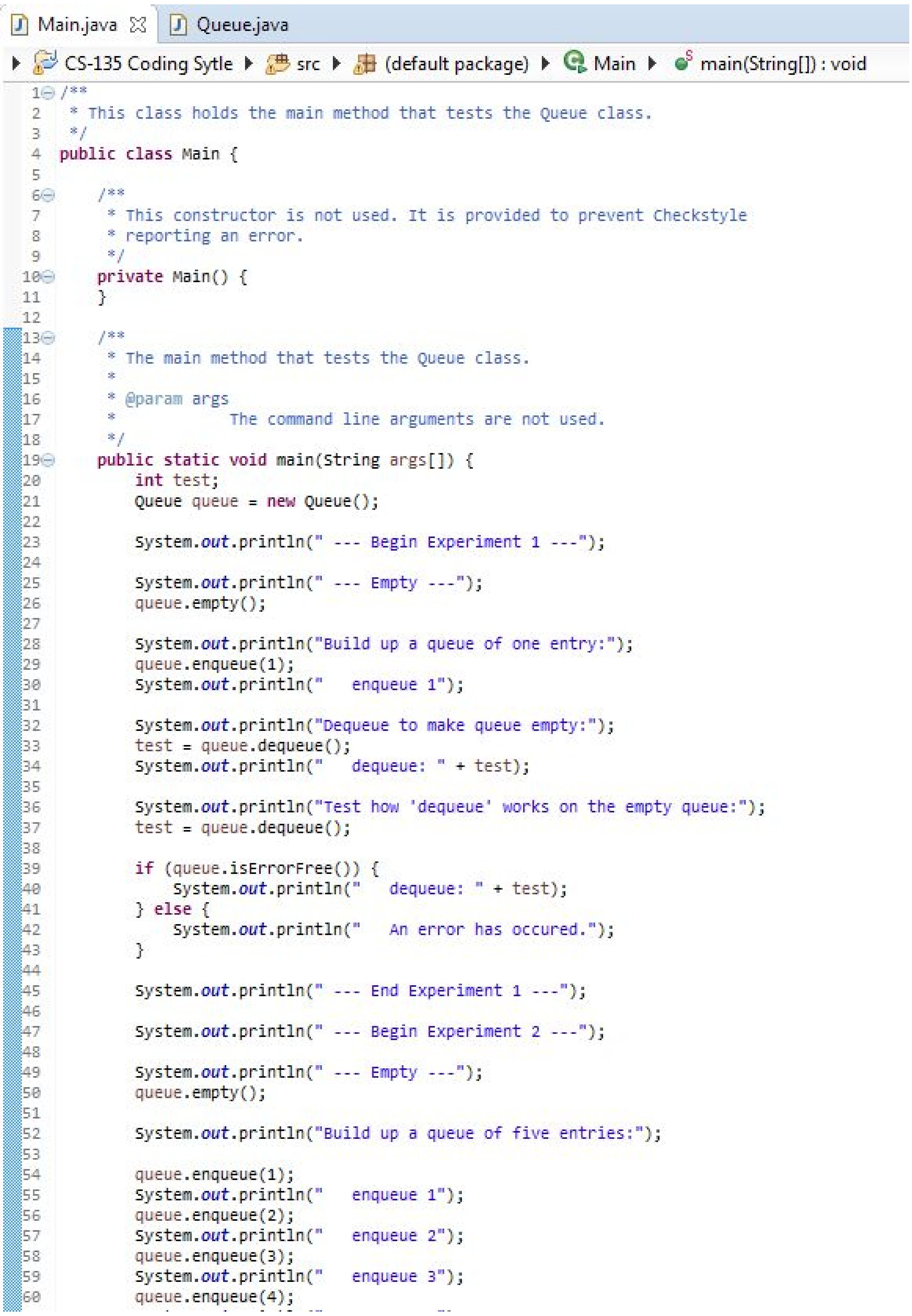
Task 7.1



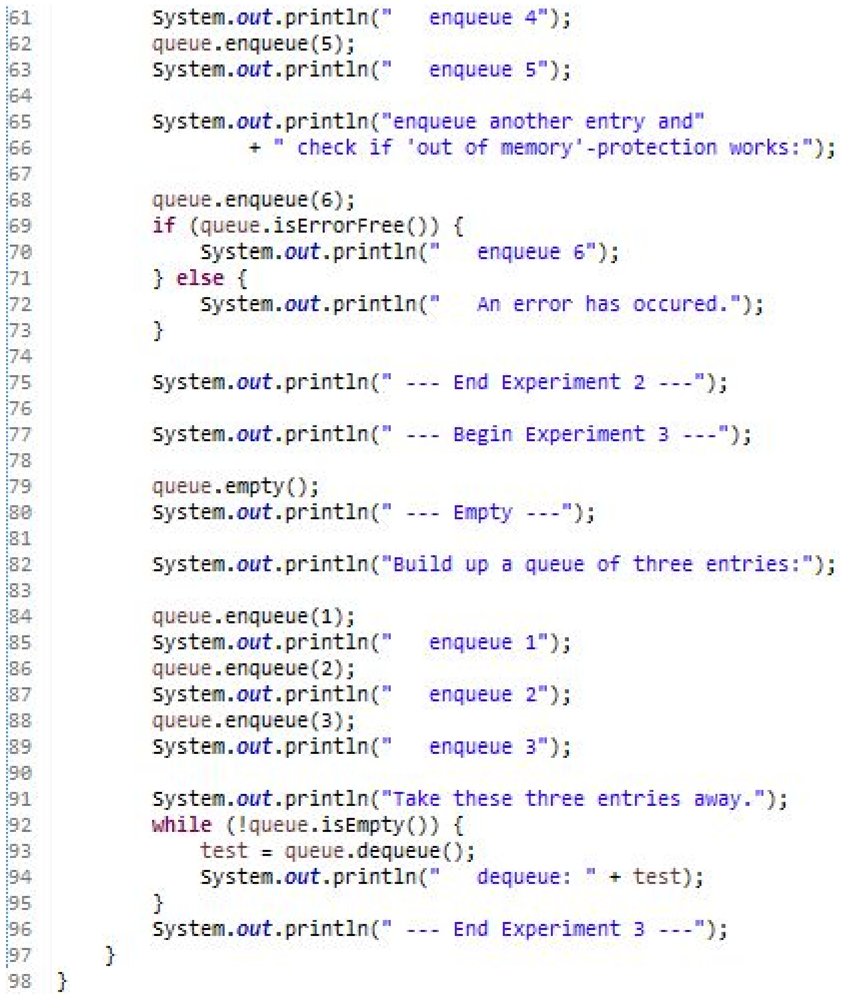
```java
import java.util.Scanner;
public class Sphinx {

    public static int unknown (int x, int y){if (x < y)return x;
        else return y;
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        System.out.print("x: ");
        int x = input.nextInt();
        System.out.print("y: ");
        int y = input.nextInt();
        System.out.println();
        System.out.println("unknown =  " + unknown(x,y));
        System.out.println();
    }
}

```

Task 7.2

Main

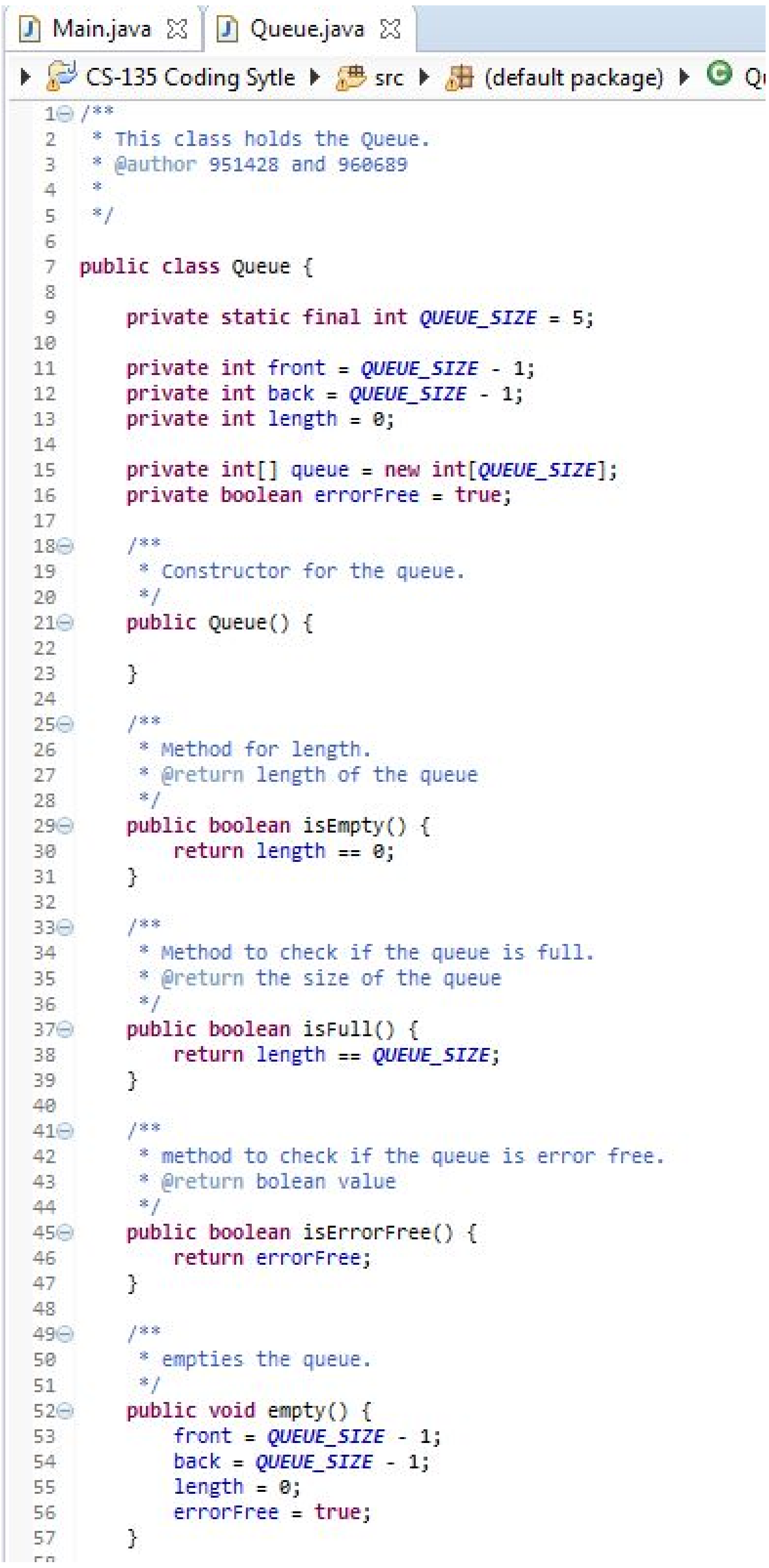


```java
/**
 * This class holds the main method that tests the Queue class.
 */
public class Main {

    /**
     * This constructor is not used. It is provided to prevent Checkstyle
     * reporting an error.
     */
    private Main() {
    }

    /**
     * The main method that tests the Queue class.
     *
     * @param args
     *            The command line arguments are not used.
     */
    public static void main(String args[]) {
        int test;
        Queue queue = new Queue();

        System.out.println(" --- Begin Experiment 1 ---");

        System.out.println(" --- Empty ---");
        queue.empty();

        System.out.println("Build up a queue of one entry:");
        queue.enqueue(1);
        System.out.println("   enqueue 1");

        System.out.println("Dequeue to make queue empty:");
        test = queue.dequeue();
        System.out.println("   dequeue: " + test);

        System.out.println("Test how 'dequeue' works on the empty queue:");
        test = queue.dequeue();

        if (queue.isErrorFree()) {
            System.out.println("   dequeue: " + test);
        } else {
            System.out.println("   An error has occured.");
        }

        System.out.println(" --- End Experiment 1 ---");

        System.out.println(" --- Begin Experiment 2 ---");

        System.out.println(" --- Empty ---");
        queue.empty();

        System.out.println("Build up a queue of five entries:");

        queue.enqueue(1);
        System.out.println("   enqueue 1");
        queue.enqueue(2);
        System.out.println("   enqueue 2");
        queue.enqueue(3);
        System.out.println("   enqueue 3");
        queue.enqueue(4);
```



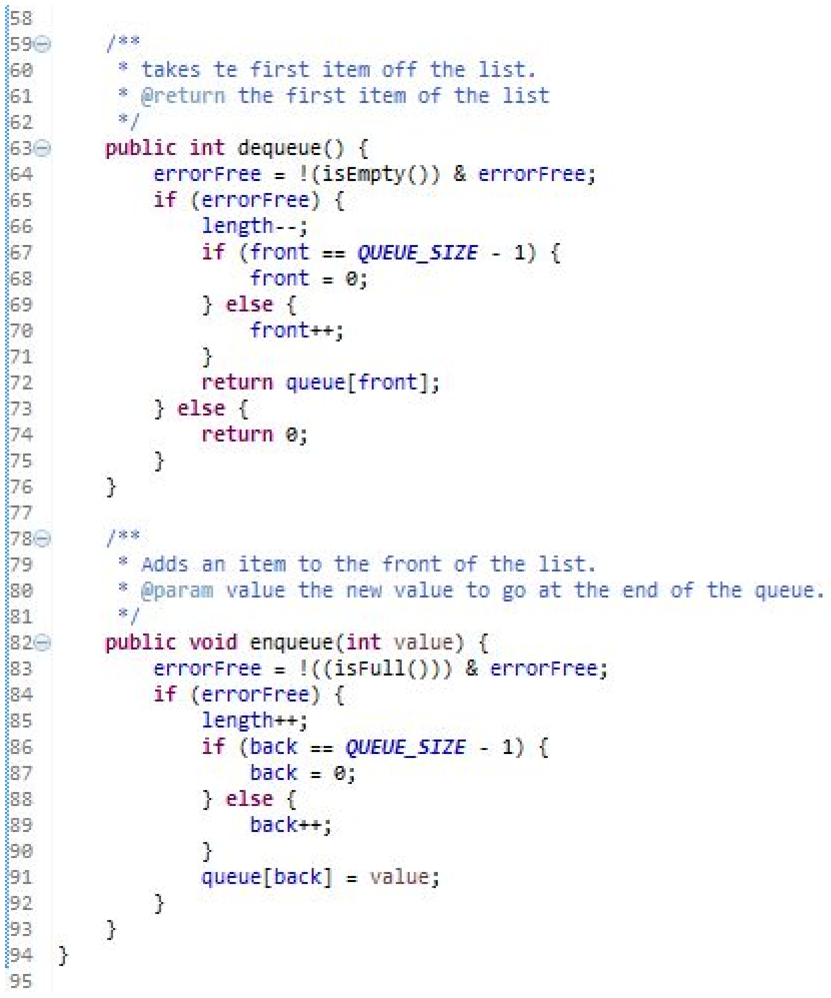
```java
        System.out.println("   enqueue 4");
        queue.enqueue(5);
        System.out.println("   enqueue 5");

        System.out.println("enqueue another entry and"
                + " check if 'out of memory'-protection works:");

        queue.enqueue(6);
        if (queue.isErrorFree()) {
            System.out.println("   enqueue 6");
        } else {
            System.out.println("   An error has occured.");
        }

        System.out.println(" --- End Experiment 2 ---");

        System.out.println(" --- Begin Experiment 3 ---");

        queue.empty();
        System.out.println(" --- Empty ---");

        System.out.println("Build up a queue of three entries:");

        queue.enqueue(1);
        System.out.println("   enqueue 1");
        queue.enqueue(2);
        System.out.println("   enqueue 2");
        queue.enqueue(3);
        System.out.println("   enqueue 3");

        System.out.println("Take these three entries away.");
        while (!queue.isEmpty()) {
            test = queue.dequeue();
            System.out.println("   dequeue: " + test);
        }
        System.out.println(" --- End Experiment 3 ---");
    }
}
```

Queue


Main.java   Queue.java

CS-135 Coding Sytle ▸ src ▸ (default package) ▸ Q

```java
/**
 * This class holds the Queue.
 * @author 951428 and 960689
 *
 */

public class Queue {

    private static final int QUEUE_SIZE = 5;

    private int front = QUEUE_SIZE - 1;
    private int back = QUEUE_SIZE - 1;
    private int length = 0;

    private int[] queue = new int[QUEUE_SIZE];
    private boolean errorFree = true;

    /**
     * Constructor for the queue.
     */
    public Queue() {

    }

    /**
     * Method for length.
     * @return length of the queue
     */
    public boolean isEmpty() {
        return length == 0;
    }

    /**
     * Method to check if the queue is full.
     * @return the size of the queue
     */
    public boolean isFull() {
        return length == QUEUE_SIZE;
    }

    /**
     * method to check if the queue is error free.
     * @return bolean value
     */
    public boolean isErrorFree() {
        return errorFree;
    }

    /**
     * empties the queue.
     */
    public void empty() {
        front = QUEUE_SIZE - 1;
        back = QUEUE_SIZE - 1;
        length = 0;
        errorFree = true;
    }
```



```java

    /**
     * takes te first item off the list.
     * @return the first item of the list
     */
    public int dequeue() {
        errorFree = !(isEmpty()) & errorFree;
        if (errorFree) {
            length--;
            if (front == QUEUE_SIZE - 1) {
                front = 0;
            } else {
                front++;
            }
            return queue[front];
        } else {
            return 0;
        }
    }

    /**
     * Adds an item to the front of the list.
     * @param value the new value to go at the end of the queue.
     */
    public void enqueue(int value) {
        errorFree = !((isFull())) & errorFree;
        if (errorFree) {
            length++;
            if (back == QUEUE_SIZE - 1) {
                back = 0;
            } else {
                back++;
            }
            queue[back] = value;
        }
    }
}

```