# 1.00/1.001

# Introduction to Computers and Engineering Problem Solving

# Fall 2011 - Final Exam

| | |
|---|---|
| **Name:** | |
| **MIT Email:** | |
| **TA:** | |
| **Section:** | |

You have 3 hours to complete this exam. In all questions, you should assume that all necessary packages have already been imported. For coding questions, you do not need to include comments. You may only add code inside the boxes. The code written outside the boxes may not be altered in any way. Good luck!

| | |
|---|---|
| **Question 1** | **/ 10** |
| **Question 2** | **/ 20** |
| **Question 3** | **/ 20** |
| **Question 4** | **/ 30** |
| **Question 5** | **/ 20** |
| **Total** | **/ 100** |

**This page is intentionally left blank.**

## Question 1 – True/False Questions (10 points)

Please **circle your answer** to the following questions.

1). A single stream can be used as both an input stream and an output stream at the same time.
**True** **False**

2). There can be several catch blocks in a single try/catch structure.
**True** **False**

3). A method can throw more than one type of Exception.
**True** **False**

4). In Java, a class can extend any number of abstract classes and an abstract class can have any number of subclasses.
**True** **False**

5). Protected data members cannot be accessed by other classes in the same package.
**True** **False**

6). A final method cannot be overridden.
**True** **False**

7). An event listener can be registered with more than one event source to receive notifications about specific types of events.
**True** **False**

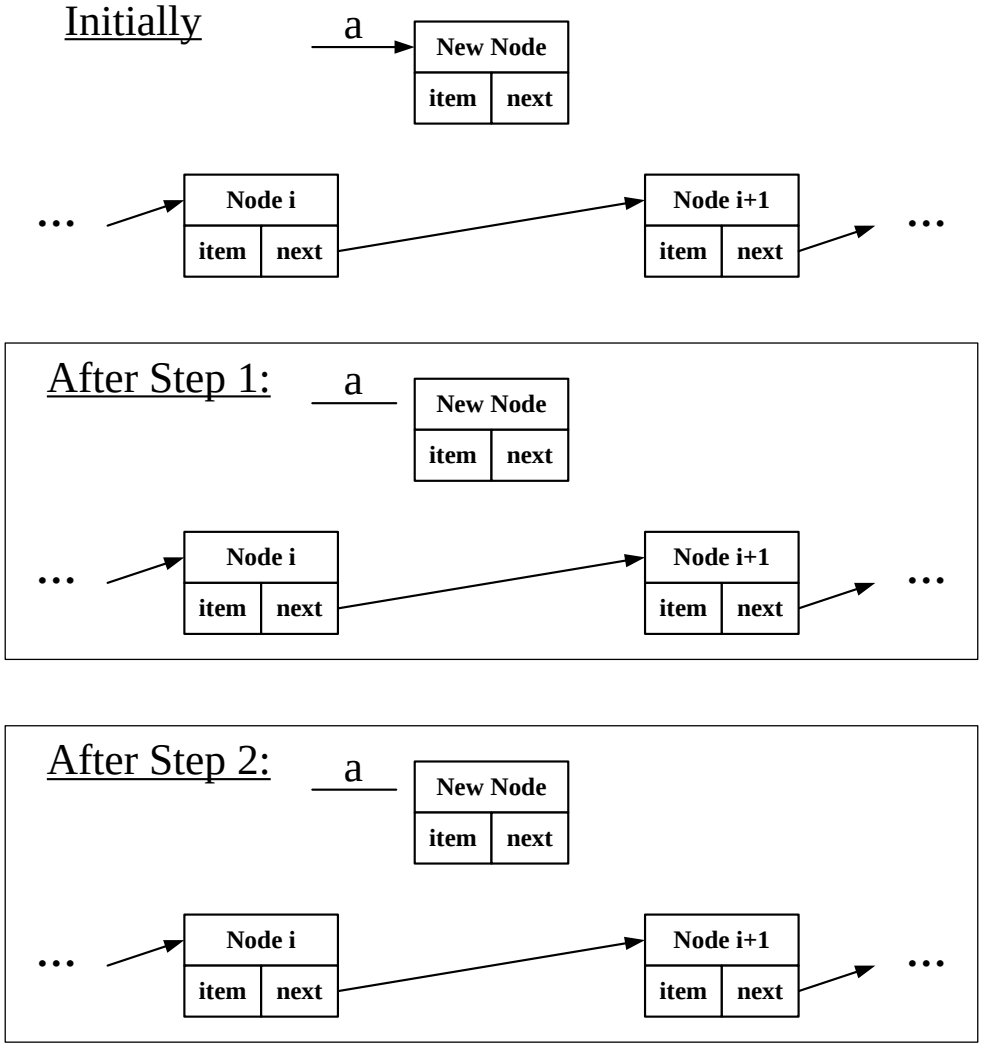
8). Strings are immutable (implicitly final) in Java.
**True** **False**

9). When writing objects to file using ObjectOutputStream, the objects cannot have methods.
**True** **False**

10). A variable declared inside a “for” loop can be referenced outside the loop.
**True** **False**

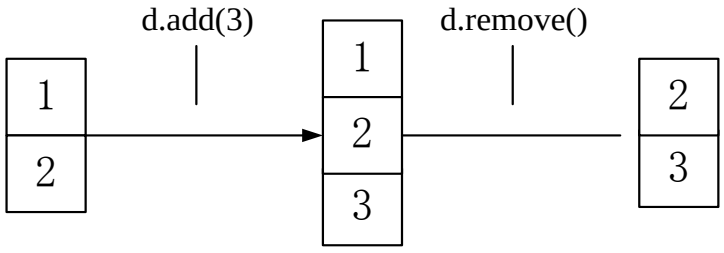
## Question 2 – Data Structures (20 points)

**2.a** Draw links (lines with arrows) to illustrate the process of inserting a node into a linked list between node i and node i+1. Use a cross (X) to denote removing a link. A step is a line of code.




**2.b** Which data structure d is used in the following two operations (add() and remove())?

Please **circle your answer**: **Stack** **Queue** **Array** **ArrayList**

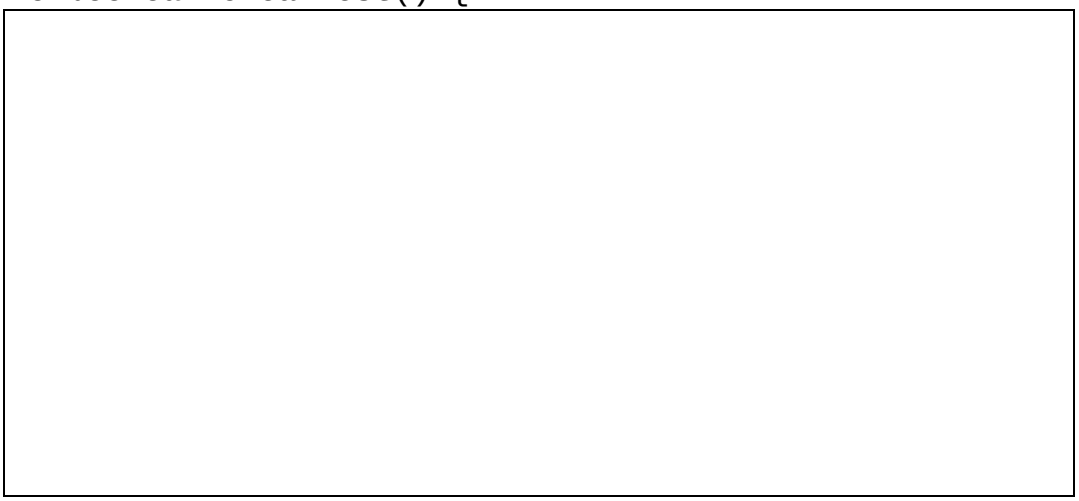


**2.c** Draw the contents of the stack after the following operations are performed. (Show only the contents that can be referenced.)

push("s")
push("e")
pop()
push("e")
pop()
pop()
push("y")
push("a")
pop()

**2.d** In the following `SLinkedList` class, implement method **`public boolean clearRest()`**, which removes all the nodes in the linked list beyond the first one. The method should return **`true`** if nodes were removed, and **`false`** if no nodes were removed. Make sure `first`, `last` and `length` are set correctly. The linked list may be empty.

```
public class SLinkedList {
    private int length = 0;
    private Node first = null;    //the first Node in the list
    private Node last = null;     //the last Node in the list

    private static class Node {
        Object item;
        Node next;
        Node( Object o, Node n ) { item = o; next = n; }
    }
    // Other existing methods here...

    public boolean clearRest() {
```



```
    }
}
```

## Question 3 – Matrices (20 points)

Recall class Matrix, as presented in lecture:

```
public class Matrix {

     private double[][] data;

     public Matrix(int m, int n) {
          data = new double[m][n];
     }

     public int getNumRows() {
          return data.length;
     }

     public int getNumCols() {
          return data[0].length;
     }

     public double getElement(int i, int j) {
          return data[i][j];
     }

     // And many other methods
} // End of Matrix class
```

**3.a Tridiagonal matrix**. A tridiagonal matrix has nonzero elements only in the main diagonal, the first diagonal below it, and the first diagonal above it:

$$\begin{bmatrix} a_{00} & a_{01} & 0 & \cdots & \cdots & 0 \\ a_{10} & a_{11} & a_{12} & \ddots & \ddots & \vdots \\ 0 & a_{21} & a_{22} & a_{23} & \ddots & \vdots \\ \vdots & \ddots & a_{32} & a_{33} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & a_{n-2,n-1} \\ 0 & \cdots & \cdots & 0 & a_{n-1,n-2} & a_{n-1,n-1} \end{bmatrix}$$

Write a method in class Matrix to check if the matrix is tridiagonal. All elements in the main diagonal and those immediately above and below it **must be** nonzero. All other elements **must be** zero. Use a tolerance of $10^{-15}$ to check if an element is zero or not. Assume the matrix is square; you do not have to check.

```
public boolean isTridiagonal() {
```

}

**3.b Graph**. A graph refers to a collection of nodes and a collection of edges that connect pairs of nodes. Graphs are used to model many engineering problems, including communication and transportation networks. Simple graphs can be described by a special type of matrix called an adjacency matrix.

An example graph is shown below. It contains 3 nodes, represented by circles with their index numbers in them. The network also has 3 edges between nodes; the edges allow flow in both directions. The matrix on the right is the node-to-node adjacency matrix for the graph. Every entry $a_{ij}$ refers to the edge between node i and j, where $a_{ij} = a_{ji} = 1$ if the edge exists and 0 if it does not.

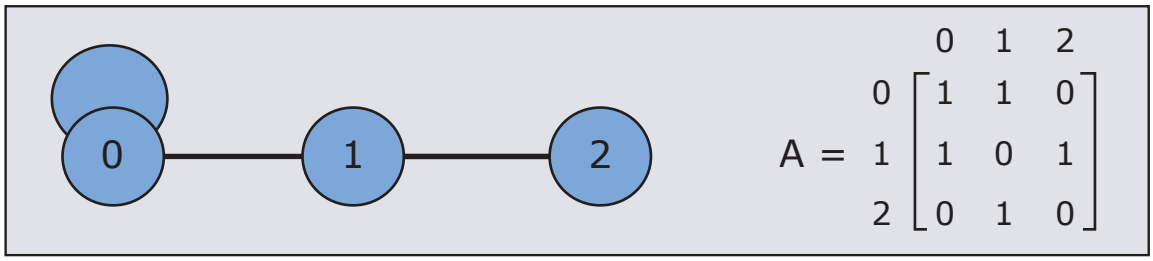



Image by MIT OpenCourseWare.

The node-to-node adjacency matrix of a simple graph is square and symmetric. Each edge is represented twice, once from i to j and once from j to i. Node 0 has a "self loop" in this example.

The Graph class is shown below; it is identical to class Matrix except that it holds ints, not doubles.

```
public class Graph {
     private int[][] data;

     public Graph(int n) {
          data = new int[n][n];
     }
     public int getNumRows() {
          return data.length;
     }
     public int getNumCols() {
          return data[0].length;
     }
     public int getElement(int i, int j) {
          return data[i][j];
     }
     // And many other methods, with ints rather than doubles
} // End of Graph class
```

Write a method `getNumEdges(int j)` for the `Graph` class that returns the total number of edges connected to node j. For example, `getNumEdges(1)` returns 2 for the graph above. Your method should print an error message and return `-1` if argument `j` is invalid (out of range).

```
public int getNumEdges(int j) {
```

```
}
```

## Question 4 – Streams (30 points)

You are given a text file (registration.txt) that contains lines of pairs of student IDs and course names.

2381, 1.001
1812, 8.01T
1812, 1.00
2381, 18.06
9091, 1.204
1230, 1.264
9279, 8.01T
…

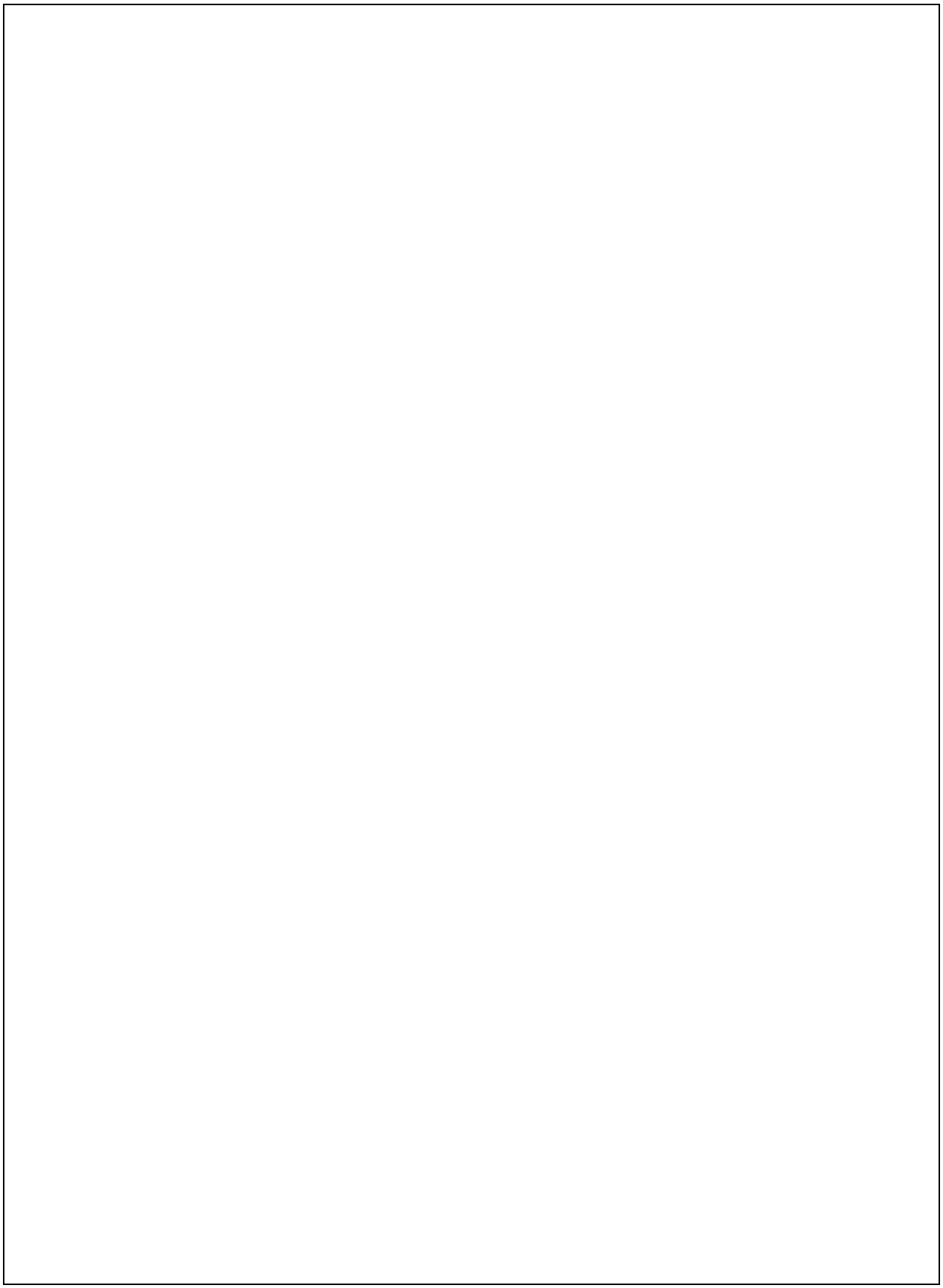
Each line represents a separate instance of a student registration for a particular course. The course is a `String`; the student ID is an `int`. The delimiter is a comma. You do not know the length of the file before you read all of it.

You are to write a program that:

1. reads the file,
2. counts the number of students registered in each course, and
3. prints a list of all courses with their enrollment to the console.

In addition:

1. Your program must handle any potential exceptions.
2. Your program must use an appropriate data structure to keep track of the course registrations, such as an array list, hash map or tree map.
3. You may write all or most of your program in a main() method in your class, if you wish. You may use a second class, if necessary. Give your class(es) descriptive name(s).
4. You do not need to keep track of the courses that each student takes. You only need to compute and output the number of students (enrollment) in each course.



## Question 5 – Phidgets and Traffic Control (20 points)

As part of a research project, you will use Phidgets to implement various traffic signal control policies at 77 Massachusetts Ave.

Currently, there are two signal phases:

1. Phase A: Green signal for traffic flowing on Massachusetts Ave. and red light for pedestrian crossing
2. Phase B: Red signal for traffic flowing on Massachusetts Ave. and green light for pedestrian crossing.

For simplicity we make the following assumptions.

1. You do not need to consider the yellow light phase.
2. Signal phases A and B alternate, with a total cycle time of 90 seconds and cycle split of 2/3 for phase A and 1/3 for phase B. That is, phase A is on for 60 seconds, then phase B is on for 30 seconds, and so on…
3. A push button for the pedestrian crossing will be added. When the button is pushed during phase A and the remaining green time for traffic flowing on Massachusetts Ave. is more than 30 seconds, pushing the button reduces that green time remaining to 30 seconds for the current A phase.
4. If the pedestrian button is pushed when there is less than 30 seconds of green time remaining in phase A, or in phase B, it has no effect.
5. Phase A begins at time = 0 in your program.
6. Assume the touch sensor generates an event **reliably**.

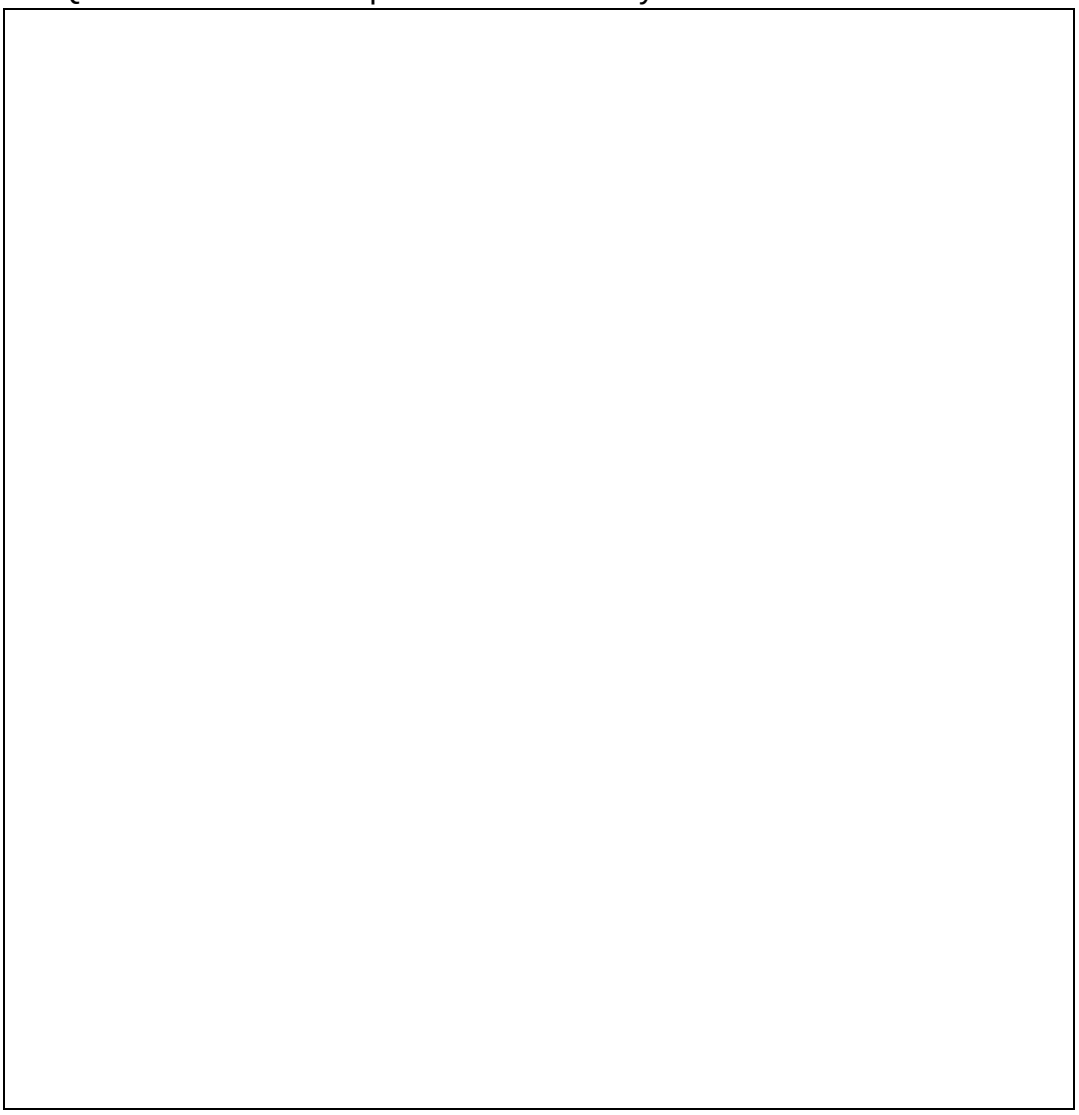
A touch sensor is connected to port 0 on the input side of the Phidgets interface board. Four LEDs are connected to the output ports:

| LED Type | Output Port | Representing |
|---|---|---|
| RED | 0 | Red signal for pedestrians, phase A (don't walk) |
| GREEN | 1 | Green signal for traffic flow, phase A |
| RED | 2 | Red signal for traffic flow, phase B |
| GREEN | 3 | Green signal for pedestrians, phase B (walk) |

Your task is to implement the signal system using Phidgets and Java. Read the code given below and complete the two methods (questions 5a and 5b) so that your program works in accordance to the specification above.

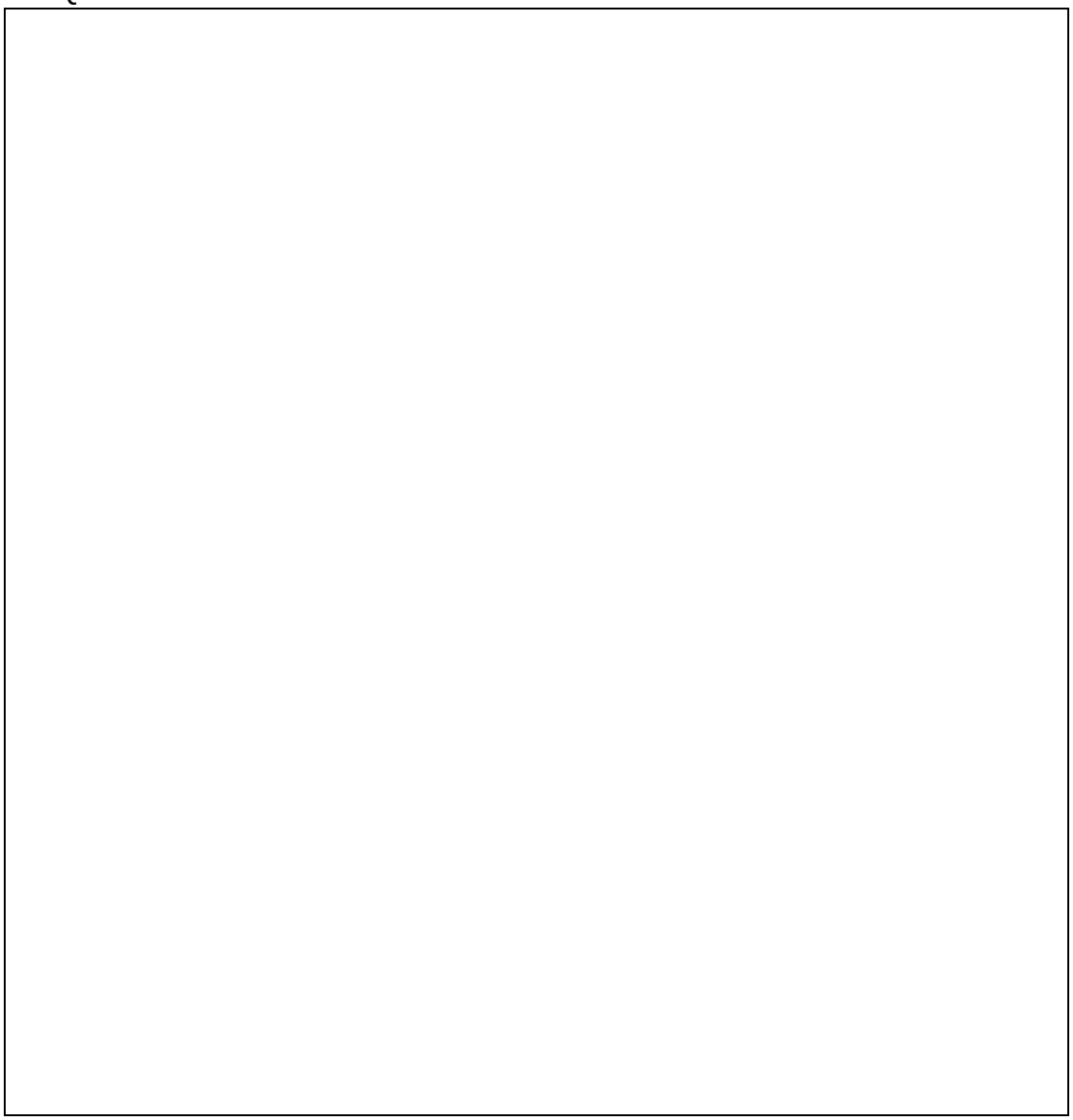
The `Timer tick` triggers an `ActionEvent` every second (1000 milliseconds). Handle exceptions where needed and print to console (`System.out.println()`) if exceptions occur during runtime.

```
public class TrafficControl {
   /* Static members */
   private static InterfaceKitPhidget interfaceKit;
   private static javax.swing.Timer tick;
   private static int counter; // Counts up from 0 seconds

   public static void main (String[] args) throws Exception {
      openIntfcKit();
      tick = new javax.swing.Timer(1000, new ActionListener(){
         public void actionPerformed ( ActionEvent e ) {

            // Question 5a: Complete the anonymous inner class
```



```
      }});
      tick.start();
      // Code omitted to close the interface kit.
   }  // End of main method
   // TrafficControl class continues on next page
```

```
  private static void openIntfcKit() {
    try {
      interfaceKit = new InterfaceKitPhidget();

      // Question 5b: Write the event handler for touch sensor
```



```
      interfaceKit.openAny();
      interfaceKit.waitForAttachment();
      interfaceKit.setRatiometric(true);
      while (!interfaceKit.getRatiometric());
    } catch (PhidgetException pe) {
      System.err.println(pe);
    }
  } // End of openIntfcKit() method
} // End of Traffic Control class
```

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012