UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFORMATICS 1 - OBJECT-ORIENTED PROGRAMMING

Thursday 6th May 2010

09:30 to 11:30

Convener: J Bradfield
External Examiner: A Preece

INSTRUCTIONS TO CANDIDATES

1. **Note that ALL QUESTIONS ARE COMPULSORY.**
2. **DIFFERENT QUESTIONS MAY HAVE DIFFERENT NUMBERS OF TOTAL MARKS. Take note of this in allocating time to questions.**

1. In each of parts (a)–(c) below, you will be asked to supply the body to a method inside a class. There will be a separate class for each part, named `OneA`, `OneB` and `OneC` respectively. You will be given a skeleton file for each of these classes, and the skeleton will contain the appropriate method declaration. You should add your definitions of the methods at the points marked as follows:

   `// ADD CODE HERE`

   (a) Implement the method `boolean inRange(int[] a, int b, int c)` in the class `OneA` (skeleton file supplied). Given an array of `int`s, `a`, and two limits, `b` and `c`, this method should return `true` if and only if every element e of the array is within the limits inclusively; that is, e is not smaller than `b` or greater than `c`. You can assume that the array `a` contains at least one element, and that `b` $\leq$ `c`.

   Expected behaviour:

```
inRange(new int[] {1, 2, 3, 4, 5}, 1, 5) -> true
inRange(new int[] {3, 6, 9}, 1, 5) -> false
inRange(new int[] {3, 5, 4, 3}, 2, 6) -> true
inRange(new int[] {1, 2, 3, 4, 5}, 2, 5) -> false
```

   [*15 marks*]

   (b) Implement the method `int[] copyRange(int[] a, int i, int j)` in the class `OneB`. Given an array of `int`s, `a`, and two *indexes* into the array, `i` and `j`, this should return an array with a copy of the elements that lie between the two indexes inclusively. For example, if `a` is {`4, 3, 2, 4`}, and `j` is 0 and `c` is 2, then the elements between the indexes will be `a[0]` (i.e., `4`), `a[1]` (i.e., `3`) and `a[2]` (i.e., `2`). Every element outside the indices should instead contain `-1`.

   Expected behaviour:

```
copyRange(new int[] {5, 4, 3, 2, 1}, 1, 3)
                              -> {-1, 4, 3, 2, -1}
copyRange(new int[] {5, 11, 3, 9}, 0, 1)
                                -> {5, 11, -1, -1}
copyRange(new int[] {3, 3, 3, 3, 3, 8}, 3, 5)
                           -> {-1, -1, -1, 3, 3, 8}
```

   [*15 marks*]

   (c) Implement the method `int[] remove(int[] a, int b)` in the class `OneC`. Given an array of `int`s, `a`, and an `int b`, this method should return a new array removing all occurrences of `b`. You may have to count how many occurrences of `b` are contained in the array first.

   Expected behaviour:

```
remove(new int[] {1, 2, 3, 4, 5}, 3) -> {1, 2, 4, 5}
remove(new int[] {2, 2, 5, 8, 4, 4, 6, 4}, 4) -> {2, 2, 5, 8, 6}
remove(new int[] {7, 5, 3, 3, 5, 7}, 7 ) -> {5, 3, 3, 5}
remove(new int[] {1, 2, 3, 4, 5}, 6) -> {1, 2, 3, 4, 5}
```

[*14 marks*]

(d) The class `QuestionOneLauncher` (skeleton file supplied) has a single `main()` method. Inside `main()`, add calls to each of the methods you have defined in parts (a)–(c) above to test that your implementations produce the correct results. You can write your tests in any way you think appropriate, but you should have at least two tests for each method. [*6 marks*]

The files that you must submit for this question are the following:

(a) `OneA.java`

(b) `OneB.java`

(c) `OneC.java`

(d) `QuestionOneLauncher.java`

2. This question focuses on the construction of a circuit simulator containing logic gates. The gates correspond to standard boolean connectives such as NOT, AND and OR.

   The circuit simulator is built upon a generic `Circuit` interface which provides the API for all concrete circuits. Implementing this interface are three classes that can be combined to created larger, more complex circuits. The `Input` class is the most basic element of a circuit, always outputting the value for which it is set. The `SingleInputCircuit` class represents a circuit that takes a single output from another circuit, passes this input through a single-input gate, and returns the output from that gate. The `DoubleInputCircuit` class represents a circuit that takes the output from two other circuits, passes these values through a double-input gate, and returns the output from that gate. The design of the simulator is shown in Figure 1.

   You have been provided with the interfaces `Circuit` and `DoubleInputGate`, and the classes `Input`, `OrGate` and `SingleInputCircuit`. In addition, a partially implemented `CircuitTester` is supplied.

   You are required to write the interface `SingleInputGate`, and the `AndGate`, `NotGate` and `DoubleInputCircuit` classes. In addition, you will need to implement some methods in `CircuitTester`. The task is broken down into more detail below, beginning with the class `AndGate`.

   (a) Write the class `AndGate`. This gate's `apply()` method must return `true` if both inputs are `true`, otherwise it must return `false`. [*5 marks*]

   (b) Write the interface `SingleInputGate`. You should be able to infer its methods from Figure 1. In addition, write the class `NotGate` which implements `SingleInputGate`. This gate must invert its input. That is, a *true* input must return a *false* output, and a *false* input voltage must return a *true* output. [*10 marks*]

   (c) Write the class `DoubleInputCircuit`. This task is broken down into more details below:

      (i) Create instance variables to store the gate that the circuit will be using, and the circuits that this class will be using as input to the gate.

      (ii) Write the constructor for this class, as specified in Figure 1, so as to initialize the relevant instance variables.

      (iii) Write the body of the method `output()`, which takes the output from the sub-circuits, passes them through the gate and returns the output from the gate. [*15 marks*]

   (d) The class `CircuitTester` (provided) contains a `main()` method and a `TestOr()` method, along with empty method declarations for `TestAnd`, `TestNeg` and `TestDblNeg`. Taking `TestOr` as a model, provide bodies for the methods

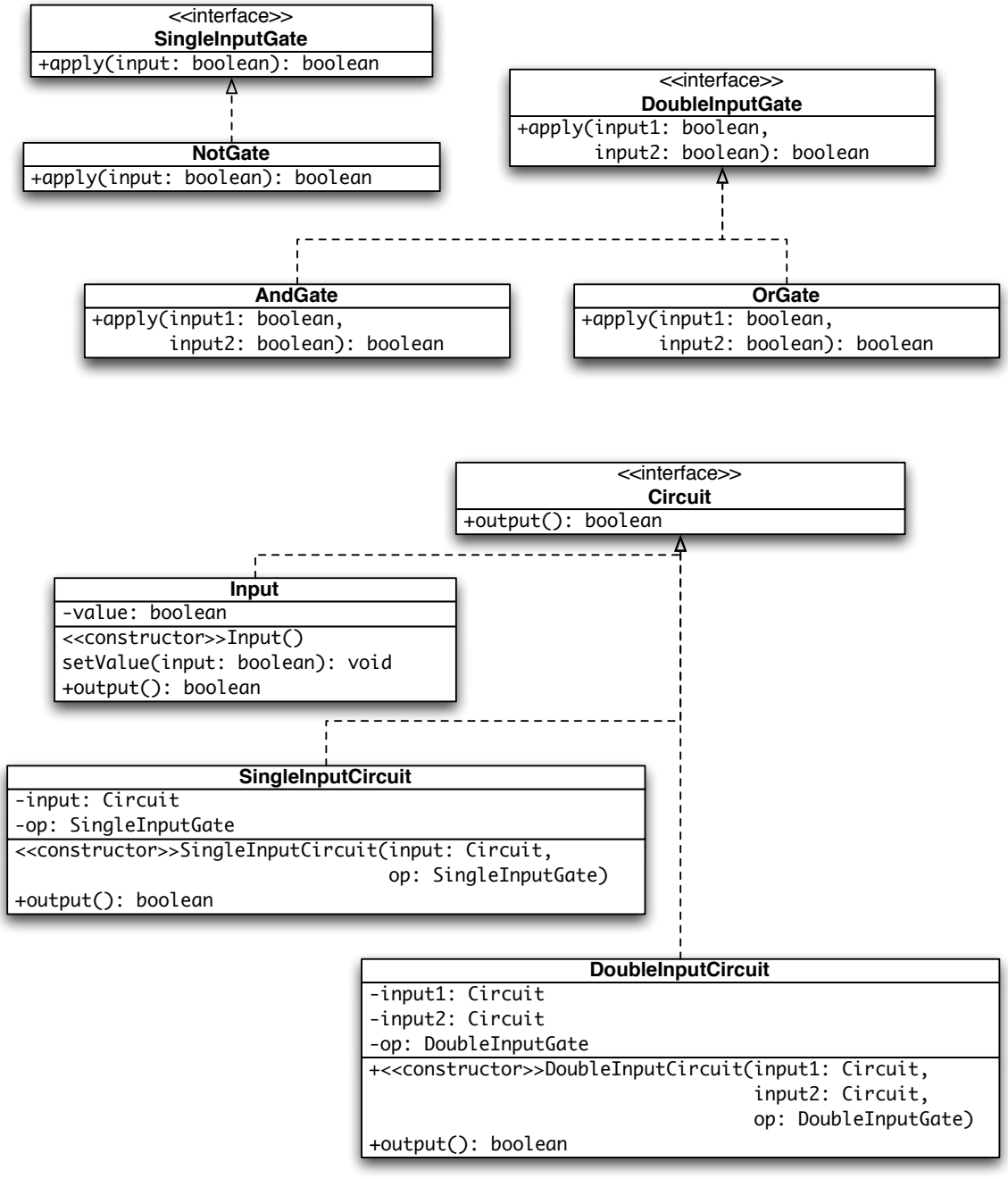

Figure 1: UML Diagram of the Circuit Simulator System

- `TestAnd`;
- `TestNeg`; and
- `TestDblNeg` — this method should create two `SingleInputCircuits` that use `NotGates`, and should use the output from one of the circuits as the input to the other.

[*15 marks*]

(e) This task involves adding a new method `TestXor()` to the class `CircuitTester`. Use a combination of `Circuit`s and `Gate`s to model a circuit which implements the boolean function XOR. Here is the truth table for XOR:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

[*5 marks*]

The files that you must submit for this question are the following:

- `AndGate.java`
- `SingleInputGate.java`
- `NotGate.java`
- `DoubleInputCircuit.java`
- `CircuitTester.java`

Final Checklist

Here is a complete list of all the files required for this exam:

```
OneA.java
OneB.java
OneC.java
QuestionOneLauncher.java
AndGate.java
SingleInputGate.java
NotGate.java
DoubleInputCircuit.java
CircuitTester.java
```