L0101/L0102/L0103 Midterm 2002

CSC 148H
St. George Campus

Duration — 50 minutes

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last Name: _____

First Name: _____

TA: _____

Section (circle one): Gries (0101) Jepson (0102) Chechik (0103)

---

*Do **not** turn this page until you have received the signal to start.*
Read this entire page or you'll miss the bonus question.

Midterm aids allowed:

- the Java API Reference booklet

Bonus question: if you write your student ID at the top of pages 2–5, you will get an extra mark.
Please write legibly.
If you run out of space on a question, use the back of the page.

1: _____/10

2: _____/10

3: _____/10

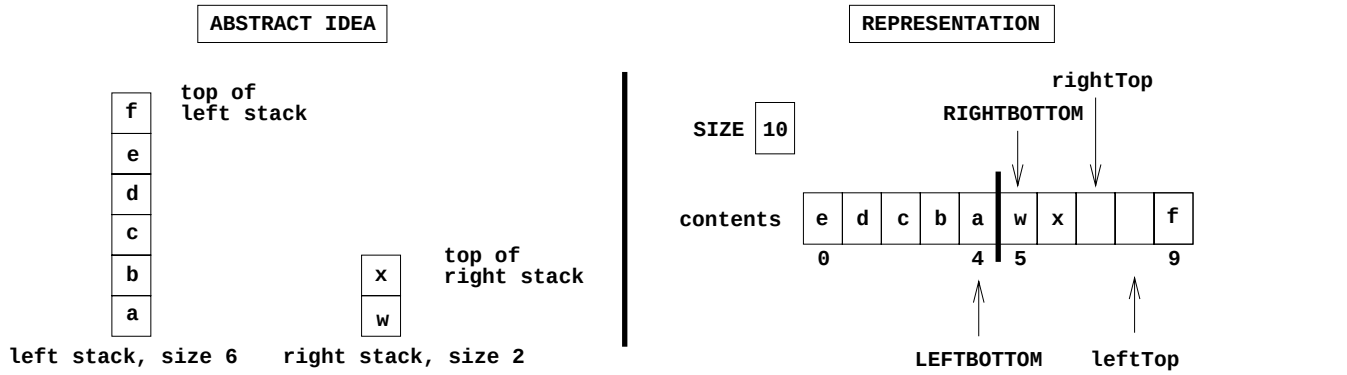
BONUS: _____/ 1

TOTAL: _____/30

Good Luck! PLEASE HAND IN

Question 1. [10 MARKS]

Recall that a stack grows and shrinks from only one end, like a stack of plates. Below on the right is a picture of two stacks stored in a single array. The bottoms touch, the left stack grows to the left, and the right stack grows to the right. In the example, the left stack has "wrapped around" the end of the array.




```
public class TwoStack {
    final int SIZE=10;                // The size of the array.
    final int LEFTBOTTOM=SIZE/2-1;    // The index of the bottom of the left stack.
    final int RIGHTBOTTOM=SIZE/2;     // The index of the bottom of the right stack.
    private Object[] contents=new Object[SIZE]; // The items in the two stacks.
    private int leftTop=LEFTBOTTOM;   // The index one above the top of the left stack.
    private int rightTop=RIGHTBOTTOM; // The index one above the top of the right stack.
    /** Push o onto the right stack. */
    public void pushRight(Object o) throws StackFullException { /* Body not shown. */ }
    /** Push o onto the left stack. */
    public void pushLeft(Object o) throws StackFullException { /* Body not shown. */ }
    /** Return the number of items in the right stack. */
    public int rightSize() {

SOLUTION:
       return (rightTop - RIGHTBOTTOM + SIZE) % SIZE;
    }
    /** Return the number of items in the left stack. */
    public int leftSize() {

SOLUTION:
       return (LEFTBOTTOM - leftTop + SIZE) % SIZE;
    }
```

Part (a) [6 MARKS] Write the bodies of `rightSize` and `leftSize`.

Part (b) [4 MARKS] Under what conditions should `pushLeft` throw a `StackFullException`?

```
SOLUTION:
    Several solutions are possible here.  The easiest is realizing that
no harm is done if we keep one slot unoccupied.  Thus, StackFullException
can be thrown when leftTop == rightTop or leftSize() + rightSize() == SIZE - 1.

    A better solution is to realize that we can use all slots of contents[]
provided that neither stack is empty.  Otherwise, we are unable to detect
the difference between an empty stack and a full stack.  Thus, StackFullException
can be thrown by pushLeft when
     (rightSize() == 0 AND leftSize() == SIZE - 1 )   OR
     (rightSize() != 0 AND rightSize() + leftSize() == SIZE)
```

Question 2. [10 MARKS]

The following line of code throws a `FileNotFoundException` if `s` is not a valid file name.

```
FileReader f= new FileReader(s);
```

The following typecast throws a `ClassCastException` if `o` does not refer to a `String`.

```
String s= (String)o;
```

Complete the method below so that it meets its contract. You may not use `if` statements. You may assume that all four parameters are not `null`; you don't have to check for that, and that is not the point of this question.
Hint: in the `try` block, write the code assuming that everything in `v` is a valid file name, and use `catch` blocks to deal with the other cases.

```
/**
 * v contains Strings, as well as other types of objects.
 *
 * Find the Strings in v, create FileReader objects from them, and
 * store those FileReaders in frs.
 *
 * Some Strings will not correspond to valid file names, and thus the
 * FileReader constructor will throw a FileNotFoundException. Store all
 * the FileNotFoundExceptions in exc.
 *
 * Store into objs every object in v that is not a String.
 */
public void partitionFileNames(Vector v, Vector frs, Vector exc, Vector objs) {
    Iterator i= v.iterator();
    while (i.hasNext()) {
        Object next= null; // The next item in the iteration.
        try {

SOLUTION:
            next= i.next();
            frs.add(new FileReader(((String)next)));
        } catch (ClassCastException e) {
            objs.add(next);
        } catch (FileNotFoundException e) {
            exc.add(e);
        }
```

Question 3. [10 MARKS]

```
public class M {
  public static int z;
  public M n;

  public void w(M e) {
    M p = this;
    while(p.n != this)
      p = p.i(p, e);
    p.i(p, e);
  }

  public M i(M p, M e) {
    e.n = p.n;     // Line 1
    p.n = e;
    return e.n;
  }

  public static void main(String[] s) {
    M a = new M();
    a.n = a; a.z = 99;
    M b = new N(5);       // Line 3
    a.n = b; b.n = a;
    a.w(new N(7));        // Line 5
    // ...
  }
}
```

```
public class N extends M {
  public int z;
  public M n;

  public N(int n) {
   z = n;
  }

  public M i(M p, M e) {
    return p.n;
  }
}
```

On the next page we have drawn a correct memory model for the point at which the above program reaches the beginning of line 3 in the main method.

Continue to trace this program until you first reach "Line 1" of method `i` in class `M`, which happens during the call to method `w` on "Line 5" of the main method. We recommend that you use the space below for any scratch work (we will NOT mark it), and then copy your final answer onto the picture on the next page.

Question 3. (CONTINUED)