

CSC108H1 F October Midterm 2002

Duration — 50 minutes

Aids allowed: none

Student Number: _____

Lab day, time, room: _____

Last Name: _____

First Name: _____

Lecture Section: _____

Instructor: _____

*Do **not** turn this page until you have received the signal to start.*

*(Please fill out the identification section above,
and read the instructions below.) Good Luck!*

This midterm consists of 3 questions on 5 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.* Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.

1: _____/10

2: _____/10

3: _____/10

Write your student number at the bottom of pages 2-5 of this test.

If you use any space for rough work, please indicate clearly what you want marked.

TOTAL: _____/30

	<u>Abbreviation</u>	<u>Stands for</u>
To the right are abbreviations the you may use. You must write everything else in full.	S.o.p	System.out.print
	S.o.pln	System.out.println
	I.pl	Integer.parseInt
	JOP.sID	JOptionPane.showInputDialog

Short Java API descriptions (all methods are public):

class Integer:

static int parseInt(String s) // = s's value, as an int.

class Double:

static double parseDouble(String s) // = s's value, as a double.

class Boolean:

static boolean parseBoolean(String s) // = s's value, as a boolean.

class String:

String substring(int i, int j) // = the letters between i (inclusive) and j (non-inclusive).

String substring(int i) // = the letters from i (inclusive) to the end.

int indexOf(String s) // = the index of s in this String; -1 if s is not a substring.

int indexOf(String s, int i) // = index of s in this String after index i; -1 if s not found.

int length() // = the number of letters in this String.

class JFrame:

JFrame() // An empty window with no title, not visible on the screen.

JFrame(String s) // An empty window title s, not visible on the screen.

Container getContentPane() // this JFrame's content pane.

int getWidth() // this JFrame's width.

int getHeight() // this JFrame's height.

int getX() // this JFrame's horizontal coordinate.

int getY() // this JFrame's vertical coordinate.

void setSize(int w, int h) // set this JFrame's size to w wide and h high.

```

void setTitle(String s) // set this JFrame's title to s.
void setLocation(int x, int y) // set this JFrame's location to (x, y).
void show() // make this JFrame visible.
void hide() // make this JFrame invisible.
void pack() // resize this JFrame according to its content pane's contents.
class JOptionPane:
    static String showInputDialog(String m) // get input from the user, prompting with m.
class JTextArea:
    JTextArea(int r, int c) // text area with r rows and c columns, and no initial text.
    JTextArea(String s, int r, int c) // text area with r rows, c columns, and initial text s.
class JButton:
    JButton() // A button with no name.
    JButton(String s) // A button named s.
    String getText() // = this JButton's name.
class Container:
    // add item i in location specified by loc, which is "North", "South", "East", "West", or
    // "Center". For example, add(new JButton("A"), "East") adds a new JButton in the east.
    add(Component i, String loc)

```

Question 1. [10 MARKS]

Read the following class outline thoroughly. On the next page is a constructor; fill in the body of the constructor so that it meets its specification.

We have intentionally left the description of what the user is asked to type vague (in places): what you ask the user and what they will type are not always completely specified, although when we give you a format you are expected to follow it.

If you can't do some of this question, do the parts that you can. If you do this, please identify the areas that you know are incomplete; provide English descriptions of what you want to do for part marks.

You may assume that all user input will be correct.

```

import javax.swing.*;
import java.awt.*;

/** A window with up to two buttons and a text area. */
public class CustomJFrame extends JFrame {

    /** The first button. The text is "b1". */
    private JButton b1;

    /** The second button, if it exists. The text is "b2". */
    private JButton b2;

    /** Text area, in the center. */
    JTextArea ta;

    // Methods, including the constructor, are not shown.

    // The constructor (which you have to write) is on the next page.

}

```

```
/** Perform the following actions:
 *
 * 1. Ask the user for the initial content of the text area. Make the text area 10 x 20.
 *
 * 2. Ask the user how many buttons to create (1 or 2) and their location(s), all in one
 * line. The format is the number of buttons, followed by a space, followed by the
 * locations. For example, "2 North West" specifies to create 2 buttons, the first in
 * the north and the second in the west.
 *
 */
public CustomJFrame() {
    Container c = this.getContentPane(); // add the buttons and the text area to c.
```

```
} // End of constructor
```

Question 2. [10 MARKS]

Show the output if `Q.do()` is called, writing your answer below each call to `println`.

```

public class Computer {
    private int memory;
    private String type;
    public Computer(String t, int m) {
        type = t;
        memory = m;
    }
    public int getMemory() {
        return memory;
    }
    public void upgradeMemory(int m) {
        memory = memory + m;
    }
}

public class Q {
    public static void do() {
        boolean a= false;
        boolean b= false;
        System.out.println("a == b: " + (a == b));

        a= b;
        b= true;
        System.out.println("a: " + a + " b: " + b);

        Computer d= new Computer("Mac", 0);
        Computer e= d;
        e.upgradeMemory(256);
        e= new Computer("Mac", 128);
        Computer f= e;
        e.upgradeMemory(64);
        System.out.println("d: " + d.getMemory() +
            " e: " + e.getMemory() +
            " f: " + f.getMemory());

        d= new Computer("Mac", 128);
        e= new Computer("Mac", 128);
        System.out.println("d == e: " + (d == e));

    }
}

```

Question 3. [10 MARKS]

In this question, you will develop part of class `Student`, which describes simple student records in a university database. You do *not* need to write comments, but your method names and instance and class variable names must be descriptive. *Use the back of the page if you need more room.*

A student has the following features; declare instance or static variables for them, and declare any other instance or static variables that you will need.

grade point average (GPA): the grade point average; a real number between 0 and 4;

student number: the unique id; an integer (*see below*).

personal information: an instance of class `Person` (assume `Person` is already written).

To create a `Student`, the user (the registrar) must specify the *personal information*. The *student number* must be generated automatically and be the integer immediately following the number of the last `Student` created. The number given to the very first student in the database shall be `888444333`. Every student shall start out with a GPA of zero. Write the `Student` constructor.

Common operations on students include *checking student number*, *updating GPA* to a new given value, and *checking the student's personal information*. Write the full method (including the method body) for *checking the student's personal information*. Provide method headers for the rest of the methods (leave the bodies blank).

```
public class Student {
```

Total Marks = 30