UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER EXAMINATIONS 2003

CSC 108H1 F

St. George and Erindale Campuses

Duration — 3 hours

Aids allowed: None.

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

Last Name: ____________________

First Name: ____________________

Lecture Time (circle one): 9am 10am 1pm 2pm 6pm

Instructor (circle one): Jeremy Sills Paul Gries Steve Engels

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

This examination consists of 4 questions on 14 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.* If you need more space for one of your solutions, use one of the blank pages near the end of the exam and *indicate clearly the part of your work that should be marked.*
Unless otherwise specified, you are allowed to write helper methods and comments are not required. However, comments can be helpful when assigning part marks.
Below are the abbreviations that you may use:

Abbreviation	**Stands for**
`S.o.p`	`System.out.print`
`S.o.pln`	`System.out.println`
`I.pI`	`Integer.parseInt`
`D.pD`	`Double.parseDouble`
`JOP.sID`	`JOptionPane.showInputDialog`
`BR`	`BufferedReader`
`BR.rL`	`BufferedReader.readLine`

# 1:	______/22
# 2:	______/20
# 3:	______/30
# 4:	______/18
TOTAL:	______/90

Good Luck!

Question 1. [22 MARKS]

Part (a) [2 MARKS]

Write your student number in the space provided at the bottom of every page of this exam.

Part (b) [20 MARKS]

On the opposite page some of the code fragments may have errors. For each fragment:

- If the fragment has a compile error then place a checkmark $\surd$ in the first column.
- If the fragment compiles but has a run-time error then place a checkmark $\surd$ in the second column.
- If there is no error then in the third column briefly describe the result of executing the fragment. If there is an error then *do not write in the last column.* If you know that there is no error but you are unsure of the result then place a checkmark $\surd$ in the last column.

You should assume the following:

- All necessary `import` statements have been done.
- All classes are in separate files.
- All code fragments appear in separate classes and methods.

Code fragment	**Compile error?**	**Run-time error?**	**Result, if successful?**
int i = 23.0;			i's value:
String a = "" + 4 + 5;			a's value:
double d = (int) 4.7;			d's value:
double d = (double) "50";			d's value:

String s = new String("63487");
s = s.substring(1, 3)
      + s.substring(4, 5);
			s's value:

String t = "Fred";
t = t.substring(5);
			t's value:

boolean b;
int j = 5;
if (0 < j < 10) {
 b = true;
} else {
  b = false;
}
			b's value:

String s = br.readLine();
while (s != "done") {
  System.out.println(s);
  s = br.readLine();
}
			brief description of output:

int[][] k = {
    {5, 7, 8},
    {3, 1, 9},
    {6, 4, 2}
  };
 	
int total = 0;
for (int m = 0; m < k[0].length; m++) {
  total = total + k[m];
}
			total's value:

public class Animal {
}
 	
public class Insect extends Animal {
}
 	
Insect i = new Insect();
Animal a = i;
			What is the value of the expression a.equals(i)?

Question 2. [20 MARKS]

Part (a) [8 MARKS] Rotate your exam and complete the following code. Comments are not necessary.

```
import java.util.*;

/**
 * Stores the data for an appointment: the day and
 * time of the appointment and a one-line description.
 */
public class Appointment {

  // Here, declare any instance variables needed to
  // store the appointment data:







  /**
   * An Appointment described by s occurring at time d.
   * @param d The time and day this appontment occurs.
   * @param s Description of appointment.
   */
  public Appointment(Date d, String s) {






  }




// Appointment is continued in the next column
```

```

  /**
   * Return when this appointment occurs.
   * @return when this appointment occurs.
   */
  public Date getDate() {




  }

  /**
   * Set this appointment's day and time to d.
   * @return Date d the new day and time.
   */
  public void setDate(Date d) {




  }

  /**
   * Return this appointment's description.
   * @return this appointment's description.
   */
  public String getDescription() {




  }
}  // End of class Appointment
```

Part (b) [12 MARKS] Complete the following code. Comments are not necessary.

```
import java.util.*;

/** Maintain a list of appointments for a single day. */
public class DaySchedule {

  // Here, declare appropriate variables to store the list
  // of appointments and the number of appointments:






  /**
   * A new DaySchedule that can hold n Appointments.
   * @param n the maximum number of appointments in this day.
   */
  public DaySchedule(int n) {





  }

  /**
   * Add appointment app. The appointment cannot be added if there is another
   * appointment already scheduled at the same day and time.
   * @param app the new Appointment
   * @return Whether the appointment was added.
   */
  public boolean addAppointment(Appointment app) {











  }

// DaySchedule is continued on the next page.
```

```
  /**
   * Retrieve the appointment starting at time d.
   * @param d the Date of the appointment to return.
   * @return The appointment starting at time d, or null if there is no
   * such appointment.
   */
  public Appointment getAppointment(Date d) {

  }

  /**
   * Change the day and time of a to newT. The appointment cannot
   * be changed if there is another appointment at the new time.
   * @param a the appointment to change.
   * @param newT the new time of the appointment.
   * @return Whether the change was successful.
   */
  public boolean changeAppointment(Appointment a, Date newT) {

  }
} // End of DaySchedule
```

Question 3. [30 MARKS]

Part (a) [2 MARKS]

Write a statement that declares a variable that can store any of the following values: `2.0`, `99.5`, `-2048.333`.

Part (b) [2 MARKS]

Write a statement that declares a suitable variable and stores the value `"Fred"` in it.

Part (c) [2 MARKS]

Assume a `double` variable named `mark` has been declared and given a value. Write an expression that evaluates to `true` if the value in `mark` is a passing mark in a university course, and `false` otherwise.

Part (d) [2 MARKS]

Assume a `String` variable `s` is initalized with a `String` with a length greater than 3. Write an expression that produces a `String` containing the last 3 characters in `s`.

Part (e) [2 MARKS]

Write a single statement that declares an array variable called `list` and intializes it to refer to an array that can hold 22 `String` objects.

Part (f) [4 MARKS]

Assume `low` and `high` are `int` variables and they are initialized. Assume `low < high`. Write a loop that prints the integer values between `low` and `high` one value per line, including `low` and `high`.

Part (g) [6 MARKS]

Assume a class called `Matrix` has one instance variable: a two-dimensional integer array called `intList`. Write a `Matrix` instance method called `equals` that has one `Matrix` parameter and returns `true` if the contents of the integer arrays are equal, and `false` otherwise.

Part (h) [6 MARKS]

Assume that a class `C` has one instance variable: a 1-dimensional array named `list` that can hold objects of class `T`. Assume class `T` includes an `equals` method that returns `true` if a pair of variables refer to the same object, and `false` otherwise. Write an instance method called `allSame` that returns `true` if all the objects in `list` refer to the same object and `false` otherwise. You should assume that the array is not `null`, but items in it may be. Your method should be as efficient as possible: only process until the answer is known.

Part (i) [2 MARKS]

Assume class `C` in the previous question has a constructor that takes an `int` specifying how big to make the array. Write a statement that declares a variable of type `C` and initializes it to refer to a new instance of `C` that can hold 122 items of class `T`.

Part (j) [2 MARKS]

Assume the declaration and initialization statement in the previous part. Write a statement that declares a new variable and sets it to the result of calling the `allSame` method.

Question 4. [18 marks]

Part (a) [12 marks]

Write a `static` method called `sort` that takes a `String[]` and sorts it according to the following general picture:

```
---------------------------------------------------------------------------
|  smallest items, sorted  |   unknown section  |  largest items, sorted  |
---------------------------------------------------------------------------
```

On each iteration you must find the smallest and largest items in the unknown section and move them to the appropriate ends of that unknown section.

For example, if this is your array after 3 iterations:

```
-------------------------------------------------
| 0 |  1 |  3 |   unknown section   | 7 | 8 | 9 |
-------------------------------------------------
```

Then on the next iteration your code should find the smallest and largest items in the unknown section and move them to the spot right after the 3 and the spot right before the 7, respectively.

Part (b) [6 MARKS]

Write a set of JUnit `test` methods for your `sort` method, including comments. (You don't need to use any `@` tags in your comments, just describe what you are testing and why. You also don't need to write the class header or any imports: just write the `test` methods.)

This page is provided for rough work and any answers that didn't fit.

This page is provided for rough work and any answers that didn't fit.

Short Java API descriptions (all methods are public)

```
Object:
  boolean equals(Object o) // = "this Object is equal to o"
  String toString() // = a String representation of this Object
Integer:
  static int parseInt(String s) // = s's value, as an int
Double:
  static double parseDouble(String s) // = s's value, as a double
Boolean:
  static boolean parseBoolean(String s) // = s's value, as a boolean
String:
  String substring(int i, int j) // = the letters from i (inclusive) to 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 after index i (inclusive); -1 if s not found
  int compareTo(Object o) // = < 0, 0, or > 0 depending on whether this < o, = o, or > 0.
  int length() // = the number of characters in this String
  boolean equals(String s) // = this String has the same contents as s
JFrame:
  JFrame() // An empty window with no title, not visible on the screen
  JFrame(String s) // An empty window with 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
JOptionPane:
  static String showInputDialog(String m) // get input from the user, prompting with m
  static void showMessageDialog(Component c, Object message) // alert the user, with m
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
  void setText(String s) // set the text in this text area to s
  void append(String s) // append s to the text in this text area
  String getText() // = the text in this text area
JButton:
  JButton() // A button with no name
  JButton(String s) // A button named s
  String getText() // = this JButton's name
BufferedReader:
  BufferedReader(Reader in) // Create a reader reading from 'in'
  // Return the next available line in the BufferedReader, or null if at end
  String readLine()
FileReader:
  FileReader(String f) // Create a reader reading from a file named f
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)
Math:
  static double abs(double a) // = the absolute value of a
  static double pow(double a, double b) // = a^b
  static double min(double a, double b) // = minimum of a and b
  static double max(double a, double b) // = maximum of a and b
Vector:
  // add o at index i, shifting this[i..] to make room; always return true
  void add(int i, Object o)
  boolean add(Object o) // add o at the end; always return true
  void clear() // remove all elements
  boolean contains(Object o) // = "o is an element of this Vector"; uses o.equals
  Object get(int i) // = the element at index i
  // = the index of the first occurrence of o; -1 if none; uses o.equals
  int indexOf(Object o)
  int size() // = the number of items in this Vector
  boolean remove(int i) // remove the element at index i
  // remove the first occurrence of o (if present); return true iff o was removed;
  // uses o.equals
  boolean remove(Object o)
  Iterator iterator() // return an Iterator of the items in this Vector
  // return a String representation of this Vector in the form [e1, e2, ... eN]
  String toString()
Iterator:
  boolean hasNext() // = "there are more elements to return"
  Object next() // = the next element
```

Total Marks = 90