

PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science
DECEMBER EXAMINATIONS

CSC 108H/A08H
Duration — 3 hours

Examination Aids: None

Student Number: _____

Last (Family) Name(s): _____

First (Given) Name(s): _____

Instructor: _____
(circle one) Campbell Engels Gries Szamosi

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 *carefully.*)

MARKING GUIDE

- # 1: _____/ 20
- # 2: _____/ 20
- # 3: _____/ 20
- # 4: _____/ 10
- # 5: _____/ 20
- # 6: _____/ 20
- # 7: _____/ 20

This final examination consists of 7 questions on 18 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.*

Comments are not necessary except where we ask for them.

For 2 bonus marks, write you student number in the space provided at the bottom of *both* sides of *every* page of this exam.

TOTAL: _____/130

Good Luck!

Question 1. [20 MARKS]

1. Write a class called **Anniversary** that customizes the **Date** class, and contains a **String** instance variable called **spouseName**.
2. Given an integer called **hoursLeft**, write a **while** loop that decrements **hoursLeft** by 1 each iteration and continues looping as long as **hoursLeft** is not negative.
3. Given a **StringTokenizer** object called **dispenser**, write a **while** loop that prints all of **dispenser**'s tokens to the screen, one per line.
4. Given an array called **bookTitles** that contains at least 1 **Object**, write an expression that evaluates to **true** if the first element of **bookTitles** is a **String** object.
5. Write a single statement that declares and instantiates an array of four **String** objects called **seasons**, and initializes it to the values "**Spring**", "**Summer**", "**Autumn**" and "**Winter**".
6. Given an array **studentNumbers**, write an expression that refers to the last element of the array.

Question 2. [20 MARKS]

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

- If the fragment has a compile error, then place a checkmark ✓ in the first column.
- If the fragment compiles, but has a run-time error, then place a checkmark ✓ 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 there is no error, but you are unsure of the result, then place a checkmark ✓ 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?
<code>char a = 3 + 'c';</code>			a's value:
<code>boolean b = 1 && 0;</code>			b's value:
<code>String c = "final exam"; char d = c.charAt(c.length - 1);</code>			d's value:
<code>String e = "testing 123"; e = e.substring(1, 4); char f = e.charAt(3);</code>			f's value:
<code>Integer g = new Integer(25); System.out.println(g * 3);</code>			output of System.out.println:
<code>double h = int(3 * 5 / 2);</code>			h's value
<code>JFrame j = new JFrame("Final"); JFrame k = new JFrame("Exam"); j = k; k = j; String t = k.getTitle();</code>			t's value:
<code>Date[] d = new Date[5]; int month = d[0].getMonth();</code>			month's value:
<code>int i = 0; int[] a = {1,2,3,4,5}; while (i <= a.length) { a[i] = a[i] + 1; }</code>			a's contents:
<code>int[][] z = {{1,2},{4,5},{5,6}}; for(int i = 0; i < z.length; i++) { z[i][i] = z[i][i] * 2; }</code>			z's contents:

Question 3. [20 MARKS]**Part (a)** [6 MARKS] Complete the following code.

```
/** A song with a title and an integer length (in seconds). */  
public class Song {
```

```
    // Declare any necessary instance variables here.
```

```
    /** A new song with title t and length len. */  
    public Song(String t, int len) {
```

```
    }
```

```
    /** Return the title of this song. */  
    public String getTitle() {
```

```
    }
```

```
    /** Set the title of the song to t. */  
    public void setTitle(String t) {
```

```
    }
```

```
    /** Get the length of the song. */  
    public int getLength() {
```

```
    }
```

```
}
```

Part (b) [14 MARKS] Complete the following code. You will be marked on both design and correctness.

```
/**
 * An album with a list of songs. The first song is at index 0. An album has
 * both a maximum number of songs and a maximum length (in seconds).
 */
public class Album {

    // Declare any necessary instance variables here.

    /** A new album that can hold up to n songs and s seconds worth of songs. */
    public Album(int n, int s) {

    }

    /** Return the index of the song with title t, or -1 if there is no song with that title. */
    public int getSongIndex(String t) {

    }

}
```

```
/** Return true if there is room on this album for another song, and false otherwise. */
public boolean isRoom() {

}

/**
 * Add s to the end of the list of songs, unless there is no room or there is already a
 * song with that title on the album. Return whether s was successfully added.
 */
public boolean addSong(Song s) {

}

/**
 * Remove the song with title t from the album (if it is currently on the album) and
 * return whether the song was removed. The order of the remaining songs does not change.
 */
public boolean removeSong(String t) {

}

}
}
```

Question 4. [10 MARKS]

Part (a) [2 MARKS] An array of `String` arrays is declared and initialized as follows:

```
String[] [] firstNames = new String[3] [];
firstNames[0] = new String[1];
firstNames[1] = new String[3];
firstNames[2] = new String[5];
```

How many names will fit into the 2D array referred to by `firstNames`? _____

Part (b) [8 MARKS] Consider class A:

```
public class A {
    public static char[] [] smart(char[] [] inputArray, int r, int c) {
        char[] [] t = new char[c][r];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                t[j][i] = inputArray[i][j];
            }
        }

        return t;
    }

    public static void printCharArray(char[] [] charArray) {
        for (int i= 0; i < charArray.length; i++){
            for (int j=0; j < charArray[i].length; j++) {
                System.out.print(charArray[i][j] + " ");
            }

            System.out.println();
        }
    }
}
```

What is the output of the following set of statements?

```
char[] [] kp = {{'1','4','7','*'}, {'2','5','8','0'}, {'3','6','9','#'}};
char[] [] tkp = A.smart(kp, 3, 4);
A.printCharArray(tkp);
```

Output:

Question 5. [20 MARKS]**Part (a)** [10 MARKS]

Write a `public static void` method called `diagonal` that has a square two-dimensional array of integers as its parameter and sets all the values along the two diagonals to `-1`. (The diagonals go from the upper-left corner down to the lower-right corner, and the upper-right corner down to the lower-left corner.) You may assume the parameter is always a square array. Your code should be as simple as possible.

Also write a Javadoc comment for your method, including an `@param` tag.

```
public class Diagonal {
```

```
}
```

Part (b) [10 MARKS]

Write a good set of JUnit test methods to test `diagonal`. You probably will need a `private` helper method to compare arrays. *Only test with square arrays.*

```
import junit.framework.TestCase;

/** Test method diagonal. */
public class TestDiagonal extends TestCase {
```

```
}
```

Question 6. [20 MARKS]

Part (a) [10 MARKS]

Consider the following array of **Strings**.

```

+=====+=====+=====+=====+=====+=====+=====+
| Sleepy | Happy  | Grumpy | Dopey  | Snezy  | Bashful | Doc   |
+=====+=====+=====+=====+=====+=====+=====+

```

Perform the first four iterations of insertion and selection sort, and draw a snapshot of the array contents in each case in the array diagrams shown below. In this case, four iterations means that four elements have been considered and sorted in alphabetical order, with the remaining three left to go. (You may assume that **String**'s `compareTo` method has been used to determine ordering.)

Insertion Sort after 4 iterations:

```

+=====+=====+=====+=====+=====+=====+=====+
|         |         |         |         |         |         |         |
|         |         |         |         |         |         |         |
+=====+=====+=====+=====+=====+=====+=====+

```

Selection Sort after 4 iterations:

```

+=====+=====+=====+=====+=====+=====+=====+
|         |         |         |         |         |         |         |
|         |         |         |         |         |         |         |
+=====+=====+=====+=====+=====+=====+=====+

```

Use this space for rough work:

Part (b) [10 MARKS]

Write a `public static` method called `alphaOrder` that takes in a `String` parameter called `inputString`. This method returns a `String` that has the same length and contains the same letters as `inputString`, but with the letters in alphabetical order, from `a` to `z`.

Characters in `inputString` may occur more than once. You may assume that all the characters of `inputString` are lowercase letters. You will be marked on both design and correctness.

Question 7. [20 MARKS]

This question is about input and output.

Assume that all appropriate classes have been imported.

Part (a) [8 MARKS] Complete this method:

```
/** Return the number of lines in the file named fileName. */  
public static int numLines(String fileName) throws IOException {
```

```
}
```

Part (b) [12 MARKS]

Complete the method below so that it prints the prompt "Enter an integer: ", waits for a user to type an integer, triples the entered value, and writes the result to a file named fileName.

```
public static void writeInt(String fileName) throws IOException {
```

```
}
```

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

Short Java API descriptions (everything is 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
```

File:

```
File(String pathname) // Create a File linked with a file named f
```

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
FileReader(File f) // Create a Reader reading from a file f
```

InputStreamReader:

```
InputStreamReader(InputStream in) // Create a Reader reading from in
```

FileOutputStream:

```
FileOutputStream(File f) // Create an OutputStream writing to f
FileOutputStream(String f) // Create an OutputStream writing to a file named f
```

PrintStream:

```
PrintStream(OutputStream out) // Create a print stream sending output to out
print(String s) // Print s to the output
print(int i) // Print i to the output (similar methods exist for all primitive types)
println(String s) // Print s to the output
println(int i) // Print i to the output and terminate the line
// (similar methods exist for all primitive types)
println() // terminate the lines
```

System:

```
static InputStream in // input from the keyboard
static PrintStream out // standard output stream
```

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
```

Container:

```
// add item c 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 c, String loc)
add(Component c) // add c to the next spot in the container
void setLayout(LayoutManager lm) // set this JFrame's layout manager to lm
```

BorderLayout:

```
BorderLayout() // A LayoutManager with "North", "South", "East", "West", and "Center"
```

GridLayout:

```
GridLayout(int r, int c) // A LayoutManager with r rows and c columns
```

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
```

JLabel:

```
JLabel() // A label with no text
JLabel(String s) // A label containing text s
String getText() // = this JLabel's text
String setText(String s) // = set this JLabel's text to s
```