

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science
APRIL EXAMINATIONS 2001
CSC 108H1 S
Duration — 3 hours

PLEASE HAND IN

Examination Aids: *The Java API: An Introduction for Students*, July 2000 edition.

Student Number: _____

Last Name: _____

First Name: _____

Lecture Section: (circle one)
Erindale: L0101 (Arnold Rosenbloom)
St. George: L0101 (Paul Gries) St. George: L5101 (Paul Gries)

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*.)

This final examination consists of 8 questions on 13 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the examination is complete.* Answer each question directly on the examination paper, in the space provided.

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero (0) so write legibly. In your answers, you may use any class or method mentioned in *The Java API*, unless otherwise indicated for a particular question. You may *not* use classes or methods that are not specifically mentioned in *The Java API*.

General Hint: We were careful to leave ample space on the examination paper to answer each question, so if you find yourself using much more room than what is available, you're probably missing something. Also, remember that hints are just hints: you are not required to follow them if you can think of a different solution.

1: _____/ 12
2: _____/ 12
3: _____/ 24
4: _____/ 14
5: _____/ 12
6: _____/ 22
7: _____/ 16
8: _____/ 10

TOTAL: _____/122

Good Luck!

Question 1. [12 MARKS]**Part (a)** [7 MARKS]

Write the body of method `equalCaseMixed`:

```
/**
 * Return true if and only if both of the following are true:
 *
 * 1. Ignoring capitalization, s and t are equal.
 *
 * 2. When you consider capitalization, no corresponding characters
 *    in s and t have the same case. Stated another way, for every
 *    index i, s.charAt(i) != t.charAt(i).
 *
 * Examples:
 *
 *   equalCaseMixed("ABcd", "abCD") returns true.
 *
 *   equalCaseMixed("ABcd", "abCE") returns false, because the Strings
 *   are not equal, even ignoring capitalization.
 *
 *   equalCaseMixed("ABcd", "abCd") returns false, because the 'd' is
 *   lowercase in both Strings.
 *
 * Preconditions: s != null and t != null, and s and t contain ONLY
 * chars in the ranges a-z and A-Z --there is no punctuation or whitespace.
 */
public static boolean equalCaseMixed(String s, String t) {
```

}

Part (b) [5 MARKS]

We have started a list of test cases for method `equalCaseMixed`, described on the previous page. Finish writing a good set of test cases. When doing the testing, you are allowed to consider the approach you took to solve the problem, although you don't have to. That is, if you wrote the solution with a particular approach that allows you to combine or ignore test cases, you can do so. If you prefer, you can also test based entirely on the method specification.

s	t	expected output	reason for test
"a"	"a"	false	both length 1, both lowercase
"A"	"A"	false	both length 1, both uppercase

Question 2. [12 MARKS]Complete method `addRows`.

```
/**
 * Return a list containing the sums of the rows of m.
 * For example, if m is this 2D array:
 *
 *      [1 2 3]
 *      [4 5 6]
 *
 * then the result should be the array [6 15].
 *
 * Here, m is in row-major order: it's an array of rows.
 */
public int[] addRows(int[][] m) {
```

}

Question 3. [24 MARKS]**Part (a)** [12 MARKS]

Write the body of method `largest` so that it meets its specification.

Example: if `v=["cart", "sid", "zoo", "bee", "smart"]` then `largest(v)` returns `"zoo"`.

Hint: you will need `String` method `compareTo`. `"zoo"` is larger than `"ace"`, hence the following:

- `"ace".compareTo("zoo")` returns a negative `int`.
- `"ace".compareTo("ace")` returns `0`.
- `"zoo".compareTo("ace")` returns a positive `int`.

```
// Return the largest String in v, or null if v
// is either empty or null.
// Precondition: v contains only Strings.
public static String largest(Vector v) {
```

```
}
```

Part (b) [12 MARKS]

Write a method `getNLargestSorted(Vector v, int n)` that returns a new `Vector` containing the `n` largest elements in `v`, in increasing order. Those `n` largest elements should be removed from `v`.

Feel free to put a precondition on the value of `n`, and whether `v` may be `null`.

You can call method `largest` from part A of this question, even if you didn't finish it –you can assume that it will work.

Example: if `v` is `["cart", "sid", "zoo", "bee", "smart"]` then `getNLargestSorted(v,3)` returns the `Vector ["sid", "smart", "zoo"]`.

Answer:

Question 4. [14 MARKS]**Part (a)** [4 MARKS]

Complete the identification section at the top of page 1 *including your lecture section*, then write your student number **legibly** at the bottom of every page of this examination (where indicated).

Part (b) [10 MARKS]

Write a **static** method `replace` (including a method comment) that behaves as follows:

- `replace("cart",3,'s')` returns "cars"
- `replace("cars",2,'n')` returns "cans"
- `replace("cans",0,'p')` returns "pans"

You may assume that the second argument is a valid index into the first argument.

Question 5. [12 MARKS]**Part (a)** [6 MARKS]

What is the output of the following program?

```
public class ValHold {
    public int i = 10;
}

public class ObParm {
    public static void main(String[] argv) {
        ObParm o = new ObParm();
        o.aMethod();
    } // End of main

    public void aMethod() {
        int i = 99;
        ValHold v = new ValHold();
        v.i = 30;
        another(v,i);
        System.out.println(v.i);
    } // End of aMethod

    public void another(ValHold v, int i) {
        i = 0;
        v.i = 20;
        ValHold vh = new ValHold();
        v = vh;
        System.out.println(v.i + " " + i);
    } // End of another
}
```

Answer

Part (b) [6 MARKS]

On the back of the previous page, draw the memory model for `ObParm` just *before* line 3 of method `another` is executed. Don't bother drawing boxes for the Java API classes (like `System` and `PrintStream`).

Question 6. [22 MARKS]

An electronic combination lock has a display consisting of 3 integers, which a user has entered:

Enter guess: 41 15 22

A lock also has a combination consisting of three *hidden* integers, one for each integer in the display. A lock's combination is set by the creator of the lock; a lock is locked when it is first created. Once a lock is created, there is no direct access to the combination.

The three displayed integers (41, 15, and 22, above) are *guesses*, and all initially have value 0. These guesses can be changed by someone trying to unlock the lock. The guesses can be changed in any order. When each guess matches its corresponding hidden integer, the lock is unlocked. A lock can be reset, meaning that the three guesses are set to 0 (and the lock, therefore, is locked).

Every lock is either locked or unlocked, and the lock can be asked whether it is locked or unlocked. The lock also can produce a **String** describing its state, including the values of the three current guesses and whether the lock is locked or unlocked.

Part (a) [12 MARKS]

Design and write a class to model electronic combination locks. If you use reasonably descriptive method and instance variable names you don't have to write comments. *Do not prompt the user to input the guesses: instead, provide methods that a main program can call.*

Use the back of the previous page when you run out of room.

Part (b) [5 MARKS]

Write a program that implements the following algorithm:

1. Create a combination lock with combination 77,34,39.
2. Print the state of the lock.
3. Enter 33 for the second guess.
4. Print the state of the lock.
5. Enter 77 for the first guess, then 39 for the third guess.
6. Print the state of the lock.
7. Enter 34 for the second guess.
8. Print the state of the lock.
9. Reset the lock.
10. Print the state of the lock.

Part (c) [5 MARKS]

In the space below, show the output of the algorithm.

Question 7. [16 MARKS]

The following code defines a small inheritance hierarchy, but it's a bit odd: `Rectangle` extends `Square`, but in geometry not all `Rectangles` are `Squares`. On the back of the previous page, rewrite `Square` and `Rectangle` so that `Square` extends `Rectangle`. Your new classes should have as little duplicate code as possible, and should produce **exactly** the same output as the provided code. You don't have to rewrite the comments, and `ShapeTest` doesn't change.

```

-----
public class Square {
    // this Square's width.
    private double width;

    // A Square with width w.
    public Square(double w) {
        width = w;
    }

    // Return this Square's area, perimeter
    // and width.
    public double getArea()
        { return width*width; }
    public double getPerimeter()
        { return 4*width; }
    public double getWidth()
        { return width; }

    // Return this Square's String
    // representation.
    public String toString(){
        return
            "Width=" + getWidth();
    }
}

public class Rectangle extends Square {
    // this Rectangle's height.
    private double height;

    // A Rectangle of width w and height h.
    public Rectangle(double w, double h) {
        super(w);
        height = h;
    }

    // Return this Rectangle's area, perimeter
    // and height.
    public double getArea()
        { return getWidth()*getHeight(); }
    public double getPerimeter()
        { return 2 * (getWidth()+getHeight()); }
    public double getHeight()
        { return height; }

    // Return this Square's String
    // representation.
    public String toString() {
        return super.toString() + ", height="
            + getHeight();
    }
}
-----

```

```

public class ShapeTest {
    public static void main(String [] args){
        Square s = new Square(12);
        Rectangle r = new Rectangle(3,5);

        System.out.println(s);
        System.out.println(r);
    }
}
/*
Output: Width=12.0
        Width=3.0, height=5.0
*/

```

Question 8. [10 MARKS]

Write a public method `m1` that has three parameters: a `Vector v` that contains only `Strings`, a `String s`, and a `boolean b`. You may assume that `s` contains an integer. For example, it might refer to the `String "134"`. The method returns `true` if and only if the following statement is true:

If `b` is true then the value in `s` is the number of items in `v`, and if `b` is false then `s` contains the index in `v` of a `String` that is equal to `s`.

As an example, if `Vector vec` contains `["a", "1"]`, then the method call `m1(vec, "1", false)` evaluates to `true`.

It is fine for your method to throw an exception if `s` does not contain an integer, or if the integer in `s` is not a valid index into `v`. You may also assume that none of the parameters are `null`.

If you use an `if` statement (or the `?:` construct) you can get a maximum of 7 marks. For the full 10 marks, solve it with *only* a `return` statement.

There is NO question on this page!

*[If you need extra space to answer a question, use the space below and indicate **clearly** the question number.]*

Total Marks = 122

Student #: _____

Page 13 of 13

END OF EXAMINATION