**Cornell University**
**Department of Computer Science**

# CS 99: Fundamental Programming Concepts
# Summer 2000

## Preliminary Examination #2
## July 25, 2000

New/Updated Instructions:

- Again, this is a difficult exam. Remember that the entire course will be curved at the end of the term.
- There are 7 problems on this exam. The number of points each problem is worth is listed in brackets before the problem. Partial credit will be awarded if you provide your reasoning.
- Do not begin reading the rest of the exam until instructed to begin. When you start, make sure that you have all 10 pages of the exam.
- You may use the abbreviation `println` for `System.out.println` (and the same for `print` and `System.out.print`).
- Correctness is our primary concern, not style. No comments are necessary in your code. Indentation is not an important issue. However, unreadable code may be penalized.

Old Instructions:

- You will have one hour to complete the exam.
- Write all your answers on the exam itself. If you run out of space on a problem, write OVER on the front side of the page, and use the back side to continue your answer.
- This is a closed book exam. You may not use any materials, including calculators, during the exam other than the exam handout and writing instruments.
- If you feel a question is ambiguous, stay in your seat, raise your hand, and someone will come to help you. If you still feel the question is ambiguous, state some reasonable assumptions and solve the problem accordingly.
- If you finish before the last 10 minutes of the exam, you may leave quietly. If you finish in the last ten minutes of the exam, please remain in your seat until the end of the exam. Do not take out any additional materials.
- Write your 4-digit secret number (from the student questionnaire you filled out on the first day of the class – probably the last 4 digits of you student ID number) below. You may also write your name, if you wish, or we will look it up based on your number *after* we grade your exam.

**Name or secret number:** _____

| Problem | Points | Score |
|---|---|---|
| 1 | 20 | |
| 2 | 25 | |
| 3 | 10 | |
| 4 | 15 | |
| 5 | 10 | |
| 6 | 15 | |
| 7 | 35 | |
| Total | 130 | |

General Comments:

1. Don't put I/O in a class unless it is specified
2. Primitive types are not classes, and therefore you cannot invoke a method on a primitive variable. That means it is impossible to use `.equals()` on a primitive; you must use `==`.
3. Variables declared inside a method are only in scope until the end of the method REGARDLESS of what modifiers with which the method is declared. Local data is *always* local, even if the method is static.
4. The meaning of `static` is different for methods than for fields. Many people confuse the two. All methods are always in scope throughout an entire class, whether they are static or not.
5. `void` means no return value, not no return type. The return type of `main`, for example, is `void`.

1. **True/False.** [20 points, each correct answer is worth 2 points, each answer left blank is worth 1 point, each incorrect answer is worth 0 points]

F a. Bounds checking refers to Java's built-in checks to make sure you don't assign an out-of-range value to a variable.

T b. Using two references to the same object is called aliasing.

T c. The `final` keyword is used to declare constants.

T d. `static` fields are in scope throughout all methods in a class.

F e. The last element of an array named `list` can be indexed as `list[list.length]`.

T f. It is not possible to change the size of an array object once it is created.

F g. Loop index variables are declared inside a loop body.

T h. Instance data should be declared using the `private` keyword.

T i. A sentinel value can be used to obtain an unknown amount of input.

F j. An infinite loop results in a compile-time error.

2. **Short Answer.** [25 points]

   a. [2] What is the difference between a pre-test and a post-test loop? Give an example of each in Java (just the keyword for the loop will suffice).

   A pre-test loop (e.g., `while` or `for`) checks its condition before executing the body; a post-test (e.g., `do`), after.

   b. [3] Since `while` loops can always be rewritten as `for` loops, and vice-versa, why are both included in Java? (Hint: when should you use one instead of the other?)

   Both are included so that fixed and variable repetition constructs are available.

   c. [2] What is the default value of reference variables?

   `null`

   d. [2] What are the two kinds of members of a class?

   fields and methods
   NOT public/private or static/non-static

   e. [2] Give one of the two meanings of *default constructor*:

   The constructor that Java generates automatically if you supply none, or a constructor with an empty parameter list. Defining *constructor* was not worth any points.

f. [4] Consider the following code:

```
int k = 0;
for (int i = 0; i < 4; i++) {
     for (int j = 0; j < 10; j++) {
          System.out.println(i+j);
     }
}
```

How many lines of output are produced by the code?

40

What is the final value printed by the code?

12

g. [10] The header for `main` is given below. Identify the purpose of each component.

```
public static void main(String[] args)
```

`public:` Makes the method visible from outside of its class.

`static:` Method is associated with entire class, not a particular object.

`void:` Method returns nothing.

`main:` Name of method, first method invoked in a program.

`String[]:` Array of strings, type of `args.`

`args:` Command-line arguments supplied to program.

3. **Program output.** [10 points] For each of the following program fragments, provide the exact output produced by the fragment. Use the box below the program to provide the output.

   a. [5] The list of values that the user will enter (if prompted enough times) is:

   42, 0, 1000, 1, 999, 50

```
int c = Console.readInt();
while (c != 999) {
     if (c > 0) {
          System.out.println(c);
     }
     c = Console.readInt();
}
```

| |
|---|
| 42<br>1000<br>1 |

   b. [5]

```
class Averager {

     static double avg2(int i, int j,
                        double avg) {
          avg = (double) i / j;
          return avg;
     }

     public static void main(String[] args) {
          int a = 1, b = 2;
          double average = 0;
          avg2(a, b, average);
          System.out.println(average);
     }
}
```

| |
|---|
| 0.0 |

4. **Using Loops.** [15 points]

a. [5] Rewrite this `for` loop as a `while` loop:

```
for (int i = 0; i <= LIMIT; i += 2) {
     foo(i);
}
```

```
int i = 0;
while (i <= LIMIT) {
     foo(i);
     i += 2;
}
```

b. [5] Input an integer. Perform data validation to enforce that it is in the range 1 to 100, inclusive.

```
int i;
boolean valid = false;
do {
      System.out.println("Enter an integer (1-100): ");
      i = Console.readInt();
      if (1 <= i && i <= 100) {
            valid = true;
      } else {
            System.out.println("Must be between 1 and"
                               + "100.");
      }
} while (!valid);
```

-3 for just using an `if` and not looping

c. [5] Write a method named `minValue` to find the minimum element of an array of integers and return its value.

```
int minValue(int[] a) {
     int min = a[0];
     for (int i = 0; i < a.length; i++) {
          if (a[i] < min) {
               min = a[i];
          }
     }
     return min;
}
```

No credit for just finding the element at the minimum index (`a[0]`).

5. **Using Arrays.** [10 points]

   a. [3] Declare and create an array of twenty `double`s named `times`.

   ```
   double[] times = new double[20];
   ```

   b. [2] Assign the 5th element of `times` the value 5.56.

   ```
   times[4] = 5.56
   ```

   c. [2] Assign `x` the sum of the first and last elements of `times`.

   ```
   x = times[0] + times[19];
   ```
   or
   ```
   x = times[0] + times[times.length-1]
   ```

   d. [3] Assume that you have a method named `sumArray` that calculates the sum of all the values in an array. *You do not need to write the method as part of this problem.* Using this method, assign `x` the sum of the values in `times`.

   ```
   x = sumArray(times);
   ```

   You had to pass an array to the method to receive full credit.

6. **Using Classes.** [15 points] The following questions use the `Rectangle` class. The class includes the following methods:

```
Rectangle(int w, int l)
int area()
boolean equals(Rectangle r)
```

*You do not need to write the above methods as part of this problem.*

a. [2] Declare a reference to an object of class `Rectangle`. You may pick your own name for the reference variable.

```
Rectangle rect;
```

b. [2] Create an object of class `Rectangle`, with a width of 5 and a length of 7, and assign it to the reference you created above. Do not redeclare the reference.

```
rect = new Rectangle(5, 7);
```

c. [3] Using the `area()`method, write code to print out the area of the rectangle you created in part b.

```
System.out.println(rect.area());
```

d. [3] Given two objects of class `Rectangle`, `r1` and `r2`, write code that prints a message stating whether the objects are equal or not.

```
if (r1.equals(r2)) {
     System.out.println(“Equal”);
} else {
     System.out.println(“Not equal”);
}
```

e. [5] What specific problem(s) would occur if the `area()` method were made `static`? (You may wish to do problem 7 first.)

It would not be able to access any instance data – the `length` and `width` fields wouldn’t be in scope.

7. [35 points] Write the class `Rectangle`. You should provide the following public methods, and any private members that you wish to include.

```
Rectangle()    // init. to width=1, length=1
Rectangle(int w, int l)
int area()
boolean isSquare()
boolean equals(Rectangle r)
```

The class also has the following private methods that you DO NOT need to write. They validate the width and length of a rectangle. Use them as appropriate in the code that you write. Each accepts a potential value, and returns a validated value (e.g., if the potential value were out-of-range, the method would return a default in-range value).

```
int validateWidth(int w)
int validateLength(int l)
```

```
class Rectangle {                          // 1 point
     private int width, length;            // 4 points

     public Rectangle() {                  // 6 points
          width = length = 1;
     }

     public Rectangle(int w, int l) {      // 8 points
          width = validateWidth(w);
          length = validateLength(l);
     }

     public int area() {                   // 4 points
          return width * length;
     }

     public boolean isSquare() {           // 4 points
          return width == length;
     }
                                           // 8 points
     public boolean equals(Rectangle r) {
          return this.width == r.width
                 && this.length == r.length;
     }
}
```