Computer Science Java Placement Exam

The Computer Science Department now has available a placement exam for students who wish to demonstrate their competence in the material covered in the course CSCI C101 Computer Programming I. Students who pass this exam at a high level may enroll in CSCI C201 Computer Programming II without first completing C101. Such students may also have the course CSCI C101 placed on their transcript for 4 credit hours, provided they pay the required fees.

Topics to Be Covered on the Java Exam

The placement exam is designed to test carefully whether a student has mastery of the topics covered in C101 so that the student can do the work in C201 without burdening the instructor with inappropriate questions. Mastery of all the following topics is required:

1. Use of a Java-compatible IDE (Integrated Development Environment) such as Eclipse or NetBeans. Students must demonstrate the ability to edit source code efficiently, to save a file, to compile source code into an executable program, to run a program, and to use the debugger, all from the IDE.

2. Systematic program development.

Students must demonstrate ability to organize a program into appropriate functions and to design the algorithms used by the functions.

3. The elementary features of the Java programming language.

A complete checklist of all the required features is given farther along in this document.

4. Good programming style. Students must demonstrate that they can

a. use a clear and consistent indenting method to produce a logically formatted program;
b. choose appropriate identifiers for variables;
c. write complete, understandable documentation for a program and each of its functions, and also provide helpful line-by-line comments where these are appropriate.

Format of the Java Exam

The exam will be in two parts. The first part will last one hour and will require the student to answer a number of written questions about Java and program development. The second part will be conducted in a computer laboratory, where the student will be given four hours in which to write one or more complete Java programs and/or fill in the missing parts of some incomplete programs.

A Warning about the Programming Part of the Java Exam

A student program must not only correctly exhibit the behavior prescribed in a programming problem, but also be well written. Thus, to give some examples, a program that is poorly formatted, or poorly documented, or that makes unnecessary tests of data, or that uses a poor algorithm will have points deducted from the grade for these failings.

A Checklist of Java Features That Must Be Mastered

1. Structure of Java programs:
   a. placement of "import" statements;
   b. placement of the main function and other functions.

2. Basic syntax rules for
   a. identifiers;
   b. declarations and initializations of variables;
   c. control structures ("if", "if...else...", "switch", "while", "do...while", "for").

3. Fundamental data types:
   a. int, boolean, char, float, long, double; understanding of the inexactness of stored floating point values;
   b. enum types.

4. Common operators and the value of an operator expression:
   a. assignment operators: = , += , -= , *= , /= , %= ;
   b. arithmetic operators: + , ++ , - , -- , * , / , % ;
   c. relational operators: < , <= , > , >= , == , != ;
   d. logical operators: && , || , ! ;
   e. type casting operators ;
   f. elementary precedence rules among these operators.

5. Elementary I/O (input/output) in Java using the following operators and functions defined in the System and Scanner classes:
   a. System.out.print, println, printf;
   b. System.in.read, readline,
   c. Scanner nextInt, nextFloat, nextLine();

6. Conditional statements using if... or if...else...; nested conditional statements.

7. Loops using while... or do...while... or for....

8. Functions and parameter passing. Difference between pass by value and pass by object. Function declarations (prototypes) and function definitions (code).

9. These functions from Java.lang.Math: sqrt(); pow(); abs();

10. Elementary scope rules.

11. Arrays and how they work as function parameters.

Some Formatting Requirements for the Java Programming Exam

1. Use all upper case letters for identifiers that denote constants. This is standard practice in C and Java programming.

2. The common Java convention is that variable, attribute, parameter, and function (method) names start with a lowercase letter, and class names start with an uppercase letter.

3. Use a consistent style for identifiers for variables. You may choose any one of the following styles, but then use only that one style you have chosen:

```
location_of_2nd_target      // Digits and lower case letters are used,
                            // with words separated by the underscore
                            // character.

locationOf2ndTarget         // Each word of the identifier except for the
                            // first starts with an upper case letter or
                            // a digit.  Words are not separated.

location_Of_2nd_Target      // Each word of the identifier starts with
                            // an upper case letter or a digit, and
                            // words are separated.
```

4. Use exactly four spaces for each level of indenting.

```
Correct:    while (i < LIMIT)
                if (a[i] > a[best])
                    best = i;

Incorrect:  while (i < LIMIT)
             if (a[i] > a[best])
               best = i;

Incorrect:  while (i < LIMIT)
               if (a[i] > a[best])
                  best = i;
```

5. Use any one of the following styles for placement of the braces in a compound statement in a Java program. Whichever style you pick, use that style consistently throughout the program.

```
Style 1:   if (x < 0)
           {
               System.out("The value is negative.");
               System.out("Enter another value: ";
               x = scan.nextInt();
           }

Style 2:   if (x < 0) {
               System.out.println("The value is negative.");
               System.out.println("Enter another value: ");
               x = scan.nextInt();
           }
```

6. Place a blank on each side of every binary operator symbol.

```
Correct:    while (i < LIMIT)
                if (a[i] > a[best])
                    best = i;

Incorrect:  while (i<LIMIT)
                if (a[i]>a[best])
                    best=i;
```

7. Make liberal use of blank lines to group related statements within a function.

8. Every function except **`main()`** must have its own documentation section immediately preceding the code for the function. The parameters should be described, and all assumptions that are being made about the parameters should be explicitly stated. The action(s) of the function should be described, but no mention should be made of other function(s) that will be calling this function or how it "fits into" the program as a whole. The documentation for a function should be preceded by a "banner" line that names the function.

9. Don't let comments spoil the logical structure of the code on the page. The best place for comments is out to the right of the actual code. When you must write inter-linear comments, indent them the same amount as the code they describe.

ACCEPTABLE:

```
   if (balance < 0.0)
       // Print a message denying service.
       System.out.println("Login failed.");
```

BETTER:

```
   if (balance < 0.0)   // then print a message denying service.
       System.out.println("Login failed.");
```

UNACCEPTABLE:

```
   if (balance < 0.0)
// Print a message denying service.
       System.out.println("Login failed.");
```