S1007 - Introduction to Computer Science
Practice Final Exam
Summer 2003

Name:

CUNIX ID:

You have 180 minutes to answer all of the questions below. Write your answers in the space provided. You should read over the entire exam before you answer any questions and budget your time accordingly.

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
<th>Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>
1. (8 points) What is object-oriented programming? What are some features you would expect to find in an object-oriented programming language? Are these features present in Java? Use brief examples to illustrate your answer.

An object-oriented language encourages the programmer to think of the program as a set of self-contained components called objects. Objects contain data and methods with which to manipulate that data. Some typical features of an object-oriented language are:

- **Encapsulation** - data can be hidden inside an object so that it cannot be altered by other objects. Java implements encapsulation using visibility modifiers like `public`, `private`, and `protected`.

- **Inheritance** - objects can inherit data and methods from other objects, allowing specialization. Java implements inheritance using the `extends` keyword:

```java
class A {
    int x;
    ...
}
class B extends A {
    int y;
    ...
}
```

In this example, the class `B` has two fields: `x` and `y`. It inherited `x` from `A`.

- **Polymorphism** - objects that are related by inheritance can be used interchangeably in methods that deal with general common attributes and subclasses can be used wherever a superclass is expected.

```java
A obj = new B();
```

In this example, the variable `obj` will look like an `A`, but act like a `B` (if `B` overrides any of `A`'s methods).
2. (8 points) What is a type in Java? What are the primitive types? What is a user-defined type? How does Java use types to make programming easier and more robust?
3. (10 points) Find ten errors (there are more than ten) in the following Java class. The errors are both syntactic (compiler errors) and logical (bugs). For each error, identify the line number and briefly explain how to fix it.

```
import java.io.*;

/* A class that counts the letter frequency in a text input stream. */

public class Count {

/* Counts the letter frequency in a character input stream. */
    public static count(String fileName) {
        /* initialize the frequency counts */
        int[] alpha = new int[] ;
        for( int i=0 ; i <= alpha.length ; i++ )
            alpha[i] = 0

        /* Keep track of line numbers for error reporting. */
        try {
            LineNumberReader in2 = new LineNumberReader(  
                new FileReader( fileName ) ) ;
            /* Read through to end of file. */
            while( ( ch = in2.read() ) != -1 )
                char c2 = (char) ch ;

            /* Update character counts. */
            if( Character.isLetter( c2 ) )
                alpha[c2-'a']++ ;
        } catch( Exception e ) {
            System.err.println("Could not count file.");
        } catch( IOException e ) {
            /* Handle input error. */
            System.err.println("Error reading line ",
                in2.getLineNumber()) ;
            System.err.println(e.getMessage()) ;
        } finally {
            /* Make sure the stream is closed. */
            try { in2.close() ; } catch( Exception e ) { }
        }
    }
}
```
/* Print out results. */
for( int i=0 ; i < alpha.length ; )
    System.out.println("\"\"+(char)('a'+i)+"\": "
           +alpha[i]) ;
}

/* Takes a list of files to process. */
public static void main(string[] args) {
    for( int i=0 ; i < args.length ; i++ )
        count( args[i] ) ;
}

3. (cont’d)
4. (12 points) Write the output of the following method called with the parameters
q(nums,0,nums.length-1), where nums is an array initialized as follows:

```java
int[] nums = {5, 7, 3, 1, 4, 7};
```

Include all output from entrance to the method to final exit, including the output of all
recursive calls.

```java
public static void q(int[] nums, int p, int r) {
    if( p < r ) {
        int x = nums[p] ;
        int i = p-1 ;
        int j = r+1 ;

        while( i < j ) {
            do { j-- ; } while( nums[j] > x ) ;
            do { i++ ; } while( nums[i] < x ) ;

            if( i < j ) {
                int tmp = nums[i] ;
                nums[i] = nums[j] ;
                nums[j] = tmp ;
            }
        }

        for( int n=p; n <= r ; n++ )
            System.out.print(nums[n] + " ");
        System.out.println("\np=\\n+\np=\\n+\n\nr=\\r=\\r="+r) ;

        q(nums,p,j) ;
        q(nums,j+1,r) ;

        for( int n=p; n <= r ; n++ )
            System.out.print(nums[n] + " ");
        System.out.println();
    }
}
```

4. (cont’d)
5. (10 points) Write a method named `sqrt` that takes a `double` parameter and returns its square root with a margin of error ±0.01. There should be no calls to other methods inside `sqrt`. Do not worry about the class containing `sqrt`, just work out the method itself.

```java
double sqrt(double x) {
    double a = 0.0;
    double step = 1.0;
    double b = a + step;

    while (step > 0.001) {
        while (b * b < x) {
            a = b;
            b += step;
        }
        step /= 2;
        b = a + step;
    }

    return a;
}
```
6. (6 points) What is the approximate running time of your \texttt{sqrt} method from Question 5 on an input \(n\)? What is the order of magnitude of the running time, using \(\Theta\)-notation? What is a lesser order of magnitude? What is a greater order of magnitude?

7. (6 points) Suppose the class \texttt{Sub} extends the class \texttt{Sandwich}. Which of the following assignments are legal? Mark them “OK” or “NOT OK”.

```java
Sandwich x = new Sandwich();
Sub y = new Sub();
x = y;
y = x;
y = new Sandwich();
x = new Sub();
```