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><strong>Total</strong></td>
<td><strong>60</strong></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?

A type is a set of values and operations that may be performed on them. For instance int is a 32-bit integer type, with values ranging from $-2^{31}$ to $2^{31} - 1$. Valid operations on int are addition, subtraction, multiplication, division, the bitwise operations, etc. Java's primitive types are: int, byte, short, long, float, double, char and boolean.

A user-defined type is a composition of built-in types with special operations defined on it. In Java, a user-defined type is called a class or an interface.

Java checks to make sure every operation is valid with respect to its operands' types at compile time. This prevents careless errors, like treating an integer value as a floating-point, or vice versa. Type checking ensures that every operation in the program is conceptually valid before the program has a chance to run and crash.
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.

```java
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)

Line 10: **count** does not have a return type.
Line 12: The initializer for **alpha** does not specify the length of the array.
Line 13: The **for** loop will exceed the array bounds when \( i = \) **alpha.length**.
Line 14: No semicolon after **alpha[i] = 0**.
Line 23: There is no declaration of the variable **ch**.
Line 23-28: There should be curly braces around the **while** loop.
Line 28: \( c2 \) may be upper or lower case. It should be converted to lower case for comparison with ’a’.
Lines 34 and 38: **in2** is local to the **try** block, it isn’t defined in the **catch** and **finally** blocks.
Line 34: There is no concatenation operator (‘+’) before **in2.getLineNumber()**.
Line 31: The **catch** block for **Exception** will match every possible exception. The **catch** block for **IOException** will never execute.
Line 42: There is no increment in the **for** loop. The loop will never exit.
Line 48: **string** should be capitalized: **String**.
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:
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="+p+" j="+j+" 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)

```
4 1 3 7 5 7
p=0  j=2  r=5
3 1 4
p=0  j=1  r=2
1 3
p=0  j=0  r=1
1 3
1 3 4
7 5 7
p=3  j=4  r=5
5 7
p=3  j=3  r=4
5 7
5 7 7
1 3 4 5 7 7
```
5. (10 points) Write a method named \texttt{sqrt} that takes a \texttt{double} parameter and returns its square root with a margin of error ±0.01. There should be no calls to other methods inside \texttt{sqrt}. Do not worry about the class containing \texttt{sqrt}, just work out the method itself.

```java
/* Finds the square root of \(x\), one binary digit at a time. 
* Iteratively steps through fractional powers of 2 (0.5, 
* 0.25, 0.125, 0.0625, etc.) until the answer is within the 
* stated threshold. The answer should be slightly more 
* accurate than required; better safe than sorry. */

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 `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?

`sqrt` approaches \( \sqrt{n} \) in steps of 1.0, then refines its approximation by stepping through the fractional powers of 2. It will take \( \lfloor \sqrt{n} \rfloor \) steps to get within \( \pm 1.0 \) and a constant number of steps (\( \approx 7 \)) to calculate the significant digits.

\[
T(n) \approx \lfloor \sqrt{n} \rfloor + 7 = \Theta(\sqrt{n})
\]

Lesser orders of magnitude are \( \Theta(\lg n) \) and \( \Theta(1) \). Greater orders of magnitude are \( \Theta(n^2) \) and \( \Theta(c^n) \).

7. (6 points) Suppose the class Sub extends the class 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();
```

Sandwich x = new Sandwich();    OK
Sub y = new Sub();              OK
x = y;                          OK
y = x;                          NOT OK
y = new Sandwich();             NOT OK
x = new Sub();                  OK