

# ISE 208 Midterm Exam 2

Practice Version

Fall 2009

NAME (please print legibly): \_\_\_\_\_

Your University ID Number: \_\_\_\_\_

- Please leave at least one seat between yourself and the person next to you.
- No books or notes can be used during the exam, except for your “cheat sheet”.
- Any **CHEATING** will result in an F as well as being written-up on academic dishonesty.
- The last page contains a summary of some useful methods.

**NO BS bonus:** If you do not know the answer to a problem and leave it blank you will receive 1 point for each sub-part (e.g., a,b,c, etc.) that you leave blank. If you write anything in the space and it is wrong, you will receive a zero. Thus, the score for a completely blank exam is 12. Not all questions are of equal difficulty/points value, so read through the entire exam before beginning!

QUESTION	VALUE	SCORE
1	6	
2	12	
3	6	
4	6	
5	6	
6	10	
7	14	
TOTAL	60	

1. (6 points)

A `String` is a *palindrome* if either of the following conditions are true:

1. It has length 0 (the empty string) or 1
2. The first and last characters are identical, and the rest of the string is also a palindrome

```
public boolean isPalindrome (String s)
{
    if (_____)
        return true;
    else
        return (_____) + isPalindrome (_____);
}
```

Which of the following code fragments is an **INCORRECT** completion for one of the blanks above?

- (a) `s.substring(1, s.length())`
- (b) `s.length() < 2`
- (c) `s.charAt(0) == s.charAt(s.length() - 1)`
- (d) None of the above (all three code fragments are correct)

ANSWER: \_\_\_\_\_

## 2. (12 points)

Some of the constructors in the `SonOfBoo` class will not compile. Circle (or otherwise indicate) any constructors that will not compile. NOTE: There may be multiple answers for this question; if this is the case, you MUST select ALL correct answers to receive full credit.

```
public class Boo {

    public Boo(int i) { }

    public Boo (String s) { }

    public Boo (String s, int i) { }

}

class SonOfBoo extends Boo {

    public SonOfBoo () { super("boo"); }

    public SonOfBoo (int i) { super("Fred"); }

    public SonOfBoo (String s) { super(42); }

    public SonOfBoo (int i, String s) { }

    public SonOfBoo (String a, String b, String c) { super(a, b); }

    public SonOfBoo (int i, int j) { super("man", j); }

    public SonOfBoo (int i, int x, int y) { super(i, "star"); }

}
```

**3. (6 points)**

The `ActionListener` interface contains one method, `actionPerformed`. The `MouseListener` interface contains two methods, `mouseMoved` and `mouseDragged`. The `JFrame` class defines 29 methods, including 4 constructors, and inherits 295 other methods.

What is the minimum number of methods you need to define for the `MyFrame` class below such that when compiled, it doesn't generate a syntax error?

```
public class MyFrame extends JFrame implements ActionListener, MouseMotionListener
{
    ...
}
```

- (a) 1
- (b) 2
- (c) 29
- (d) None of the above

ANSWER: \_\_\_\_\_

**4. (6 points)**

Given the inheritance hierarchy:

```
FileNotFoundException extends IOException extends Exception
```

```
ArrayIndexOutOfBoundsException extends Exception
```

Which of the following is an acceptable ordering for three `catch` blocks?

- (a) `Exception`, `FileNotFoundException`, `IOException`
- (b) `IOException`, `ArrayIndexOutOfBoundsException`, `FileNotFoundException`
- (c) `FileNotFoundException`, `ArrayIndexOutOfBoundsException`, `IOException`
- (d) `IOException`, `Exception`, `ArrayIndexOutOfBoundsException`

ANSWER: \_\_\_\_\_

**5. (6 points)**

Given the following inheritance hierarchy:

```
class Person { ... }  
class Student extends Person { ... }  
class GraduateStudent extends Student { ... }
```

would the following assignment statement be legal? Why or why not?

```
GraduateStudent g = new Student();
```

**6. (10 points)**

Describe the flow of execution through a set of `try...catch...finally` clauses.

**7. (14 points)**

Carefully examine the recursive method below:

```
public static int acker (int m, int n)
{
    if (m == 0)
        return (n + 1);
    else if (n == 0)
        return acker(m - 1, 1);
    else
        return acker(m - 1, acker(m, n - 1));
}
```

Draw the sequence of method calls that will result if this recursive method is invoked from `main()` with the following statement:

```
acker(2, 1)
```

You may begin with the initial call to `acker(2, 1)`.

## THIS PAGE DESCRIBES SOME USEFUL JAVA METHODS

### The Scanner class

- `next()` – Returns a **String** containing the next line of input
- `nextInt()` – Reads and returns an integer value from the input

### The String class

- `charAt(n)` – Returns the character at index  $n$ . Indices are numbered from 0-(length-1).
- `length()` – Returns the number of characters contained in the current **String**.
- `toLowerCase()` – Returns an all-lowercase version of the current **String**.
- `substring(i, j)` – Returns a new **String** containing the characters from index  $i$  up to, but not including, index  $j$ .