CS 102 - Introduction to Programming
Midterm Exam #2 - Prof. Reed
Fall 2008

What is your name?: ___________________________

There are three sections:

I. True/False. . . . . . . . . . . . . .60 points; (30 questions, 2 points each)
II. Multiple Choice  . . . . . . . .40 points; (10 questions, 4 points each)

---------------
100 points total

This test is worth 15% of your final grade. You must put your answers on the bubble form. All code is in Java unless stated otherwise. This test is open book and open notes. For the multiple choice problems, select the best answer for each one and select the appropriate letter on your answer sheet. Be careful - more than one answer may seem to be correct. Some questions are tricky. You have 50 minutes.

I. True/False: (2 points each)

T   F   1. The following code stores the value 3 into average.  int average = (int)7.84/2;
T   F   2. A constant in Java must also be static, otherwise its value can be changed.
T   F   3. A variable in Java can also be declared as static.
T   F   4. The output of the statement below is:  1
   System.out.println( 7 % -3);
T   F   5. After running the code shown below, the value stored in variable y is:  12
   int y = 7;
   y = --y + --y;
T   F   6. After running the code shown below, the value stored in variable z is:  15
   int y = 7;
   int z = y + y++;
T   F   7. The output of the following statements is:  7 Done
   int x = 3;
   int y = 4;
   System.out.print( x + y + """);
   System.out.println(" Done");
T   F   8. If class Transaction contains a static variable called howMany, then each instance of the class will get its own copy of variable howMany.
T   F   9. A static method can not access non-static instance variables.
T   F   10. A non-static method can access static instance variables.
T   F   11. Private instance variables can only be accessed through code in the class where they are declared.
T   F   12. Two methods in a Java class can have the same method name.
13. The output of the program segment below is: Exiting

```java
boolean notDone = false;
int numberOfPieces = 4;

if( notDone = true) {
    if( numberOfPieces > 4)
        System.out.println("You won");
} else {
    System.out.println("Keep playing");
}

System.out.println("Exiting");
```

14. The following code prints the words: second Done

```java
char c='b';
switch (c){
    case 'a': System.out.print("third");
    case 'b': System.out.print("second");
    case 'c': System.out.print("first");
    break;
}
System.out.println(" Done");
```

15. A Constructor is a special kind of method.

16. An array can be used on the left-hand-side of a statement, or on the right-hand-side, but not on both sides of the same statement.

17. The index of an array reference can itself be an array element, such as: theArray[ theArray[3]]

18. Although arrays normally start with 0 as the first index, we can change the default to be 1 instead.

19. As discussed in class, a board game that uses a two-dimensional board (such as checkers, chess, or tic-tac-toe) would best represent the board using a two-dimensional array.

20. Binary search will also work with data that is not sorted, except it takes much longer to find the answer an average.

21. Consider the code used for program 3, where we had a Piece class, a Board class, and a PlayGame class. Assume the following code would be in the PlayGame class. The code shown in the box below would give the same result as the code:

```java
theColor = theBoard.getPieceAtIndex( 5).getCircle().getColor();
```

```
int x = 5;
Piece aPiece = theBoard.getPieceAtIndex( x);
Circle aCircle = aPiece.getCircle();
theColor = aCircle.getColor();
```

22. Every set of if - else - if statements can be alternatively represented using a switch - case statement.

23. Every set of switch - case statements can be alternatively represented using if - else - if statements.
24. Consider the code used for program 3, where we had a *Piece* class, a *Board* class, and a *PlayGame* class. Assume the following code would be in the *Board* class.

```java
Piece thePiece = new Piece(25, 25, 60, "red", yes, "");
Piece anotherPiece = thePiece;
anotherPiece.setColor("green");
```

When run this code will display two different colored pieces on the Canvas.

25. A method can call itself in Java.

26. If method A() calls method B() in Java, and method B() calls method C(), then method C() may not call methods A() or B(), since that would create an endless loop.

27. The following code in Java stores in variable *sum* the sum of positive odd integers less than 10:

```java
int sum = 0;
int x;
for(int x=-1, sum=0; x<=10; x+=2)
    if( x>0) sum+=x;
```

28. The *length()* method is used for both strings and arrays. You must be sure to include the parenthesis.

29. Assume the code shown below, where method *swapValues* is called.

Output of this segment of code is:  Values are: 3 7

```java
int[] numbers = {3, 7};
swapValues1( numbers[0], numbers[1]);
System.out.println("Values are: " + numbers[0] + " " + numbers[1]);
// ... other code
public void swapValues1(int num1, int num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

30. Assume the code shown below, where method *swapValues2* is called.

Output of this segment of code is:  Values are: 3 7

```java
int[] numbers = {3, 7};
swapValues2( numbers, 0, 1);
System.out.println("Values are: " + numbers[0] + " " + numbers[1]);
// ... other code
public void swapValues2(int[] a, int num1, int num2)
{
    int temp = a[num1];
    a[num1] = a[num2];
    a[num2] = temp;
}


II. Multiple Choice (5 pts. each)

31. Consider the code given below. Its output is:

```java
int[] values = new int[15];
for( int i=0; i<values.length; i++) {
    values[ i] = i;
}
int answer = 0;
for( int i=0; i<values.length; i++) {
    if( i%2 == 1) {
        answer = answer + values[ i];
    }
}
System.out.println("Result is: " + answer);
```

a) Result is: 49  
b) Result is: 56  
c) Does not compile  
d) Compiles, but gives a run-time error  
e) None of the above

32. What is the return value of method `checkIt()` shown below?

```java
int checkIt() {
    int value = 0;
    int[] theNumbers = new int[7];
    for( int i=0; i<theNumbers.length; i++) {
        theNumbers[ i] = i * 2;
        value = value + theNumbers[ i];
    }
    return value;
}
```

a) 84  
b) 63  
c) 42  
d) Program does not compile.  
e) None of the above
33. What is the output of the program segment below when method `checkColor()` is called? Assume the `Piece` class is similar to that used in program 3.

```java
void checkColor()
{
    Piece[] theBoard = new Piece[8];
    if( theBoard[0].getColor().equals("white") ) {
        System.out.println("It is white.");
    }
    else {
        System.out.println("NOT white.");
    }
}
```

a) It is white.
b) NOT white.
c) Program does not compile
d) Program compiles, but it gives a run-time error.
e) None of the above

34. Assume that you create class `Employee` that includes an instance of class `Date` to store the startDate for each employee. Assume that you have written some test code in class `EmployeeDriver` shown below, where you change the startDate for `e1`. To your surprise when you run this code the startDate for `e2` has changed as well. What is the most likely explanation for this?

```java
class EmployeeDriver
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee;
        Employee e2 = new Employee;

        e1.changeDate( 9, 30, 1923);
        System.out.println(e1);
        System.out.println(e2);
    }
}
```

a) The Date class fields are declared as `static`
b) The Employee copy constructor does not create a new Date
c) There is no copy constructor for the Date class
d) The new value happens to be the same as the default value
e) None of the above
35. What is the output of the code given in the two columns below when an object of type `Confuse` is created and used to call method `startUp()`?

<table>
<thead>
<tr>
<th>class Confuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>private int x;</td>
</tr>
<tr>
<td>private int y;</td>
</tr>
<tr>
<td>public Confuse()</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>x = 3; y = 5;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>private void first(int y)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>x = y; x++;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>private void second(int s, int t)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>setXY( (y+s), (x-t));</td>
</tr>
<tr>
<td>s = 3; t = 1;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

| private void setXY( int s, int y) |
| { |
| x = s; |
| this.y = y; |
| first( y); |
| } |
| private void display() |
| { |
| System.out.println(x + y); |
| } |
| public void startUp() |
| { |
| first( x+1); |
| second( y, x); |
| display(); |
| } |

| a) 1 |
| b) 3 |
| c) 5 |
| d) 8 |
| e) None of the above |

36. What is the output of the code given below when `problem36Driver()` is called?

| public void problem36Driver() |
| { |
| String phrase = "eschew surplussage"; |
| // create character array from String |
| char[] theArray = phrase.toCharArray(); |
| prob36( theArray); |
| // display array contents |
| System.out.print( theArray); |
| } |

| public void prob36(char[] w) |
| { |
| int x = w.length; |
| char c; |
| for (int i=0; i<x/2; i++) { |
| c = w[i]; |
| w[i] = w[x-i-1]; |
| w[x-i-1] = c; |
| } |

| a) The contents of the original array in reverse order |
| b) The contents of the original array in the original order |
| c) The contents of the original array with half of the characters reversed |
| d) The original array with characters rearranged so they are neither in the original nor reversed order |
| e) None of the above |
37. How many times would method swap get executed when bubbleSort is called using the following:

```java
public void bubbleSort(int[] theArray) {
    int pass, current;

    for (pass=1; pass < theArray.length; pass++) {
        for (current=0; current < theArray.length-pass; current++) {
            if (theArray[current] < theArray[current+1]) {
                swap(theArray, current, current+1);
            }
        }
    }
}
```

```java
public void swap(int[] theArray, int i, int j) {
    int temp = theArray[i];
    theArray[i] = theArray[j];
    theArray[j] = temp;
}
```

a) 7
b) 9
c) 10
d) 12
e) None of the above
T F 38. The call to method \texttt{tryIt} prints:

```
public String convert( String original)
{
    String newString = "";
    char c;
    for( int i=0; i<original.length(); i++) {
        c = original.charAt(i);
        if( (c>=65) && (c<=90)) {
            c = (char)(c + 32);
        }
        newString = newString + c;
    }
    return newString;
}
```

```
public void tryIt()
{
    System.out.println("Result is: " + convert("YES"));
}
```

\begin{itemize}
\item[a)] Result is: YES!
\item[b)] Result is: yes!
\item[c)] Result is: yes
\item[d)] does not compile
\item[e)] None of the above
\end{itemize}

39. What is the output of the following code when it is called with \texttt{System.out.println( methodB(2,5) )};

```
public int methodA( int x, int n)
{
    if (n == 0)
    
    else
        return x + methodA( x, n-1);
}
```

```
public int methodB( int x, int n)
{
    if (n == 0)
    
    else if (n > 0)
        return methodA( methodB( x, n - 1), x);
    return 0;
}
```

\begin{itemize}
\item[a)] 7
\item[b)] 10
\item[c)] 25
\item[d)] 32
\item[e)] None of the above
\end{itemize}
40. What is the output of the code given below when called with:

```java
ClassE classEInstance = new ClassE( 3);
```

```java
class ClassD {
    private int x;
    public ClassD(int value) {
        x = value;
        method1(x);
    }
    private void method1(int x) {
        x++;
        ClassE instance1 = new ClassE(x);
    }
} // end of classD

class ClassE {
    private int x = 2;
    public ClassE(int x) {
        if (x < 5) {
            ClassD instance1 = new ClassD(x);
        }
        x++;
        System.out.print(" "+x);
    }
    public void method1(int x) {
        x = x + 2;
    }
} // end of classE
```

a) 3 4  
b) 6  
c) 5 4 3  
d) 6 5 4  
e) None of the above.