

CS 102 - Introduction to Programming
Midterm Exam #2 - Prof. Reed
Fall 2009

What is your name?: _____

There are three sections:

- I. True/False50 points; (25 questions, 2 points each)
- II. Multiple Choice50 points; (10 questions, 5 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 *turnin* command for a project will overwrite any previous files you previously submitted for that project.

T F 2. The following is a valid declaration in Java: `boolean if = true;`

T F 3. The following statements compile and run in Java:

```
String one = "two";  
if( one == "two") {  
    System.out.println("Are same");  
}
```

instance variables

T F 4. When writing a class that is to be used by others, you should at a minimum implement the *get* and *set* methods, default constructor, fully qualified constructor, copy constructor, *equals* method, and the *toString* method.

T F 5. Two methods in Java can have the same name and same number of parameters.

T F 6. Two methods in Java can have the same name, type and number of parameters, as long as the method return types are different.

T F 7. In order for the code shown below to compile and run properly, a *copy constructor* for the Date class must exist.

```
Date d1 = new Date(2,14,2000);  
Date d2 = new Date( d1);
```

T F 8. To *chain* constructors you must use *this*.

T F 9. If a parameter has the same name as a class (instance) variable name, using that variable will by default refer to the instance variable, not to the parameter.

T F 10. If a parameter has the same name as a class (instance) variable name, we can explicitly refer to the instance variable by using *this*.

- T F 11.** Given the `Date5` class discussed in our lectures, the follow code will give the output:
NOT equal

```
Date5 d1 = new Date5( 1,1,2010);
Date5 d2 = new Date5( 1,1,2010);
if( d1==d2)
    System.out.println("Are equal");
else
    System.out.println("NOT equal");
```

- T F 12.** Assuming the `Date5` class as developed in our lectures, the `toString` method gets called automatically in the following code:

```
Date5 d3 = new Date5();
System.out.println("Date is: " + d3);
```

- T F 13.** Given the `Date5` class discussed in our lectures, the follow code will give the output:
1/31/2009

```
Date5 d1 = new Date5( 12,31,2009);
Date5 d2 = d1;
d1.setMonth(1);
System.out.println( d2);
```

- T F 14.** A *static* variable's value cannot be changed.

- T F 15.** A *static* method such as `main()` can not access non-static instance variables.

- T F 16.** A non-static method can access static instance variables.

- T F 17.** Private instance variables can only be accessed through code in the class where they are declared.

- T F 18.** The following Java code would compile and run:

```
int[] values = {1,2,3,4,5};
int x = values[ values[4] ];
System.out.println( x);
```

- T F 19.** Consider the *Square* class and *Board* class as used in our programming assignments. Assume the following code would be in the *Board* class.

```
Square s1 = new Square(30,30,60,"red",true,"");
Square s2 = s1;
s2.setColor("green");
```

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

- T F 20.** Every program can be written with all the looping code implemented using only *for* loops.

- T F 21.** Every program can be written with all the looping code implemented using only *while* loops.

- T F 22.** Every program can be written with all the looping code implemented using only *do* loops.

- T F 23.** Every program can be written with all the looping code implemented using *recursion*.

- T F 24.** The `length()` method is used for both strings and arrays. You must be sure to include the parenthesis in both cases.
- T F 25.** Assume the code shown below, where method `swapValues2` is called.
Output of this segment of code is: Values are: abcd

```
char[] letters = {'a','c','b','d'};
swapValues2( letters, 1, 2);
System.out.print("Values are: ");
System.out.println( letters);
// for the line above, think of class Monday...

public void swapValues2(char[] a, int num1, int num2)
{
    char temp = a[num1];
    a[num1] = a[num2];
    a[num2] = temp;
}
```

II. Multiple Choice (5 pts. each)

- 26.** Consider the method given below. It can best be described as a method that:

```
public boolean method26( int number)
{
    if( number==1) return false;
    if( number==2) return true;

    for (int i=2; i <= number/2; i++) {
        if ( number%i == 0) {
            return false;
        }
    }
    return true;
}
```

- a) checks to see if a number is even
- b) checks to see if a number is odd
- c) checks to see if a number is prime
- d) does not compile
- e) None of the above

27. Consider the method given below. What would be the output if this were called using the statement:
`System.out.println(method27(1357));`

```
int method27( int number)
{
    int x = number;
    int y = 0;
    while ( x > 0) {
        y = y * 10 + x%10;
        x = x / 10;
    }
    return y;
}
```

- a) 1
- b) 7
- c) 35
- d) 7531
- e) None of the above

28. Consider the code shown at right below, that uses the Square class demonstrated during class. What does the output of this code look like?

<ul style="list-style-type: none"> a) A square that itself moves in a clockwise inward shrinking spiral. b) A square that itself moves in a clockwise constant size circle. c) A square that itself moves in a counter-clockwise constant size circle. d) A square that itself moves in a counter-clockwise outward growing spiral. e) None of the above. 	<pre>public class Picture { public static void main(String[] args) { Square firstSquare = new Square(); firstSquare.makeVisible(); for (int i=31;i>0;i--) { firstSquare. slowMoveVertical(i*2); firstSquare. slowMoveHorizontal(i*2); firstSquare. slowMoveVertical((-i)*2); firstSquare. slowMoveHorizontal((-i)*2); }//end for(int i... }//end main... } } //end class Picture</pre>
--	--

32. Consider the class given below, along with the driver class for it.

<pre>class ClassB { private int value; public ClassB() { value = 1; } public void setValue(int value) { value = value + 2; } } //end ClassB</pre>	<pre>class ClassBDriver { public void doIt() { int number = 2; ClassB instance1 = new ClassB(); instance1.setValue(number); number = instance1.value; System.out.println("value is: "+number); } } //end ClassBDriver</pre>
---	--

When running method doIt() in the ClassBDriver class, what is the result? output will be:

- a) Output will be: value is: 1
- b) Output will be: value is: 2
- c) Output will be: value is: 4
- d) Doesn't compile
- e) None of the above

33. 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?

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

        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) The Date class copy constructor does not chain to the fully qualified constructor
- d) The new value happens to be the same as the default value
- e) None of the above

34. What is the output of the code given below when method `useArray()` is called?

<pre>public void useArray() { char[] theArray= {'E','n','t','r','o','p','y'}; processArray(theArray); for(int i=0; i<theArray.length; i++) System.out.print(theArray[i]); }</pre>	<pre>public void processArray(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; } }</pre>
--	---

- 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

35. What is the output of the code below when `bubbleSort` is called using the following:

```
int[] values = {1,5,7,8,9,13,15};
bubbleSort( values);
```

```
void bubbleSort( int[] theArray)
{
    int counter = 0;
    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);
                counter++;
            }
        }
    } //end for ( current...
} //end for (pass...

System.out.println( counter);
} //end method bubbleSort

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

- a) 7
- b) 19
- c) 20
- d) 21
- e) None of the above