What is your name? ____________________________ (4 points)

This test has the following sections:
I. True/False.......................... 60 points; (30 questions, 2 points each)
II. Multiple Choice................... 36 points; (6 questions, 6 points each)

96 + 4 points for name = 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) On your bubble form fill out a for true and b for false.

T  F  1. Any program written with a for loop could be rewritten using either a while loop or a do-while loop.

T  F  2. Any code that can be written with multiple if-else statements could also be written with a switch-case statement.

T  F  3. Java variable names may begin with an alphabetic character or a number.

T  F  4. Private instance variables can only be accessed directly through code in the class where they are declared.

T  F  5. If private instance variables were made public, we would not need the get and set methods.

T  F  6. Two methods in Java can have the same name and same number of parameters.

T  F  7. Two methods in Java can have the same name, same number of parameters with the same types as long as the return type is different.

T  F  8. If toString() is not implemented in a class, then printing a class object displays the object’s class name and address in memory.

T  F  9. A static variable’s value cannot be changed.

T  F  10. The section of code shown below would compile and run and give as output: 

```java
int sum = 0;
for( int i=0; i<3; i++)
    sum = sum + i;
System.out.println( sum);
```
11. To see if variable $x$ is equal to or greater than variable $y$, we could use:

```java
if( x >= y) {
    System.out.println("x >= y");
}
```

12. The following code gives as output:

```java
for( int i=0; i<7; i++) {
    System.out.println( i + " ");
}
```

13. The output of the code below is:

```java
System.out.println( 2 + 3 + "skidoo");
```

14. The output of the code below is:

```java
System.out.println( "2" + 3 + "skidoo");
```

15. The output of the code below is:

```java
System.out.println( "" + 2 + 3 + "skidoo");
```

16. The output of the following code is: Smaller Done

```java
int x = 4;
if( x > 5)
    System.out.println("Bigger ");
else
    System.out.println("Smaller ");
System.out.println("Done ");
```

17. The section of code shown below would compile and run and give as output: < 5 End

```java
int x = 3;
if( x < 5)
    System.out.println("< 5 ");
if( x < 7)
    System.out.println("< 7 ");
System.out.println("End");
```

18. The section of code shown below would compile and run and give as output: Not Done End

```java
boolean notDone = false;
if( notDone = true)
    System.out.println("Not Done ");
else
    System.out.println("Done ");
System.out.println("End");
```
19. The output of the following code is: Different

```java
String word1 = "Hello";
String word2 = "Hello";
if (word1 == word2)
    System.out.println("Same ");
else {
    System.out.println("Different");
}
```

20. The following statements compile and run in Java:

```java
String last = "first";
if (last.equals( "first")) {
    System.out.println("first = last");
}
```

21. The following statements compile and run in Java:

```java
char[] letters = {'Y', 'e', 's'};
for (int i=0; i<letters.length; i++) {
    System.out.println( letters[ i]);
}
```

22. The following code prints: 2 Day

```java
int x = 2;
switch (x){
    case 1: System.out.print("1");
    case 2: System.out.print("2");
    case 3: System.out.print("3");
    case 4: System.out.print("4");
}
System.out.println("Day");
```

23. In order for the code shown below to compile and run properly, a copy constructor for the Date class must have been created by the user.

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

24. A chained constructor call can be used anywhere within the calling constructor code.

25. The following code will compile and run:

```java
int[] theArray = {1,3,5,7};
System.out.println("Array is: " + theArray);
```

26. Assuming `keyboard` is a properly declared instance of `Scanner`, the following code will compile and run:

```java
System.out.print("Enter the size: ");
int size = keyboard.nextInt();
int[] theArray = new int[ size];
```
27. Once an array is created Java will automatically grow the array if we add only a single element at a time to the end of the array. **F**

28. All the elements of an array in Java must be of the same type. **T**

29. Given the final version of the *Date* class discussed in our lectures, the follow code will give the output: 
```
Are equal
```
```
Date d1 = new Date(1,1,2010);
Date d2 = new Date(d1);
if(d1==d2)
    System.out.println("Are equal");
else
    System.out.println("NOT equal");
```

30. Assuming the final version of the *Date* class as developed in our lectures, the *toString* method gets called automatically in the following code: 
```
Date d3 = new Date();
System.out.println("Date is: " + d3);
```
II. Multiple Choice (6 pts. each)

31. Consider the code shown below. What does this method do?

```java
void display()
{
    for (int i=1; i<=10; i++) {
        for (int j=1; j<=10; j++) {
            System.out.printf("%4d", i*j);
        }
        System.out.println();
    }
}
```

a) displays part of a multiplication table  
   b) displays a full multiplication table with all the columns lined up  
   c) displays a full multiplication table, though all the columns are not lined up  
   d) displays values from a multiplication table, all in one long row  
   e) None of the above

32. Consider the code shown below. What does this method return?

```java
void changeUp( char[] letters) {
    int size = letters.length;
    for( int i=0; i<size/2; i++) {
        char temp = letters[i];
        letters[ i] = letters[ size - i - 1];
        letters[ size - i - 1] = temp;
    }
}
```

a) the letters from parameter letters in reverse order  
   b) the letters from parameter letters in their original order  
   c) the letters from parameter letters in reverse order when the length of letters is odd  
   d) the letters from parameter letters in reverse order when the length of letters is even  
   e) None of the above

33. Consider method second shown at right, which itself uses method first. For positive numbers, how would you best describe its return value?

```java
public int first( int x, int y) {
    int z=x;
    for( int i=0; i<y; i++) {
        z = z - 1; //subtract 1 from x y times
    }
    return z;
}
```

```java
public int second( int x, int y) {
    int z=x;
    for( int i=0; i<y; i++) {
        z += 1; //odd 1 to x y times
    }
    return first( z, y);
}
```

a) $x + y$  
   b) $x \times x$  
   c) $x \times y$  
   d) $x$ equals $x$  
   e) None of the above
34. Consider the class given below, along with the driver class for it.

```java
class ClassA {
    private int x;

    public ClassA() {
        x = 4;
        addValue(x);
    }

    public void addValue(int val) {
        x = x + val;
    }
}

class ClassADriver {
    public void doIt() {
        int value = 2;
        ClassA instance1 = new ClassA();
        instance1.addValue(value);
        System.out.println("value is: " + instance1.x);
    }
}
```

When running method `doIt()` in the `ClassADriver` class, the output will be:

- a) value is: 6
- b) value is: 7
- c) value is: 8
- d) doesn’t compile
- e) None of the above

35. 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);
        e1.changeDate(11, 1, 2010);
        System.out.println(e1);
        System.out.println(e2);
    }
}
```

- a) The Date class fields are declared as `static`
- b) The Date class copy constructor does not chain to the fully qualified constructor
- c) The new value happens to be the same as the default value
- d) The Employee copy constructor does not create a new Date
- e) None of the above
36. Assume an instance of class *Confusing* is created, and that instance is used to call method *doIt()*. What is the output?

- a) Answer is: 1
- b) Answer is: 2
- c) Answer is: 3
- d) Answer is: 5
- e) None of the above

37. What is the output of the following line of code:

```java
System.out.print( mystery( 1234567, 1 ) );
```

where method *mystery( )* is shown below

```java
int mystery( int x, int y)
{
    if( x==0)
        return 0;
    if( y%2 == 0)
        return mystery( x/10, y+1)*10 + x%10; //when counter even
    else
        return mystery( x/10, y+1); //when counter all
}
```

- a) 123456
- b) 1357
- c) 246
- d) 1
- e) None of the above