This test has the following sections:
I. True/False ....................... 60 points; (30 questions, 2 points each)
II. Multiple Choice............... 36 points; ( 9 questions, 4 points each)

---------------
96 + 4 points for free = 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.

True/False: (2 points each) On your bubble form fill out a for true and b for false.

T F 1. The following variable declaration would compile: char Int;

T F 2. If a Java program needs to access instance variables defined in some other class, a good way to do this is to make those instance variables public.

T F 3. The section of code shown below would compile and run and give as output: 6 Done

```
int x = 2;
int y = 4;
System.out.println(x + y + " Done");
```

T F 4. The section of code shown below would compile and run and give as output: Yes Done

```
boolean z = false;
if( z = true)
    System.out.print("Yes ");
else
    System.out.print("No ");
System.out.println("Done");
```

T F 5. The following code prints the words: Hey there

```
char c='H';
switch (c){
    case 'H': System.out.print("H");
    case 'e': System.out.print("e");
    case 'y': System.out.print("y");
    break;
}
System.out.println(" there");
```
6. The output of the following code is: Larger Done

```java
int x = 7;
if( x < 5 ) {
    System.out.print("It is ");
    if( x < 3 ) {
        System.out.print("less than 3 ");
    } else {
        System.out.print("Larger ");
    }
} else {
    System.out.print("Done ");
}
```

7. The following code prints the words: Hey there

```java
char c="H";
switch (c){
    case 'H': System.out.print("H");
    case 'e': System.out.print("e");
    case 'y': System.out.print("y");
    break;
}
System.out.println(" there");
```

8. Any program written with a while loop could be rewritten using a for loop.

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

10. Java variable names may begin with a number.

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

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

13. Two methods in Java can have the same name and same number of parameters.

14. 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.

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

16. A static variable’s value cannot be changed.

17. To see if int variable x is equal to or greater than int variable y, we could use:

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

18. The following code gives as output: 0 1 2 3 4 5

```java
for( int i=0; i<=5; i++ ) {
    System.out.println( i + " ");
}
```
19. The section of code shown below would compile and run and give as output:

```java
boolean notDone = false;
if( notDone = true )
    System.out.print("Not Done ");
else
    System.out.print("Done ");
System.out.println("End");
```

20. The following statements compile and run in Java:

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

21. The following code prints:

```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");
```

22. 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);
```

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

24. The following code will compile and run and display the contents of the array:

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

25. 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];
```

26. 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.

27. All the elements of an array in Java must be of the same type.

28. For an array with 10 elements, accessing the element with index 10 gives an error.
29. Given the final version of the *Date* class discussed in our lectures, the follow code will give the output:

```java
Date d1 = new Date(12, 20, 2012);
Date d2 = new Date(d1);
if (d1==d2) System.out.println("Same");
else System.out.println("NOT Same");
```

20

30. Assuming the final version of the *Date* class as developed in our lectures, the *toString* method gets called automatically in the following code:

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

Multiple Choice (4 pts. each)

31. Consider the code shown below. What does this method return for a String with an even number of letters?

```java
String modify( String aWord)
{
    char c, d;
    String word1 = "";
    String word2 = "";

    for( int i=0; i<aWord.length()/2; i++) {
        c = aWord.charAt( i);
        d = aWord.charAt( aWord.length() - i-1);
        System.out.println(c + " " + d);
        word1 = c + word1;
        word2 = word2 + d;
    }
    return word1 + word2;
}
```

a) The letters from aWord in reverse order  
b) The first half of aWord followed by the second half of aWord in reverse order  
c) The first half of aWord in reverse order followed by the second half of aWord  
d) The first half of aWord in reverse order followed by the second half of aWord in reverse order  
e) None of the above
For each of the 3 methods listed below, for positive numbers how would you best describe its return value? (Note that some of the methods in turn use other methods.)

32. Method first
   a) x
   b) x + y
   c) x * x
   d) x^y
   e) None of the above

33. Method second
   a) x
   b) x + y
   c) x * x
   d) x^y
   e) None of the above

34. Method third
   a) x
   b) x + y
   c) x * x
   d) x^y
   e) None of the above

35. Consider the code shown below. What is the output of this method?
   ```java
   public int first( int x, int y) {
   int z=x;
   for( int i=0; i<y; i++) {  
   z = z + 1;
   }
   return z;
   }
   ```
   ```java
   public int second( int x, int y) {
   int z=0;
   for( int i=0; i<y; i++) {
   z = first(z,x);
   }
   return z;
   }
   ```
   ```java
   public int third( int x, int y) {
   int z=1;
   for( int i=0; i<y; i++) {
   z = second(z,x);
   }
   return z;
   }
   ```
   a) Echoes the original input with each character transposed a fixed amount
   b) Echoes only the vowels from the original input
   c) Echoes only non-vowels from the original input
   d) Echoes only the consonants from the original input
   e) None of the above

```java
void change( String aWord)
{
   aWord = aWord.toUpperCase();
   for( int i=0; i< aWord.length(); i++) {
      char c = aWord.charAt( i);
      int x = c - 'A'; //convert down to range 0...25
      if( x==0 || x==4 || x==8 || x==14 || x==20) {
         System.out.print((char)(x + 65));
      }
   }
}
```
36. Consider the class given below, along with the driver class for it.

```java
class ClassA {
    private int x;

    public ClassA() {
        x = 3;
        increment(x);
    }

    public void increment(int val) {
        for (int i = 0; i < val; i++) {
            x = x + 1;
        }
    }

    public int getX() { return 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

37. 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
38. Assume an instance of class Confusing (shown below) is created, and that instance is used to call method `doIt()`. What is the output?

a) 4  
b) 6  
c) 7  
d) 8  
e) None of the above

```java
class Confusing {
    int x = 5;
    int y = 1;

    int first(int x) {
        y = x + 1;
        return y + 1;
    }

    void second(int x, int y) {
        y = first(x);
        this.y = y + 1;
    }

    public void doIt() {
        int x = first(y);
        second(y, x);
        System.out.print(x + y);
    }
}
```

39. Assume that in the context of a larger program we have the code shown at right. What is the output when `trickyDriver()` is called? (Note: this problem is difficult because it uses ideas we have not discussed yet in class.)

a) 4  
b) 7  
c) 8  
d) Does not compile  
e) None of the above

```java
void trickyDriver() {
    int[] numbers = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0};
    System.out.println(tricky(numbers, 8));
}

int tricky(int[] numbers, int index) {
    int result = 1;
    numbers[index] = 2;
    int[] offsets = {-1, -5, 1, 5};
    for (int i = 0; i < offsets.length; i++) {
        int next = index + offsets[i];
        if (numbers[next] == 1) {
            result += tricky(numbers, next);
        }
    }
    return result;
}
```