You may take this test with you afterwards, but you must turn in your answer sheet.

This test has the following sections:
I. True/False .......................... 64 points; (32 questions, 2 points each)
II. Multiple Choice................. 36 points; (6 questions, 6 points each)
--------------------------------- 100 points total

This test is worth 10% 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.

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

1. A class named _FunCode_ should be stored in a file called _FunCode.java_  
2. A Java application should have a method called _main_ to start running the program.  
3. A variable that is declared as a _constant_ in Java has the keyword _const_ placed in front of the type name, such as in this example: _const int x = 7;_  
4. Consider code used to display a table of numbers where some are 1 digit numbers, some are 2 digit numbers and some are 3 digit numbers, where we want the numbers to be right-justified and lined up in columns. (An example of this is a 10 x 10 multiplication table.) The only way to do this in Java is to use _if_ statements to test the magnitude of each number to be printed, printing extra spaces for smaller numbers.  
5. A single _System.out.print_ statement can be used to print 3 lines of output.  
6. Three _System.out.print_ statements can be used to print a single line of output.  
7. Assuming _keyboard_ is declared as a _Scanner_, then we typically would use _keyboard.nextLine()_ to read in an integer number.  
8. To retrieve a single character from a _String_ we would typically use the _valueAt_ method.  
9. A _char_ is stored internally as an integer value in Java.  
10. A _char_ variable can always be used interchangeably as-is with any _int_ variable in Java code. 
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");
   }
   ```

   Should be _if_ (x >= y) ...
12. The following code can be used to swap the values stored in variables \( x \) and \( y \).

\[
\begin{align*}
\text{int } x &= 3; \\
\text{int } y &= 7; \\
x &= y; \\
y &= x;
\end{align*}
\]
The 3 gets lost unless an extra variable is used.

13. Indentation in a Java program can help the Java compiler figure out what some code means where otherwise it would be confused.

14. Single-line style comments using `//` can be nested inside block-style comments using `/* */`.

15. If two methods had the return types, names and parameters as shown below it would cause a compiler error. Assume the other parts of the methods exist and are written correctly.

\[
\begin{align*}
\text{public int f1()}
\end{align*}
\]

\[
\begin{align*}
\text{public int f1( char c)}
\end{align*}
\]

16. If two methods had the return types, names and parameters as shown below it would cause a compiler error. Assume the other parts of the method exist and are written correctly.

\[
\begin{align*}
\text{int f3( int x)}
\end{align*}
\]

\[
\begin{align*}
\text{int f3( int y)}
\end{align*}
\]

17. If two methods had the return types, names and parameters as shown below it would cause a compiler error. Assume the other parts of the method exist and are written correctly.

\[
\begin{align*}
\text{int f4( int x)}
\end{align*}
\]

\[
\begin{align*}
\text{char f4( int x)}
\end{align*}
\]

18. Any Java program that could be written using methods could be rewritten with all the code in `main()`.

19. The following variable declaration would compile: `char Int;`

20. The following variable declaration would compile: `char word = "a String, not a char`.

21. To end a Java program the `break` statement may be used.

\[
\text{System.exit(0);}\]

22. A loop that pulls out each individual character in order from a String would best be implemented as a `for` loop.

23. A loop that displays menu options and handles user input would best be implemented using a `do` loop.

\[
\text{It executes at least once.}\]

24. Extensive comments in a program make it more clear for us as humans, but excessive comments will actually slow down a program.

25. If a Java program needs to access instance variables defined in a class, a good way to do this is to make those instance variables `public`.

\[
\text{Make them private and use get and set methods.}\]
26. The section of code shown below would compile and run and give as output: 
   6 Done
   ```java
   int x = 2;
   int y = 4;
   System.out.println(x + y + " Done");
   ```

27. The section of code shown below would compile and run and give as output: Done 6
   ```java
   int x = 2;
   int y = 4;
   System.out.println(" Done" + x + y);
   ```

28. The section of code shown below would compile and run and give as output: 2 Done
   ```java
   int x = 8;
   int y = 3;
   System.out.print( x/y);
   System.out.println(" Done");
   ```

29. The code section below would compile and run and give as output: Result: 16
   ```java
   int x = 3;
   int y = 5;
   int z = x * y / 2;
   System.out.println("Result: " + z);
   ```

30. The section of code shown below would compile and run and give as output: Yes Done
    ```java
    boolean z = false;
    if (z = true) {
        System.out.print("Yes ");
    } else {
        System.out.print("No ");
    }
    System.out.println(" Done");
    ```

31. 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");
    ```

32. 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 ");
    System.out.print(" Done");
    ```
II. Multiple Choice (6 pts. each)

31. Consider method looping1 shown below. For positive numbers, how would you best describe its return value?

```java
int looping1(int x, int y) {
    int answer = 0;
    for (int s = 0; s < x; s++) {
        for (int z = 0; z < y; z++) {
            answer++;
        }
    }
    return answer;
}
```

- a) x + y
- b) x * x
- c) x * y
- d) x
- e) None of the above

32. Consider method looping2 shown below. For positive numbers, how would you best describe its return value?

```java
int looping2(int x, int y) {
    int answer = 0;
    while (x + y < x * y) {
        answer = answer + x;
        y = y - 1;
        x = x + 1;
    }
    return answer;
}
```

- a) x + y
- b) x * x
- c) x * y
- d) x
- e) None of the above

33. Consider a situation where a program creates an instance of the user-created Date class and prints out a Date inside a System.out.println() statement. The output doesn't look like a normal Date. What is the best reason why it doesn't look right?

- a) Values have not been initialized properly for day, month and year
- b) There is no default constructor
- c) Java can't differentiate between the user-defined and the built-in Date class
- d) There is no user-defined toString() method defined
- e) None of the above
34. Consider the code shown below. What does this method return for a String with an even number of letters?

```
String modify(String aWord)
{
    char c, d;
    String word1 = "";
    String word2 = "";
    for (int i=aWord.length()/2; i>0; i--)
    {
        c = aWord.charAt(i);
        d = aWord.charAt(aWord.length()-i);
        word1 = word1 + c;
        word2 = d + word2;
    }
    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

35. Consider the code shown below. For upper-case input what does this method do?

```
void change(String aWord)
{
    aWord = aWord.toUpperCase();
    for (int i=0; i<aWord.length(); i++)
    {
        char c = aWord.charAt(i);
        int x = c - 'A';
        if (x!=0 && x!=4 && x!=8 && x!=14 && x!=20)
        {
            System.out.print((char)(x + 65));
        }
    }
}
```

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
36. Assume an instance of class Confusing is created, and that instance is used to call method \textit{doIt()}. What is the output?

```
class Confusion
{
    int x = 5;
    int y = 2;

    int first(int x) {
        int temp = y + 1;
        y = x;
        x = temp;
        return x;
    } // end method first

    void second(int x, int y) {
        first(x); // return value ignored
        this.y = y + 1;
    } // end method second

    public void doIt() {
        x = first(y);
        second(y, x);
        System.out.print("Answer is: "+(x+y));
    } // end method doIt()
}
```