SOLUTIONS

Fill in the Blanks (20 points – 2 points each). Provide the MOST appropriate answers.

a) The next() method in the Java Scanner class will read a String up until an occurrence of the _______ delimiter ________, which by default is "white space".

b) The Java compiler will convert the source code in a ______.java ________ (give extension) file into byte code in a ______.class ________ (give extension) file.

c) The ______ associativity _______ of Java operators determines the order of evaluation when the operators have the same precedence.

d) Give the value for X after the execution of the Java statements below:
   ```java
double X, Y = 4.0, Z = 2.0;
   X = 16.0 / Y / Z - 2;
   X ___ 0 _____
   ```

e) Give the values for J and K after the execution of the Java statements below:
   ```java
   int J = 6;
   int K = --J;
   J ___ 5 _____ K ___ 5 _____
   ```

f) The scope of Java method variables is the point of declaration to the end of the ______ block _______ in which they are declared.

g) The Java code segment:
   ```java
   int X;
   X = 10.0/2;
   ```

   will generate the following error: ______ possible loss of precision __________

h) List the Java relational operators _______ <, <=, >, >=, ==, != _______.

i) (Choose the correct answer) The "dangling else" in Java is an example of a [syntax error, run-time error, logic error] _______ logic error __________.

j) (Choose the correct answer) Using a Java keyword as a variable name is an example of a [syntax error, run-time error, logic error] _______ syntax error __________.
True / False (14 points – 2 points each). Explain false statements for full credit.

a) Java identifiers that are **class names** must begin with an **upper case letter** while Java identifiers that are **method names** must begin with a **lower case letter**.   **False -- These are guidelines but not required.**

b) Java .class files execute directly on the computer’s hardware.   **False -- They execute using the interpreter (Java Runtime Environment)**

c) The Java code segment below will output "1 2 3 4 5"
   ```java
   for (int X = 1; X < 5; X++)
      System.out.print(X + " ");
   System.out.println(X);
   ```
   **False – the code will cause an error "Identifier X not found" since the scope of X is only the for loop.**

d) The Java code segment below will output "Equal":
   ```java
   int X = 3;
   double Y = 7/2;
   if (X == Y)
      System.out.println("Equal");
   ```
   **True**

e) The Java code segment below will output "**UnoDosTres**":
   ```java
   int test = 1;
   switch (test)
   {
      case 1: System.out.print("Uno");
      case 2: System.out.print("Dos");
      case 3: System.out.print("Tres");
   }
   System.out.println();
   ```
   **True**

f) A Java **for loop** will execute the loop body **1 or more times**.   **False -- if the condition is false the body may not execute at all.**

g) By the idea of **functional abstraction**, a user of a Java method does **not need to know** how the method is implemented.   **True**
Tracing (16 points) Give all output produced by the execution of the Java program below. Be careful to note the print() vs. println() statements. Use the bottom of the page for your output and clearly mark it by drawing a box around it. If you need extra space use the back of this page.

```java
public class Trace1B
{
    public static void wacky(int one, int two)
    {
        for (int i = one; i <= two; i++)
        {
            int tot = 0;
            System.out.print("Tot is ");
            int ctr = i;
            while (ctr <= two)
            {
                System.out.print(tot + ", ");
                tot += ctr;
                ctr++;
            }
            System.out.println(tot);
        }
        int temp = one;
        one = two;
        two = temp;
        System.out.println("one is " + one + " and two is " + two);
    }

    public static void logical(int p, int q, int r)
    {
        boolean b1 = (q <= p) && (p <= r);
        boolean b2 = !(q <= p) && (p <= r);
        boolean b3 = (q > p) || (p > r);
        System.out.print("b1 is " + b1 + " and b2 is " + b2);
        System.out.println(" and b3 is " + b3);
    }

    public static void main(String [] args)
    {
        System.out.println("Starting trace");
        int p = 6, q = 8, r = 4;
        logical(p, q, r);
        p = 6; q = 4; r = 8;
        logical(p, q, r);
        int low = 5, high = 8;
        wacky(low, high);
        System.out.println("low is " + low + " and high is " + high);
    }
}

Starting trace
b1 is false and b2 is true and b3 is true
b1 is true and b2 is false and b3 is false
Tot is 0, 5, 11, 18, 26
Tot is 0, 6, 13, 21
Tot is 0, 7, 15
Tot is 0, 8
one is 8 and two is 5
low is 5 and high is 8