01.12 Desk Check: AdmissionV1.java

The source code for the AdmissionV1.java program is shown below. You know the drill by now; look for the big picture, then the details.

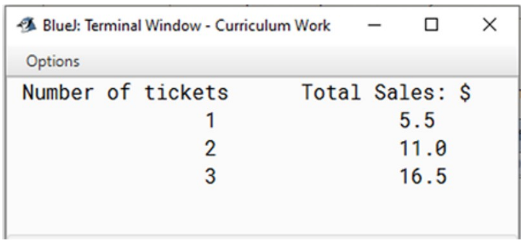
```

/**
* The purpose of this program is to count the number of
* tickets sold and the amount of money received.
*
* @author Meg Abyte
* @version 06/02/17
*/
public class AdmissionV1
{
    public static void main(String[ ] args)
    {
        //local variable
        int counter = 0;            //total tickets sold
        double ticketPrice = 0.0;   //cost of student ticket
        double totalSales = 0.0;    //total sales

        System.out.print("Number of tickets\tTotal Sales: $\n");

        //calculate total ticket sales at the game
        ticketPrice = 5.50;        //ticket price
        counter++;
        totalSales += ticketPrice;
        System.out.print("\t\t" + counter + "\t\t" + totalSales);

        //calculate total ticket sales at the game
        ticketPrice = 5.50;      //ticket price
        counter++;
        totalSales += ticketPrice;
        System.out.print("\t\t" + counter + "\t\t" + totalSales);

        //calculate total ticket sales at the game
        ticketPrice = 5.50;      //ticket price
        counter++;
        totalSales += ticketPrice;
        System.out.println("\t\t" + counter + "\t\t" + totalSales);
    }//end of main method
    }//end of class

```

The following detailed analysis of the program will help you conduct a desk check. Be sure you understand the detailed explanation of each line of code and the syntax of each statement. Two sections of code are highlighted and deserve your special attention because they use shortcut notation for the arithmetic/assignment operator and the increment operator.

Line(s)	Purpose of Statement
<1> to <8>	are comments, which the compiler ignores.
<9>	declares AdmissionV1 to be the name of the class.
<10>	is a curly brace marking the beginning of the class and matches up with the closing curly brace on Line <39>.
<11>	is the header for the **`main()`** method, the place where program execution begins.
<12>	is a curly brace marking the beginning of the main method and matches up with the closing curly brace on Line <37>.
<13>	is a comment indicating the section of code where local variables are declared and initialized.
<14>	declares **`counter`** to be a local variable of type **`int`** and assigns it the initial value of 0.
<15>	declares **`ticketPrice`** to be a local variable of type **`double`** and assigns it the initial value of 0.0.
<16>	declares **`totalSales`** to be a local variable of type **`double`** and assigns it the initial value of 0.0.
<17>	is white space to improve readability of the source code.
<18>	Prints a header for the data table—`NOTE: "/t" prints a TAB and "/n" prints a line return.`
<19>	is white space to improve readability of the source code.
<20>	is a comment indicating the purpose of the next section of code.
<21>	assigns a value to the **`ticketPrice`** variable.
<22>	increments the current value of the counter variable by 1.
<23>	adds the current value of totalSales to ticketPrice and assigns the resulting value to totalSales.
<24>	uses concatenation to print the number of tickets sold and total sales. `NOTE: "/t" prints a TAB.`
<25>	is white space to improve readability of the source code.
<26>	is a comment indicating the purpose of the next section of code.
<27>	assigns a value to the **`ticketPrice`** variable.
<28>	increments the current value of the counter variable by 1.
<29>	`adds the current value of` **`totalSales`** `to` **`ticketPrice`** `and assigns the resulting value to` **`totalSales`**`.`
<30>	uses concatenation to print the number of tickets sold and total sales. `NOTE: "/t" prints a TAB.`
<31>	is white space to improve readability of the source code.
<32>	is a comment indicating the purpose of the next section of code.
<33>	assigns a value to the `ticketPrice` variable.
<34>	increments the current value of the counter variable by 1.
<35>	`adds the current value of` **`totalSales`** `to` **`ticketPrice`** `and assigns the resulting value to` **`totalSales`**`.`
<36>	uses concatenation to print the number of tickets sold and total sales. `NOTE: "/t" prints a TAB.`
<37>	is a closing curly brace marking the end of the main() method, which matches up with the opening curly brace on Line <12>.
<38>	is a closing curly brace marking the end of the class, which matches up with the opening curly brace on Line <10>.
<39>	is white space to improve readability of the source code.

Expected Output

When you compile and run the program, you should see the following output:



```
Number of tickets       Total Sales: $
                1               5.5
                2               11.0
                3               16.5
```

Notice that 5.50 is added each time a ticket is sold and the running total of sales is displayed.

Code Analysis

The highlighted values in the following code segment show that the local variables are not only declared, but also initialized:

```
<13>            //local variables
<14>            int counter = 0;            //total tickets sold
<15>            double ticketPrice = 0.0;   //cost of student ticket
<16>            ouble totalSales = 0.0;     //total sales
```

In many cases, but not all, a compiler error will occur if a variable is used without first being initialized. Consequently, it is simply good programming practice to initialize variables when they are declared.

There are several important shortcuts to notice in the segment of code that calculates the total sales. The arithmetic/assignment operator (+=) is used in line 23 and the increment operator (++) is used in line 22.

```
<18>            System.out.print("Number of tickets\t\Total Sales: $\n");
<19>
<20>            //calculate total ticket sales at the game
<21>            ticketPrice = 5.50;     //ticket price
<22>            counter++;
<23>            totalSales += ticketPrice;
<24>            System.out|.println("\t\t" + counter + "\t\t" + totalSales);
```

Line <23> is the equivalent to the following statement:

```
totalSales = totalSales + ticketPrice;
```

In both cases, the portion of the statement to the right of the operator is evaluated then assigned to the variable on the left. The important consequence of this is that any value assigned to **totalSales will be overwritten** by the calculation on the right side of the operator. Using the shortcut arithmetic/assignment operator is completely optional, but you should get into the habit of this notation; it is considered good programming practice.

The second shortcut used in this program is the increment operator (++) as shown in line <22>. This statement is equivalent to the following code:

```
counter = counter + 1;
```

Both versions simply increase the value of the counter variable by 1. Once again, using the shortcut increment operator is optional, but it is considered good programming practice.

Modifications

As usual, the best way to extend your knowledge of programming is to play with code by making modifications and observing the results. However, be sure that when you make a modification, you understand the reasons for any changes you observe. Some changes may not make sense in the context of the program, but they will still teach you something important.

1. Which segment of code would you need to copy and paste into the program in order to simulate selling 10 tickets?
2. What happens if you change the increment operator to the decrement operator (--)?
3. What happens if you replace += with any of the other arithmetic/assignment operators(-=, *=, /=)?
4. Change the initial values of the variables and observe the results. Is there any reason you would not want to start these values at 0?
5. Are tickets to a game the same price for everyone? Change the price of a ticket for some sales to represent different prices for students, children, adults, and senior citizens using 5.50, 4.00, 6.50, and 5.00, respectively.

Be sure you complete the Desk Check of the AdmissionsV1 class carefully because you will need to apply the knowledge you gain here to the next assignment.

Increment Operators and Escape Character Reference Guide:

Operator	**Effect**	**Syntax/Example (start with n = 5)**
++	Increments a value by 1	n++; // n is now 6
--	Decrements a value by 1	n--; // n is now 5
+=	Increases a variable by a given amount	n += 3; // n is now 8
-=	Decreases a variable by a given amount	n -= 2; // n is now 6
*=	Multiplies a variable by a given amount	n*= 3; // n is now 18
%=	Modulus of a variable and a given integer	n%=4; //n is now 2

Escape Key	**Effect**
\	Prints the backslash character
\t	“prints” a TAB
\n	“prints” a line return
\"	Prints a double quotation mark

Print