HOMEWORK PROBLEMS 2

Your programs should appropriately label any output it generates. For example, if your program computes and displays the average of a set of test scores, it should not simply display the average. It should display the average with a descriptive label, such as

```
Test Score Average = 87
```

Use meaningful identifiers. For example, use `avg` not `x`, to hold an average.

Format your programs properly (use the sample programs in this textbook as models). Remember, correct indentation is absolutely essential.

1) Determine by hand the value of each of these Java expressions:

```
1 - 2 - 3 - 4 - 5 - 6 – 7 – 8 - 9 - 10
2 * 4 / 2 * 2
100 % 5 / 4
2 / 2 / 2
2 % 2 % 2
22 / 7.0
22 / 7 / 1.0
22 / (7 / 1.0)
2.1E3 + 2.4E2 + 1E1 + 1.5
```

2) What do the following statements display (assume they are embedded in a valid Java program):

```
System.out.println("2" + 3);
System.out.println(7E3);
System.out.println(7E-3);
```

3) What does the following statement display:

```
System.out.println(1 + 2 + 3 + "4" + 5 + 6);
```

4) Give a statement that declares `int` variables `i`, `j`, `k` and initializes `j` to 50. Give an assignment statement that decreases the value in `j` by 10, regardless of its current value.

5) Which of the following are legal Java identifiers: `hElLo`, `12x`, `_`, `$1234`, `bongo`

6) Determine by hand the value of each of these Java expressions. Then check your answers by writing a Java program that displays their values.

```
10%2
11%2
12%2
13%2
14%2
```

if `x%2` equals `0`, what can you conclude about `x`? `If x%2` equals 1?

7) What happens when you compile a program that contains the following statement:

```
System.out.println(5.0%2.0);
```

Is this statement illegal?

8) Compile and run the following program to determine the effect of the ++ operator.

```
 1 class C2h8
 2 {
 3    public static void main(String[] args)
 4    {
 5       int x, y;
 6       x = 1;
 7       x++;
 8       System.out.println("x++; changes x from 1 to " + x);
 9       x = 1;
10       ++x;
11       System.out.println("++x; changes x from 1 to " + x);
12       x = 1;
13       y = x++;     // increment after assignment
14       System.out.println("y = x++; changes x from 1 to " + x);
15       System.out.println("y = x++; assigns y " + y);
16       x = 1;
17       y = ++x;     // increment before assignment
18       System.out.println("y = ++x; changes x from 1 to " + x);
19       System.out.println("y = ++x; assigns y " + y);
20   }
21 }
```

++ is called the increment operator. The timing of the incrementation depends on whether ++ follows or precedes a variable. In line 13 above, the value in x is incremented *after* the assignment to y. In line 17, the value in x is incremented *before* the assignment to y.

Replace every occurrence of ++ in the code above with decrement operator --. Run the program to determine the effect of this change. In the statement on line 13, the expression ++x has a **side effect**. That is, in addition to providing a value which is then assigned to y, it does something else. In the case of x++, it increments x.

9) Incorporate the code below in a program and run. Do the first two println statements display different values for x? Do the last two println statements display different values for y? Explain the results.

```
int x, y;
x = 1;
y = ++x;
System.out.println("x = " + x + " y = " + y);
x = 1;
y = x + 1;
System.out.println("x = " + x + " y = " + y);
y = 1;
y = x + x++;      // avoid writing confusing statements like this one
System.out.println(y);
x = 1;
y = x++ + x;      // avoid writing confusing statements like this one
System.out.println(y);
```

10) Write a Java program in which you declare the `int` variables `x`, `y`, and `sum.` Assign `2` to `x` and `3` to `y`. Add `x` and `y` and assign the result to `sum`. Display the value of `sum`. When you run your program, you should see displayed on the screen

```
sum = 5
```

11) Explain what happens when you compile the following program:

```
class C2h11
{
   public static void main(String[] args)
   {
      int x, y;
      x = y + 1;
   }
}
```

12) The following statement contains a minus sign.

```
y = -x + 10;
```

This minus sign is not the subtraction operator. It is the **negation operator**. It negates (i.e., changes the sign) of the operand that follows it. Because it operates on only one operand, it is called a **unary operator** ("unary" means "one"). The plus sign here is the addition operator. Because it operates on two operands, it is called a **binary operator** ("binary" means "two").

Unary operators have higher precedence than binary operands. Thus, in the statement above, a copy of the value in `x` is negated and then the result of the negation and `10` are added. Assume the value of `x` is `5`. What value is assigned to `y`? Run a test program to check your answer. What value would be assigned to `y` if the addition operator had higher precedence that the negation operator?

13) Compile the program below by entering on the command line

```
javac C2h13.java
```

Then run the program by entering on the command line

```
java C2h13 bird deer
```

The additional items on the command above—`bird` and `deer`—are **command line arguments**. They are "passed" to the `args` parameter in the `main` method. `main` can then access these command line arguments using the `args` parameter as demonstrated in the program. We will learn more about argument passing in the Chapter 5.

```
class C2h13
{
   public static void main(String[] args)
   {
      System.out.println(args[0]);  // displays bird
      System.out.println(args[1]);  // displays deer
   }
}
```

Run the program a second time by entering

```
java C2h13 cat dog
```

What is displayed this time? Now try

```
java C2h13 up
```

What's wrong with this command?

14) Run a program that contains the following two statements:

```
System.out.println("2" + 3 + 4);
System.out.println("2" + (3 + 4));
```

Why do these statements display different strings?

15) Include the following code in a Java program. What is the effect of each statement?

```
x = 1;
x += 2;
System.out.println(x);
x *= 10;
System.out.println(x);
x /= 15;
System.out.println(x);
```

`+=` means to add and then assign. `*=` means to multiply and then assign. `/=` means to divide and then assign. For example, the second statement above adds `x` and `2` and then assigns the result to `x`.

16) Run the following program. What it displayed?

```
class C2h16
{
   public static void main(String[] args)
   {
      System.out.println("1"); /*
      System.out.println("2");
      System.out.println("3"); */
      System.out.println("4");
   }
}
```

Whatever is between `/*` and `*/` is treated as a comment by the compiler. This type of comment can span multiple lines. Comments that start with `//` are single-line comments.

17) What happens when you compile the following program:

```
class C2h17
{
   public static void main(String[] args)
   {
      int switch;
      switch = 3;
   }
}
```

What is wrong with the variable `switch`?

18) What do the following statements display:

```
System.out.println(20);
System.out.println(020);
```

Hint: Integer constants that have a leading zero are treated as octal (base 8) constants. In **octal**, the weights are powers of 8 ($8^0 = 1$, $8^1 = 8$, $8^2 = 64$, …).

19) Compile and run the following program (use the class name `String`):

```
class String             // use the name String for this class
{
   public static void main(String[] args)
   {
      System.out.println("hello");
   }
}
```

This program will not run correctly because it redefines `String`. Thus, the `args` parameter now does not have the required type (it must be `String[]`, where `String` is the *original* definition of `String`). *Once you compile this program, all the programs you subsequently compile will not run* because of the redefinition of `String`. To fix this problem, delete the `String.class` file you created when you compiled the program above. To do this on a Linux, Raspberry Pi, or Macintosh system, enter

```
rm String.class
```

On a Windows system, enter

```
del String.class
```

20) If the following code legal:

```
int m = 3;
int n = m;  // can the initial value in a declaration be a variable?
```

21) Write a program that displays five command line arguments. For example, if you run your program with

```
java Hw2a u v w x y
```

your program will display

```
u
v
w
x
y
```

Hint: See homework problem 13.

22) Write a program that assigns `1.5`, `22.4`, `-44.8`, `0.333`, and `1.123` to the variables `a`, `b`, `c`, `d`, and `e`, respectively. Next, compute their sum. Assign the result to a variable named `sum`. Next, compute the average of the values in these variables. Assign the result to a variable named `average`. Display the values in `sum` and `average` with appropriate labels.

23) 5 factorial is the product of 5, 4, 3, 2, and 1. 10 factorial is the product of the integers 10 down to 1. Write a program that computes and displays the value of 5 factorial. In the same program compute and display the value of 10 factorial making use of the result you computed for 5 factorial. Be sure to display your results with appropriate labels.

24) Write a program that computes and displays with an appropriate label the value of

$$\frac{(5)(6)(7)}{(2)(3)}$$

25) The number of characters in a `String` object is returned by the `length` method in the object. For example, if `s` is a reference variable to a `String` object, then the following statement displays the number of characters in its string:

```
System.out.println(s.length());
```

Write a program that creates three `String` objects. Your program should then determine and display the length of each string. Test your program with the following strings: `"yes"`, `"yes     "`, and `""`. The string `""` is the **null string**—the string with zero characters.

26) Is the following program legal?

```
class C2h26
{
   public static void main(String[] args)
   {
      {
         int x;  // declaration inside a block
         x = 2;
         System.out.println(x);
      }
      System.out.println(x);   // ok to access x that is inside a block?
   }
}
```

27) Is there a `toLowerCase` method in string objects? Run a test program to find out.

28) Write a program that creates three `String` reference variables. Each variable should point to the *same* object. This object should contain the string `"MeMeMe"`. Display the string pointed to by each reference variable. *Hint*: Use the assignment statement to assign one reference variable to another.

29) Write a program that creates three `String` reference variables. Each variable should point to a different object. One object should contain the string `"UPPER"`, one object should contain the string `"lower"`, and one object should contain the string `"MiXeD"`. Display the string in each object. For each object, invoke the `toUpperCase` method and display the string in the new object.

30) Suppose the following statements are inserted at the end of the program in Fig. 2.3. What will they display:

```
System.out.println(s.charAt(0));
System.out.println(s.charAt(2));
System.out.println(s.indexOf('e'));
System.out.println(s.length());
System.out.println(s.equals(p));
```

Characters in a string are numbered starting with 0. The first character in a string has index 0, the second, index 1, and so on. Each of these numbers is called an `index`. The `charAt` method provides the character at the given index. Thus, `s.charAt(0)` provides the first character in the `s` string..

31) What does the program below display?

```
class C2h31
{
   public static void main(String[] args)
   {
      String s = "hellobirdbye";
      System.out.println(s);
      System.out.println(s.length());
      String t = s.substring(0,5);
      System.out.println(t);
      System.out.println(s.equals(t));
      System.out.println(t.equals("hello"));
   }
}
```

`length`, `substring`, and `equals` are methods in `String` objects. `length` returns the number of characters in the string. `substring` creates a new object that contains the specified substring of the given string. Its first argument specifies the position at which the substring starts (0 corresponds to the first character). Its second argument specifies the position in the given string that is just beyond the end of the substring. `equals` compares two strings for equality.

32) Converting Between Binary and Hex

It is trivial to convert between binary and hex once you know the binary numbers from 0000 to 1111 and their hex equivalents (see the table below). To convert a binary number to hex, break up the binary number into groups of four bits, starting from its right end. Then substitute the hex equivalent for each four-bit group. For example, to convert 11010111000001100, we first break it up into four-bit groups starting from the right side:

```
1  1010  1110  0000  1100
```

We then substitute the hex equivalent for each four-bit group:

$$\begin{tikzcd}
1 \arrow[d] & 1010 \arrow[d] & 1110 \arrow[d] & 0000 \arrow[d] & 1100 \arrow[d] \\
1 & a & e & 0 & c
\end{tikzcd}$$

Thus, 11010111000001100 binary is equal to `1ae0c` hex. To convert hex to binary, we simply substitute the four-bit binary equivalent for each hex digit. For example, to convert `a5` to binary, substitute 1010 for `a` and 0101 for `5` to get 10100101.

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	a (or A)
11	1011	b (or B)
12	1100	c (or C)
13	1101	d (or D)
14	1110	e (or E)
15	1111	f (or F)

What are the binary equivalents of the following hex numbers: `abcdef`, `a24b`, `74810`, `35069`? What are the hex equivalents of the of the following binary numbers: `11111`, `10101100`, `1111001`, `0100111`, `101011`, `110111101010`, `1101111`, `1001010`, `1000010000100001`?

33) Write a program that includes the following declarations:

```
int x = 0;
int y = 1;
int z = 100;
```

For each of the variables, your program should shift its bits to the left three positions and insert in the vacated positions three 1 bits. For example, `y` in binary contains 0…01 which becomes 0…1000 after the shift and then becomes 0…01111 after the insertion of the three 1 bits. Thus, the final value of `y` in decimal is 15. Display the final values of `x`, `y`, and `z`.

34) Write a program that includes the following declarations:

```
int x = 2;
int y = 1;
int z = 3;
```

Your program should display the *decimal* number whose *decimal* digits left to right are the `x`, `y`, and `z` digits. Do not use 213 in your `println` statement.

35) Write a program that includes the following declarations:

```
int x = 2;
int y = 1;
int z = 3;
```

Your program should display the *decimal* number whose *hex* digits left to right are the `x`, `y`, and `z` digits.