

1 Before Getting Started on the Exercises

Review the class examples we discussed this week and make sure you understand what each command and operation does. Specifically, make sure you understand the concept and syntax of methods, parameters, and return values.

Some of these exercises are simply taking programs you have written before and revising them into a program that uses methods to complete the task. In many cases the number of lines of code is decreased due to the reuse of the method in the main program. For each exercise you will be given the main program that must be used and your task is to develop the methods needed to complete the program.

For each exercise submit the Java code file containing the program and a Word, LibreOffice, or text file containing at least 3 runs of the program.

2 Exercises

1. The first program is to construct a Fahrenheit to Celsius conversion chart that will print out a chart of Fahrenheit to Celsius conversions. The output should look like the following.

Fahrenheit	Celsius
-50	-45.56
-45	-42.78
-40	-40.00
-35	-37.22
-30	-34.44
-25	-31.67
-20	-28.89
-15	-26.11
-10	-23.33
-5	-20.56
0	-17.78
5	-15.00
10	-12.22
15	-9.44
20	-6.67
25	-3.89
30	-1.11
35	1.67
40	4.44
45	7.22
50	10.00
55	12.78
60	15.56
65	18.33
70	21.11
75	23.89
80	26.67
85	29.44
90	32.22
95	35.00
100	37.78
105	40.56
110	43.33
115	46.11
120	48.89
125	51.67
130	54.44
135	57.22
140	60.00
145	62.78

150 65.56

The main program that you must use is,

```
public static void main(String[] args) {
    System.out.println("Fahrenheit      Celsius");

    for (int i = -50; i <= 150; i += 5) {
        System.out.printf("%6d      %10.2f\n", i, Celsius(i));
    }
}
```

You need to write the method `Celsius` that takes in a single parameter, a double data type, of the Fahrenheit temperature and return the corresponding Celsius temperature, also a double data type. The conversion formula for Fahrenheit to Celsius is

$$C = \frac{5}{9}(F - 32)$$

where C is the Celsius temperature and F is the Fahrenheit temperature.

- Most computer games use physics engines to take care of many aspects of game play. The most work that a physics engine usually does in a game is collision detection. Determining when the user (or object) hits a wall, when the laser blast or bullet in a FPS hits a target, etc. Collision detection is a fairly difficult problem to solve and is beyond the scope of this class, but another, more accessible, use of a physics engine in a game is to apply gravity to an object.

You have probably seen this in high school physics or possibly as an example in a mathematics class like Calculus. If not, we will discuss the equation fully. Say you take an object, like a tennis ball or small rock, and through it up into the air. The object will have a particular height off the ground at any time after you toss it until it eventually hits the ground. The object will increase in height, until gravity overcomes its initial velocity, and then it will start to fall to the ground. To find the height of the object at any time t in seconds after you toss the object is given by the equation,

$$d = -\frac{1}{2}gt^2 + v_0t + d_0$$

where d is the current height of the object, g is the acceleration of gravity, v_0 is the initial velocity and d_0 is the initial height. The acceleration of gravity on Earth in English units is $32.17405 \text{ feet/sec}^2$, so velocity is in feet/sec, distance is in feet and time is in seconds. Also note that we usually consider the positive direction as up. So a positive d_0 would indicate that the initial position of the object is above the ground and a positive v_0 would indicate that the initial velocity is in the upward direction. Along these lines, gravity would be pulling the object down and hence the negative sign.

A sample run of the program is below.

```
Input the Initial Height (in feet): 10
Input the Initial Velocity (in feet/sec.): 25
```

Time	Height
0.0	10.000
0.1	12.339
0.2	14.357
0.3	16.052
0.4	17.426
0.5	18.478
0.6	19.209
0.7	19.617
0.8	19.704
0.9	19.470
1.0	18.913
1.1	18.035
1.2	16.835

```

1.3          15.313
1.4          13.469
1.5          11.304
1.6          8.817
1.7          6.008
1.8          2.878
1.9          Hit the ground.

```

The main program that you must use is,

```

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("Input the Initial Height (in feet): ");
    Double d0 = keyboard.nextDouble();
    System.out.print("Input the Initial Velocity (in feet/sec.): ");
    Double v0 = keyboard.nextDouble();

    double d = d0;
    double t = 0;

    System.out.println();
    System.out.println(" Time Height");

    while (d > 0) {
        d = height(d0, v0, t);
        if (d > 0)
            System.out.printf("%5.1f %15.3f \n", t, d);
        else
            System.out.printf("%5.1f %20s \n", t, "Hit the ground.");
        t += 0.1;
    }
}

```

You need to write the method `height` that takes in three parameters all double values. The first is the initial height, the second the initial velocity, and the third is the time. The return value is the current height of the object, also a double data type.

3. Nice output for a blackjack program. The output of the program is to look like the following.

```

Player 1
Card 1: King of Spades
Card 2: 8 of Spades

Player 2
Card 1: King of Hearts
Card 2: Queen of Clubs

```

For this program the main must be the following.

```

public static void main(String[] args) {
    int p1f1 = (int) (Math.random() * 13) + 1;
    int p1s1 = (int) (Math.random() * 4) + 1;
    int p1f2 = (int) (Math.random() * 13) + 1;
    int p1s2 = (int) (Math.random() * 4) + 1;

    int p2f1 = (int) (Math.random() * 13) + 1;
    int p2s1 = (int) (Math.random() * 4) + 1;
    int p2f2 = (int) (Math.random() * 13) + 1;
    int p2s2 = (int) (Math.random() * 4) + 1;

    System.out.println("Player 1");
    System.out.println("Card 1: " + CardString(p1f1, p1s1));
    System.out.println("Card 2: " + CardString(p1f2, p1s2));
    System.out.println();
    System.out.println("Player 2");
    System.out.println("Card 1: " + CardString(p2f1, p2s1));
    System.out.println("Card 2: " + CardString(p2f2, p2s2));
}

```

You need to create the method `CardString` that takes in two integer parameters. The first parameter is the face value of the card (1–13 with 1 being the Ace and 11, 12, and 13 being Jack, Queen, and

King respectively). The second parameter is the suit value of the card (1–4 which represent Hearts, Diamonds, Clubs, and Spades respectively). The method is to return the string of the card's value and suit so that the output matches the examples.

4. High-low game: The output of the program is to look like the following. These are three separate runs.

Player 1: 5 of Spades	Player 1: Queen of Hearts	Player 1: 8 of Clubs
Player 2: Ace of Clubs	Player 2: Queen of Spades	Player 2: 8 of Clubs
Player 2 Wins	Player 2 Wins	It is a draw.

For this program the main must be the following.

```
public static void main(String[] args) {
    int player1CardFace = getCardFace();
    int player1CardSuit = getCardSuit();
    int player2CardFace = getCardFace();
    int player2CardSuit = getCardSuit();

    System.out.println("Player 1: " + CardString(player1CardFace, player1CardSuit));
    System.out.println("Player 2: " + CardString(player2CardFace, player2CardSuit));

    int win = Winner(player1CardFace, player1CardSuit, player2CardFace, player2CardSuit);

    if (win == 1)
        System.out.println("Player 1 Wins");
    else if (win == 2)
        System.out.println("Player 2 Wins");
    else
        System.out.println("It is a draw.");
}
```

Copy over your CardString method from the previous exercise, then update it so that the face values go from 2–14 with 11, 12, 13, and 14 being Jack, Queen, King, and Ace respectively. Create the following three methods,

- getCardFace() that returns a random number between 2 and 14.
- getCardSuit() that returns a random number between 1 and 4.
- Winner(int p1f, int p1s, int p2f, int p2s) that takes in four parameters, the first two are face value and suit of the card for player 1 and the last two are the face value and suit of the card for player 2. The method should return 1 if player 1 wins, 2 if player 2 wins and 0 if it is a draw. Remember that the rules are that the higher face value wins (with Ace high), if the face values are the same then the suits are compared with Spades winning over Clubs which wins over Diamonds which wins over Hearts.

5. Factorial: The output of the program is to look like the following. These are five separate runs.

n = 5 5! = 120	n = 20 20! = 2432902008176640000	n = -3 Invalid input!
n = 13 13! = 6227020800	n = 0 n! = 1	

For this program the main must be the following.

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("n = ");
    int n = keyboard.nextInt();

    if (n < 0)
        System.out.print("Invalid input!");
    else if (n == 0)
        System.out.print("n! = 1");
    else
        System.out.print(n + "! = " + factorial(n));
}
```

You need to create the method `factorial` that takes in a long integer parameter n and returns a long with the value of $n!$.

6. Double Factorial: The output of the program is to look like the following. These are five separate runs.

```
n = 5          n = 20          n = -3
5!! = 15       20!! = 3715891200 Invalid input!

n = 6          n = 0
6!! = 48       n!! = 1
```

For this program the main must be the following.

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("n = ");
    int n = keyboard.nextInt();

    if (n < 0)
        System.out.print("Invalid input!");
    else if (n == 0)
        System.out.print("0!! = 1");
    else
        System.out.print(n + "!! = " + doubleFactorial(n));
}
```

You need to create the method `doubleFactorial` that takes in a long integer parameter n and returns a long with the value of $n!!$.

7. Character Counts: The output of the program is to look like the following.

```
Input a phrase: This is a test.

The A count is 1
The E count is 1
The H count is 1
The I count is 2
The S count is 3
The T count is 3

The most frequent letters are S and T with a count of 3.
```

For this program the main must be the following.

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("Input a phrase: ");
    String str = keyboard.nextLine();
    str = str.toUpperCase();

    String MaxCharacters = "";
    int maxCount = 0;
    int numMax = 0;
    int maxCharsFound = 0;

    System.out.println();

    for (char ch = 'A'; ch <= 'Z'; ch++) {
        int count = countLetters(str, ch);
        if (count > 0)
            System.out.println("The " + ch + " count is " + count);

        if (count > maxCount) {
            maxCount = count;
        }
    }

    System.out.println();

    for (char ch = 'A'; ch <= 'Z'; ch++) {
        int count = countLetters(str, ch);
```

```
        if (count == maxCount)
            numMax++;
    }

    for (char ch = 'A'; ch <= 'Z'; ch++) {
        int count = countLetters(str, ch);
        if (count == maxCount) {
            MaxCharacters += ch;
            maxCharsFound++;
            if (numMax > 1) {
                if (maxCharsFound < numMax - 1)
                    MaxCharacters += ", ";
                else if (maxCharsFound == numMax - 1)
                    MaxCharacters += " and ";
            }
        }
    }

    if (maxCount > 0) {
        if (numMax == 1)
            System.out.println("The most frequent letter is " + MaxCharacters + " with a count of " +
                               maxCount + ".");
        else
            System.out.println("The most frequent letters are " + MaxCharacters + " with a count of " +
                               maxCount + ".");
    }
}
```

You need to create the method `countLetters` that takes in a string and a character and returns the number of times that character is in the string.