

1 Exercises

For the following exercises, pair up with someone else in the class and create a project for each exercise. In your comment section at the top of each of the programs make sure that both of your names are listed in the Author line in the comments at the top of each program.

For each of the following create a new project with an appropriate name and then write a program that solves the given problem. Remember to do Shift+Ctrl+F to format the program, or Shift+Command+F on the Mac to format the code. Also remember the standard comments of at the top, Authors, Date, and Description.

As usual, you will be submitting through MyClasses the java code files and a Microsoft Word docx file, LibreOffice Writer odt, or a text file (which you can create with NotePad++) which contains the output of at least three runs of each program on different data inputs. Remember that you can copy and paste output from the Eclipse console area to the word or text program. Only one of you will need to submit the files.

-
1. Create a program that will let the user input a non-negative number n and return $n!$ (n factorial). The factorial of a positive integer n is the product of all positive integers from 1 to n . Also, we define $0! = 1$.

So if $n \geq 1$ we define $n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdots (n-1) \cdot n$ and $0! = 1$.

For example,

$$5! = 120, \quad 7! = 5040, \quad 10! = 3628800, \quad 15! = 1307674368000$$

Have the program check the value of n before calculating the factorial and if the user inputs a value less than 0 the program should print an error. Also you will write this using all three types of loops, while, do-while and for. You may put all three into the same program. In other words, do the calculation with a while loop and print out the result, then reset all your variables and use a do-while loop to calculate and then print its result, finally reset all your variables again and use a for loop. An example run is below.

```
Input n: 6
While Loop      6! = 720
Do-While Loop   6! = 720
For Loop        6! = 720
```

Solution:

```
1 import java.util.Scanner;
2
3 public class Factorial {
4
5     public static void main(String[] args) {
6         Scanner keyboard = new Scanner(System.in);
```

```

7      System.out.print("Input n: ");
8      int n = keyboard.nextInt();
9      long fact = 1;
10     int saven = n;
11
12     if (n < 0) {
13         System.out.print("Error: n must be greater than or equal to 0.");
14     } else {
15         fact = 1;
16         while (n > 0) {
17             fact = fact * n;
18             n--;
19         }
20         System.out.println("While Loop      " + saven + "! = " + fact);
21
22         fact = 1;
23         n = saven;
24         if (n > 0) {
25             do {
26                 fact = fact * n;
27                 n--;
28             } while (n > 0);
29         }
30
31         System.out.println("Do-While Loop  " + saven + "! = " + fact);
32
33         fact = 1;
34         n = saven;
35         for (int i = 1; i <= n; i++) {
36             fact = fact * i;
37         }
38
39         System.out.println("For Loop      " + saven + "! = " + fact);
40     }
41 }
42 }

```

2. Create a program that will let the user input a non-negative number n and return $n!!$ (n double factorial). The double factorial of a positive integer n is the product of every other positive integer from n down to 1. Also, we define $0!! = 1$.

So if $n \geq 1$ we define $n!! = n \cdot (n - 2) \cdot (n - 4) \cdots 1$ and $0! = 1$.

For example,

$$5!! = 15, \quad 7!! = 105, \quad 10!! = 3840, \quad 15!! = 2027025$$

Have the program check the value of n before calculating the double factorial and if the user inputs a value less than 0 the program should print an error. Also you will write this using all three types of loops, while, do-while and for. You may put all three into the same program. In other words, do the calculation with a while loop and print out the result, then reset all your variables and use a do-while loop to calculate and then print its result, finally reset all your variables again and use a for loop. An example run is below.

```

Input n: 25
While Loop      25!! = 7905853580625
Do-While Loop   25!! = 7905853580625
For Loop        25!! = 7905853580625

```

Solution:

```

1  import java.util.Scanner;
2
3  public class DoubleFactorial {
4
5      public static void main(String[] args) {
6          Scanner keyboard = new Scanner(System.in);
7          System.out.print("Input n: ");
8          int n = keyboard.nextInt();
9          long fact = 1;
10         int saven = n;
11
12         if (n < 0) {
13             System.out.print("Error: n must be greater than or equal to 0.");
14         } else {
15             fact = 1;
16             while (n > 0) {
17                 fact = fact * n;
18                 n -= 2;
19             }
20             System.out.println("While Loop      " + saven + "!! = " + fact);
21
22             fact = 1;
23             n = saven;
24             if (n > 0) {
25                 do {
26                     fact = fact * n;
27                     n -= 2;
28                 } while (n > 0);
29             }
30
31             System.out.println("Do-While Loop  " + saven + "!! = " + fact);
32
33             fact = 1;
34             n = saven;
35             for (int i = n; i >= 1; i -= 2) {
36                 fact = fact * i;
37             }
38
39             System.out.println("For Loop       " + saven + "!! = " + fact);
40         }
41     }
42 }

```

2 Challenge Exercise

Challenge Exercises are optional, they will be graded as extra credit.

If you try larger numbers in your factorial program you will see that eventually you will get the wrong answer due to integer overflow. In fact if you put in even larger numbers you always get 0 because of incorporating a sufficient power of two in the calculation.

Java has a built-in class called `BigInteger` that allows what is referred to as infinite precision, or really a better name of arbitrary precision. This `BigInteger` number will get as large as needed. So, for example, using it in place of ints and longs we can calculate 100!,

```

Input n: 100
100! = 93326215443944152681699238856266700490715968264381621468592963895217599
993229915608941463976156518286253697920827223758251185210916864000000000000000

```

000000000

or 1000! or even 10000!. This feature comes with a price, the `BigInteger` is not like the `int` or `long` in that it does not recognize arithmetic operations, so if a and b are two `BigInteger`s the code $a + b$ is a syntax error. Also, it does not recognize logical operations, so if a and b are two `BigInteger`s the code $a > b$ is a syntax error as well. There are alternatives to these operations, in addition we need to use an alternative to create a `BigInteger`, `BigInteger a = 4;` will not work. Here is a brief overview of what you will need about the `BigInteger`.

First you will need to import the `BigInteger` functionality into your program, just like we do with the `Scanner`, so put

```
import java.math.BigInteger;
```

at the top of your program. Now take a look at the following segment of code and its output.

```
BigInteger a = BigInteger.valueOf(7);
BigInteger b = BigInteger.valueOf(4);
BigInteger c;

c = a.add(b);
System.out.println(c);
c = a.subtract(b);
System.out.println(c);
c = a.multiply(b);
System.out.println(c);
c = a.divide(b);
System.out.println(c);
c = a.mod(b);
System.out.println(c);
int d = a.compareTo(b);
System.out.println(d);
d = b.compareTo(a);
System.out.println(d);
```

Output:

```
11
3
28
1
3
1
-1
```

At the top is one method for creating a `BigInteger`, the line,
`BigInteger a = BigInteger.valueOf(7);`

will create the `BigInteger` a and set it to the value of 7. Note that the value you set a to (i.e. 7 in this case) must be an `int` or `long` and it can be either a literal or variable. The arithmetic

operations are done using “functions” or “methods” just like the Math object, compare each line with the output. Also notice what divide and mod do. For logical comparison there is a compareTo function, just like with Strings, notice the output of these as well.

Take the factorial program you wrote and convert it to using BigIntegers, you need to use only one loop structure and of course put this into a separate project. Think about how you can use the arithmetic functions in place of the operators and how you can use the compareTo function in an if or while condition.

Solution:

```
1 import java.math.BigInteger;
2 import java.util.Scanner;
3
4 public class BigFactorial {
5
6     public static void main(String[] args) {
7         Scanner keyboard = new Scanner(System.in);
8         System.out.print("Input n: ");
9         long longn = keyboard.nextLong();
10        BigInteger n = BigInteger.valueOf(longn);
11
12        if (n.compareTo(BigInteger.ZERO) < 0) {
13            System.out.print("Error: n must be greater than or equal to 0.");
14        } else {
15            BigInteger fact = new BigInteger("1");
16            while (n.compareTo(BigInteger.ZERO) > 0) {
17                fact = fact.multiply(n);
18                n = n.subtract(BigInteger.ONE);
19            }
20            System.out.println("n! = " + fact);
21        }
22    }
23 }
```