# Exercise 5
# Chapter 5 loops

COMP217
Java Programming
Spring 2019

Instructor: Gil-Jin Jang

Do it yourself

5-1 Submission: Ex51a.java  Ex51b.java

# Ex5-1 While to do-While

1. (**Ex51a.java**) Write a full Java code for the following loop:

```
int n = 10;
while (n > 0) {
      System.out.println(n);
      n = n - 3;
}
```

2. (**Ex51b.java**) Change the above code using a do-while loop

3. Submission: 2 Java source files

```
Ex51a.java
Ex51b.java
```

Practice

# Ex 5-2 BreakTest.java

Write and submit **BreakTest.java**

```
import java.util.Scanner;

public class BreakTest {
  public static void main(String[] args) {
    int total = 0, count = 0;   // initialize when declared
    Scanner sc = new Scanner(System.in);

    // This example shows exiting a loop not by counting
    while ( true ) {
      int score;         // can declare inside
      System.out.print("Your score? (negative number when done) ");
      score = sc.nextInt();
      if ( score < 0 )
        break;  // (**) get out of the loop
      total += score;
      count++;
    }

    // (**) break jumps here
    // variable count is to compute average
    System.out.printf("Average score is %.2f\n",
        (double)total/(double)count);
  }
}
```

```
/*
mico:week5$ javac BreakTest.java
mico:week5$ java BreakTest
Your score? (negative number when done) 3
Your score? (negative number when done) 4
Your score? (negative number when done) 5
Your score? (negative number when done) 0
Your score? (negative number when done) -1
Average score is 3.00
*/
```

# Ex 5-3 Factorial10.java

- The java code "`Factorial.java`" in the lecture slide can compute up to 21!, due to the range of long type (64 bits), whose largest value is 2^63-1 = 9223372036854775807
  - `20!     =     2432902008176640000`
  - `2^63-1 =     9223372036854775807`
  - `21!     =    51090942171709440000`
- Note that there are four zeros in 20! and 21! from the smallest digit. One of the simple ideas to store larger factorial is, divide the current factorial by 10 as many as trailing 0's, and use the dividends in factorial calculation.
  - *Hints*:
    - `21!     =     510909421717094400 * 10000`
    - `22!     =   1124000727777607680 * 10000`
    - `23!     = 25852016738884976640 * 10000`
    - `2^63-1 = 9223372036854775807`
  - In this way, we can compute up to **23!**

Do it yourself

# Ex 5-3 Factorial10.java, continued

- Write and submit a java code "`Factorial10.java`" which can compute up to 23! EXACTLY, by representing the factorial results by _____ * 1**0…0**
  - You will need an integer, **`num10`**, that stores the number of 0 from the rightmost digit
    - For example, for 3628800: `fac = 36288; num10 = 2;`
    - After every computation of factorial, check if it is multiples of 10, and adjust the variables fac and num10:
      - while ( (fac % 10) == 0 ) { num10++; fac/=10; }  // why not 'if' instead of 'while'?
  - Add a few lines to "`Factorial.java`" to complete "`Factorial10.java`"
- Example output:

```
$ java Factorial10
n? 22
22! = 1124000727777607680000 (fac = 1124000727777607768, num10 = 4)
$ java Factorial10
n? 23
23! = 25852016738884976640000 (fac = 2585201673888497664, num10 = 4)
$ java Factorial10
n? 24
Overflowed at 24!
23! = 25852016738884976640000 (fac = 2585201673888497664, num10 = 4)
```

# Least Common Multiple (LCM)

최소공배수

- Least common multiple (LCM) of two integers, $x$ and $y$, is the smallest integer that can be divided by both $x$ and $y$.
- One of the smart ways of finding LCM is using GCD (greatest common divisor)
  - Let g = GCD(x,y); then we can express x = g*u, y = g*v, where u and v are integers as well.
  - From the definition of GCD, GCD(u,v) = 1 ➔ LCM(u,v) = u*v
  - LCM(x,y) = LCM(g*u, g*v) = g*LCM(u,v) = g*u*v = (g*u*g*v) / g = **x*y/GCD(x,y)**
- So LCM of x and y can be found as follows:
  - Using Euclid GCD algorithm (in the lecture slide) to find GCD(x,y)
  - LCM(x,y) = x*y/GCD(x,y)

Do it yourself

# Ex 5-4 LCM of 10-19 and 20-29

- For every pair of x in [10…19] and y in [20…29], find LCM(x,y), and represent them in the following matrix form:

```
$ java LCM
    |   20   21   22   23   24   25   26   27   28   29
----+----------------------------------------------------
 10 |   20  210  110  230  120   50  130  270  140  290
 11 |  220  231   22  253  264  275  286  297  308  319
 12 |   60   84  132  276   24  300  156  108   84  348
 13 |  260  273  286  299  312  325   26  351  364  377
 14 |  140   42  154  322  168  350  182  378   28  406
 15 |   60  105  330  345  120   75  390  135  420  435
 16 |   80  336  176  368   48  400  208  432  112  464
 17 |  340  357  374  391  408  425  442  459  476  493
 18 |  180  126  198  414   72  450  234   54  252  522
 19 |  380  399  418  437  456  475  494  513  532  551
```

  - All the values should be right-aligned in 5 spaces --- use “%5d”
- Submission: **LCM.java**

Do it yourself

# Ex 5-5 TwoTriangles.java

- Write a java code “TwoTriangles.java” by modifying “NestedLoop2.java” to display the following shapes:

```
$ java TwoTriangles
How many lines? 5
*
**
***
 **
  *
$ java TwoTriangles
How many lines? 6
*
**
***
***
 **
  *
```

```
$ java TwoTriangles
How many lines? 7
*
**
***
****
 ***
  **
   *
$ java TwoTriangles
How many lines? 8
*
**
***
****
****
 ***
  **
   *
```

```
$ java TwoTriangles
How many lines? 9
*
**
***
****
*****
 ****
  ***
   **
    *
$ java TwoTriangles
How many lines? 10
*
**
***
****
*****
*****
 ****
  ***
   **
    *
```

Do it yourself

# Ex 5-6 DiamondStar.java

- Write a java code “DiamondStar.java” to display the following diamond-shaped asterisks:

```
$ java DiamondStar
How many lines? 5
  *
 ***
*****
 ***
  *
$ java DiamondStar
How many lines? 6
  *
 ***
*****
*****
 ***
  *
```

```
$ java DiamondStar
How many lines? 7
   *
  ***
 *****
*******
 *****
  ***
   *
$ java DiamondStar
How many lines? 8
   *
  ***
 *****
*******
*******
 *****
  ***
   *
```

```
$ java DiamondStar
How many lines? 9
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
$ java DiamondStar
How many lines? 10
    *
   ***
  *****
 *******
*********
*********
 *******
  *****
   ***
    *
```

# References
## Useful slides for exercise 5 (NO NEED TO SUBMIT)

Factorial.java

Gcd.java

NestedLoop2.java for triangle display

Practice

# Factorial.java

```java
/* Class Factorial computues
 n! = n(n-1)(n-2)...1 */

import java.util.Scanner;

public class Factorial {
  public static void main(String[] args) {
    long fac;   // long: factorial is very large
    long pre_fac;       // to check overflow
    int i, n;
    Scanner sc = new Scanner(System.in);

    System.out.print("n? ");
    n = sc.nextInt();

    // start from fac = 0! = 1
    for (i=1, fac=1L; i<=n; i++) {
      pre_fac = fac;
      fac *= i;

      // check if overflowed
      if ( pre_fac != fac / i ) {
        System.out.println("Overflowed at " + i + "! = " + fac);
        fac = pre_fac;  // roll back to the previous, unoverflowed
        break;
      }
    }

    // [Q] Why (i-1)?
    System.out.println((i-1) + "! = " + fac);
  }
}
```

```
/*
mico:week5$ javac Factorial.java
mico:week5$ java Factorial
n? 10
10! = 3628800
mico:week5$ java Factorial
n? 20
20! = 2432902008176640000
mico:week5$ java Factorial
n? 30
Overflowed at 21! = -4249290049419214848
20! = 2432902008176640000
*/
```

Practice

# GCD (greatest common divisor)

- Euclid algorithm
  - Input: two integers x and y
  1) For x ≥ y
  2) if x *mod* y is zero, y is the gcd
  3) otherwise, the gcd of x and y is the gcd of y and x *mod* y

```
/*
$ javac Gcd.java
$ java Gcd
Enter two integers: 240 36
The greatest common divisor is 12
*/
```

```
import java.util.Scanner;

public class Gcd {
  public static void main(String[] args) {
    int x, y, r;
    Scanner scan = new Scanner(System.in);
    System.out.print("Enter two integers: ");
    x = scan.nextInt();
    y = scan.nextInt();

    if ( x < y ) {
      /* swap x and y to satisfy x>=y */
      r = x; x = y; y = r;
    }
    while (y != 0) {
      r = x % y;
      x = y;
      y = r;
    }
    System.out.println(
        "The greatest common divisor is "
        + x);
  }
}
```

Practice

# NestedLoop2.java

```java
import java.util.Scanner;

public class NestedLoop2 {
  public static void main(String[] args) {
    int n;
    Scanner sc = new Scanner(System.in);

    System.out.print("How many lines? ");
    n = sc.nextInt();

    // can delare variable inside for loop
    for (int y=1; y<=n; y++) {
      for (int x=1; x<=y; x++) {
        System.out.print("*");
      }
      System.out.println("");   // change line
    }
  }
}
```

```
/*
mico:week5$ javac NestedLoop2.java
mico:week5$ java NestedLoop2
How many lines? 5
*
**
***
****
*****
mico:week5$ java NestedLoop2
How many lines? 10
*
**
***
****
*****
******
*******
********
*********
**********
*/
```