Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Computer Science Department

Tutorials

CS140 – Computer Programming 1

(last updated: January 2019)

Table of Contents

Note to students

This tutorial exercises are prepared for tutorial sessions of the course CS140 – Computer Programming 1.

To take benefit from tutorial sessions, students must work continuously and must feel committed to the course during the whole semester. To do so, students have to:

- Before the tutorial: Review the course, read, understand, and, eventually, start answering the tutorial exercises.
- During the tutorial:
    - Pay attention to the instructor explanation and try learning how to deal with the problems and how to resolve them gradually
    - Take notes about the solution done during the tutorial session because sometimes the provided solution may be different from the shared one.
    - Participate to the tutorial session by asking pertinent questions and answering the instructor questions
    - Compare your eventual solution with the one provided by the instructor
- After the tutorial: try to resolve again the exercises of the tutorial especially those that you haven't correctly resolved, or you haven't resolved at all before the tutorial and compare again your solution to the one proposed by the instructor.

Each tutorial in this document is intended to be done in one tutorial session (2 hours) unless if it is indicated otherwise in the detailed course schedule. After the end of each tutorial session, the course instructor provides a soft copy of the detailed solution of the exercises to students.

We are happy to have you among our students and we wish you a successful course with an excellent level of learning and practicing.

Tutorial 01: Fundamentals of Java Programs, Input/Output, Variables, and Arithmetic

Exercise 1 (textbook)

Write an application that displays a checkerboard pattern, as follows:

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

Exercise 2

What is the output of the following program?

```
public class Exercise02 {
    public static void main(String[] args) {
        System.out.print("Learn\tJava\n\tthe\nHard\tWay\n\n");
        System.out.print("\tLearn Java the \"Hard\" Way!\n");

        System.out.print("Hello\n");
        System.out.print("Jellow\rY\n");

        System.out.println("\\ // -=- \\ //");
        System.out.println("\\\\ \\\\\\ \\\\\\\\");
    }
}
```

Exercise 3

What is the output of each of the following instructions?

```
1) System.out.printf("'%5d'", 10);
2) System.out.printf("'%-5d'", 10);
3) System.out.printf("'%05d'", 10);
4) System.out.printf("'%+5d'", 10);
5) System.out.printf("'%-+5d'", 10);
6) System.out.printf("'%.1f'", 10.3456);
```

```
7) System.out.printf("'%.2f'", 10.3456);
8) System.out.printf("'%8.2f'", 10.3456);
9) System.out.printf("'%8.4f'", 10.3456);
10) System.out.printf("'%08.2f'", 10.3456);
11) System.out.printf("'%-8.2f'", 10.3456);
12) System.out.printf("'%s'", "Hello");
13) System.out.printf("'%10s'", "Hello");
14) System.out.printf("'%-10s'", "Hello");
```

Exercise 4 (textbook)

Assuming that x = 2 and y = 3, what does each of the following Java statements display?

1) System.out.printf("x =");
2) System.out.printf("x = %d\n", x);
3) System.out.printf("Value of %d + %d is %d\n", x, x, (x + x));
4) System.out.printf("%d = %d\n", (x + y), (y + x));

Exercise 5

Write a Java program that computes the value of `y` where $y=x^3-x+1$ and `x` is given by the user.

Exercise 6 (textbook)

State the order of evaluation of the operators in each of the following Java statements, and show the value of x after each statement is performed:

1) x = 7 + 3 * 6 / 2 - 1;
2) x = 2 % 2 + 2 * 2 - 2 / 2;
3) x = (3 * 9 * (3 + (9 * 3 / (3))));

Exercise 7 (textbook)

So far, you learned about integers and the type `int`. Java can also represent floating-point numbers that contain decimal points, such as 3.14159. Write a Java application that inputs from the user the radius of a circle as an integer and prints the circle's diameter, circumference and area using the floating-point value 3.14159 for π.

Use the following formulas (r is the radius):

diameter = 2r

circumference = $2\pi r$
area = πr^2

Do not store the results of each calculation in a variable. Rather, specify each calculation as the value that will be output in a `System.out.printf` statement. The values produced by the circumference and area calculations are floating-point numbers. Such values can be output with the format specifier `%f` in a `System.out.printf` statement. You'll learn more about floating-point numbers in the next studied topics.

Tutorial 02: Introduction to Problem Solving, Selection Control Statements & Logical Operators

Exercise 1

Write a Java program that computes the maximum of 3 numbers entered by the user.

Exercise 2 (textbook)

Write a Java application that reads two integers, determines whether the first is a multiple of the second and prints the result.

Problem (past exam)

(`class ConvertHours`) Write a Java application that takes a positive integer representing a duration in hours as input, and outputs the same duration decomposed into weeks, days, and hours as detailed below.

unit	suffix used in output	conversion
week	wk	1 week = 7 days
day	d	1 day = 24 hours
hour	hr	

You must consider the following:

- Accept only strictly positive value of hour and less than or equal to 8760
- Only include quantities with non-zero values in the output (e.g., print "1 d" and not "0 wk, 1 d, 0 hr ").
- Give larger units precedence over smaller ones as much as possible (e.g., print 2 d, 10 hr and not 1 d, 34 hr or 58 hr).

Tutorial 03: Introduction to Problem Solving, Selection Control Statements & Logical Operators

Exercise 1 (textbook)

Write a Java program that inputs five real numbers and determines and prints the number of negative numbers input, the number of positive numbers input and the number of zeros input.

Exercise 2

Write a Java program that determines whether a number **n** given by the user is inside the range [**a**, **b**] or not, where **a** and **b** are also given by the user in any order (your program should ensure that a always smaller than b). Hint: Swap their values when needed.

Exercise 3

Write a Java program that determines if a number **n** given by the user is multiple of 11 and smaller than 100. Handle all (4) possible cases.

Exercise 4

Write a Java program to simulate a simple calculator. It should ask the user to enter a first number, the operation, and the second number. Addition ("+"), Subtraction ("-"), Division ("/"), Multiplication ("*"), and Modulus ("%") are the basic operations that should be implemented. Use the switch statement for the selection and the type `char` to represent the desired operation sign.

When the operation sign is not recognized, your program should print an error message.

Hints: (1) Declare a variable of type `char` for the operator, (2) Read the operation sign using `input.next().charAt(0)` where input is a variable of type `Scanner`, and (3) use operation signs (as char) in the switch's cases (`case '+' :`).

Problem 1 (past exam)

(class CheckOrder) Write a Java program that performs the following tasks:

- Ask the user to input 4 real numbers.
- Determine if the numbers are in ascending order, descending order, or not ordered at all.
- In all cases, compute and display the average of the four numbers.

Example 1:

```
Enter four numbers: 1.0 2.0 3.0 4.0

The numbers are in ascending order.
Their average is: 2.500000.
```

Example 2:

```
Enter four numbers: 9.9 7.7 5.5 3.3

The numbers are in descending order.
Their average is: 6.600000.
```

Example 3:

```
Enter four numbers: 20.0 10.0 40.0 30.0

The numbers are not ordered.
Their average is: 25.000000.
```

Problem 2 (past exam)

(class BankAccount) Write a Java program that simulates primitive bank operations. The program will perform the following tasks:

- Declare the amount variable and initialize it with the value 5000.
- Print a menu of possible operations: 1) Amount, 2) Deposit, and 3) Withdraw.
- Ask the depositor to choose an operation:
    - If the operation is 1, print the amount.
    - If the operation is 2, ask the user to enter the amount to deposit and add it to the old amount and print the result.
    - If the operation is 3, ask the user to enter the amount to withdraw and add subtract it from the old amount (only if it is possible) and print the result.

Example 1:

```
Choose an operation:
      1) Amount
      2) Deposit
      3) Withdraw
Please choose 1, 2 or 3:  1

The amount is 5000.000000.
```

Example 2:

```
Choose an operation:
      1) Amount
      2) Deposit
      3) Withdraw
Please choose 1, 2 or 3:  2
Enter the amount to deposit: 300

The new amount is 5300.000000.
```

Example 3:

```
Choose an operation:
      1) Amount
      2) Deposit
      3) Withdraw
Please choose 1, 2 or 3:  3
Enter the amount to withdraw: 3202

The new amount is 1798.000000.
```

Tutorial 04: Formulating Algorithms, Repetition Control Statements

Exercise 1 (textbook)

Write a Java program that computes $\mathbf{n^p}$ where **n** and **p** are two numbers given by the user.

Exercise 2

Write a Java program that computes the product of odd numbers that are in the range]a, b[where a and b are given by the user (without nesting the control statements).

Exercise 3

Phase 1:

Write a Java program called **`CircleComputation`**, which prompts user for a radius (in double) and compute the area (πr^2) and circumference ($2\pi r$) of the circle rounded to 2 decimal places

Phase 2:

Modify the above exercise. The program shall repeatedly prompt for the radius, until the user enters -1.

Problem 1 (past exam)

(class CheckPower) Write a Java program that asks the user to enter 2 numbers and checks:

- If the first number is multiple of the second or not
- If the first number is a perfect square of the second or not (e.g.: 16 is a perfect square of 4 since: $16 = 4^2$)
- If the first number is power of the first (e.g.: 128 is power of 2 since: $128 = 2^7$)

Example 1:

```
First number = 9
Second number = 80

80 is not multiple of 9
```

Example 2:

```
First number = 128
Second number = 2

128 is multiple of 2
128 is not a perfect square of 2
128 is power of 2, 128 = 2 ^ 7
```

Example 3:

```
First number = 25
Second number = 5

25 is multiple of 5
25 is a perfect square of 5
25 is power of 5, 25 = 5 ^ 2
```

Problem 2 (past exam)

(class `GradeComputations`)

Write a Java program that asks the teacher (user) to enter a set of grades (0-100) in ascending order or to enter 1 to exit.

For each grade entered by the teacher, your program should check if it is greater than the previous entered grade, if yes, it will consider the grade to compute the sum of even grades and the product of odd grades. Print error message when the entered grade is not in a good order and don't consider it for computation.

Example:

```
First grade: 65
Next number: 76
Next number: 70
70 is smaller than 76. Try again.
Next number: 85
Next number: 92
Next number: -1

Sum evens = 168
Prod odds = 5525
```

Tutorial 05: Formulating Algorithms, Repetition Control Statements

Exercise 1

Write a Java program that computes the minimum of n numbers given by the user (use counter and sentinel (value: 99999) approaches).

Exercise 2

Write a Java program that computes the least common multiplier (LCM) of two numbers **n** and **p** given by the user.

Exercise 3

What is the output of the following codes?

```
int n = 4;
int i = 1;
int c = 0;

while (i < n){
      if (i % 2 == 0) {
            i+=2;
      }
      else {
            --i;
      }
      c++;
}
System.out.println( "c = " + c );
System.out.println( "i = " + i );
```

Exercise 4

Write a Java program that calculate the number of consonants (one counter for all consonants) and the number of each vowel (a counter by vowel) in a sequence of characters entered by the user. When the user wants to stop entering characters he will enter the character '#' (use switch).

Problem 1 (past exam)

(class `CheckOrder`)

Write a Java program that checks if a set of 10 numbers entered by the user are in the ascending order (in this case, print congrats message) or not (in this case, print the number of conflicts).

Example 1:

```
Number 1 = 3
Number 2 = 14
Number 3 = 25
Number 4 = 33
Number 5 = 67
Number 6 = 100
Number 7 = 102
Number 8 = 190
Number 9 = 230
Number 10 = 300

Congrats. All numbers are ordered.
```

Example 2:

```
Number 1 = 4
Number 2 = 6
Number 3 = 2
Conflict: 6.0 > 2.0
Number 4 = 20
Number 5 = 20
Number 6 = 30
Number 7 = 29
Conflict: 30.0 > 29.0
Number 8 = 40
Number 9 = 50
Number 10 = 60

Unfortunately, you have 2 conflicts.
```

Problem 2 (past exam)

(class `HailstoneSeries`)

Write a Java program that asks the user for a positive integer N and then calculates a new value for N based on whether it is even or odd:

- if N is even, the new N is N/2. (Use integer division.)
- if N is odd, the new N is 3*N + 1.

Repeat with each new N until you reach the value 1. An error message is displayed if the user enters a negative value of N.

In addition to calculating and printing out the sequence for a particular initial N, keep track of the length of the sequence and the maximum integer in the sequence and print them out when the sequence reaches its end.

Example 1:

```
N = 12

12 even
6 even
3 odd
10 even
5 odd
16 even
8 even
4 even
2 even

Max = 16
Length = 9
```

Example 2:

```
N = -15

N must be positive number.
```

Tutorial 06: Methods, Introduction to Recursion

Exercise 1

Write a Java method f that computes f(x) where $f(x) = x^3 - x + 1$. (2 solutions are expected: with and without calling pow() from Math class)

Exercise 2

Write a function **g** that computes **g(x, y)** where $\mathbf{g(x, y) = x^2 + \sqrt{2y} + 1}$.

Exercise 3

What will be the output when calling the following function?

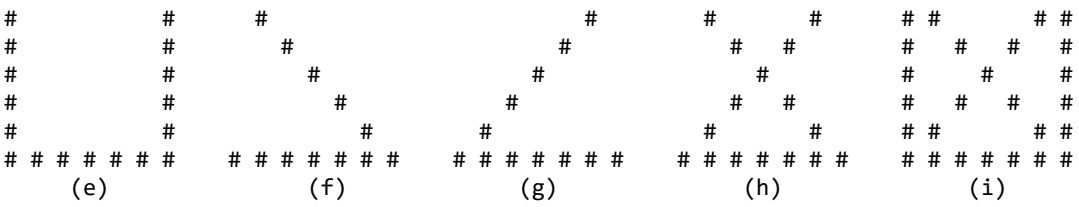
```
public static void mat()
{
  for (int count1=1; count1 <= 5; count1++)
  {
     for (int count2=1; count2 <= 5; count2++)
        System.out.print (count1*count2 + " ");
     System.out.println();
  }
}
```

Exercise 4

Write Java functions to print each of the followings patterns using nested loops. The size of the shape is asked from the user using the main function then it is passed as parameter to the concerned function.

```
#                 # # # # # # # #     # # # # # # # #                 #
# #               # # # # # # # #       # # # # # # #               # #
# # #             # # # # # # #           # # # # # #             # # #
# # # #           # # # # # #               # # # # #           # # # #
# # # # #         # # # # #                   # # # #         # # # # #
# # # # # #       # # # #                       # # #       # # # # # #
# # # # # # #     # # #                           # #     # # # # # # #
# # # # # # # #   # #                               #   # # # # # # # #
    (a)               (b)               (c)               (d)

# # # # # # #   # # # # # # #   # # # # # # #   # # # # # # #   # # # # # # #
```


(e)
(f)
(g)
(h)
(i)

Tutorial 07: Methods, Introduction to Recursion

Exercise 1

Write a Java method random that generates and returns a random number between 14 and 59 (using SecureRandom). Test your method several times using a main function.

Exercise 2 (BankAccount)

Write a Java program that simulates primitive bank operations. To achieve this goal, follow these steps:

1. Declare a static field `amount` with default value `5000.00`.
2. Declare another static field of type `java.util.Scanner`.
3. Define a method `displayMenu()` that prints the available operations (Amount, Deposit, Withdraw, and Exit).
4. Define a method `getChoice()` that calls `displayMenu()`, prompt the user to enter his/her choice, reads and returns the choice (1, 2, 3, or 4).
5. Define a method `displayAmount()` that display the current `amount` value.
6. Define a method `depositAmount()` that takes an amount as parameter, adds it to the current `amount`, and displays the new value of the `amount`.
7. Define a method `withdrawAmount()` that takes an amount as parameter, removes it from the current `amount`, and displays the new value of the `amount`.
8. Write a `main()` method that calls `getChoice()` repeatedly until the user enters the choice 4 (exit), performs (calls) the right operation (method) according to the user choice.

The `main()` content could be:

```
int operation;
while ( (operation = getChoice()) != 4 )
{
   switch (operation)
   {
      case 1 :
         displayAmount();
         break;
      case 2:
         System.out.print("Enter the amount to deposit: ");
         depositAmount(input.nextDouble());
         break;
      case 3:
         System.out.print("Enter the amount to withdraw: ");
         withdrawAmount(input.nextDouble());
         break;
      default:
         System.out.println("Unknown operation.");
   }
}
```

Exercise 3 (class Calculation)

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. Argument lists could differ in:

1. Number of parameters.
2. Data type of parameters.
3. Sequence of Data type of parameters.

Write different version of method **`sum()`** that display the sum of the values received as parameter according to the following main method content:

```
public static void main(String[] args)
{
   sum ( 10, 10 );

   sum ( 10, 10, 10 );

   sum ( 10.0, 10.0 );

   sum ( 10, 10.0 );

   sum ( 10.0, 10);

}
```

You have to define five functions with the specified types. Then, demonstrate the Argument Promotion concept by reducing the number of method to two.

Exercise 4 (class Ackermann)

The Ackermann recursive function is defined as follows:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Its arguments are never negative and it always terminates. Write a Java method which returns the value of ***A(m,n)***.
Write the following code fragment inside the main method to test Ackermann function and report what will happen.

```
for (m = 0; m <= 4; m++)
    for (n = 0; n < 6 - m; n++)
      System.out.printf("A(%d, %d) = %d\n", m, n, ackermann(m, n));
```

Problem

(class `RandomBarcode`)

Write a Java application that generates random barcodes where their number is given by the user. For simplicity, the barcode is composed of 3 parts:

- Part 1: a country code (we consider only 3 codes: 627 (Kuwait), 628 (Saudi Arabia), and 629 (Emirates)).
- Part 2: a product code between 10000 and 99999.
- Part 3: a control code between 0 and 9.

First, define **`displayBarcode`** method that generates and display (3 numbers composing) a barcode (using `SecureRandom` class). The function returns to the caller only the value of the generated country code number.

Then, write a main method that gets the number of the barcodes from the user, generates and prints the desired number of addresses by calling **`displayBarcode`** method, and, for every generated barcode, prints the country name.

Run

```
How many barcodes? 7

629 16035 1 (Emirates)

627 29043 4 (Kuwait)

628 92851 5 (Saudi Arabia)

627 66560 3 (Kuwait)

628 35648 8 (Saudi Arabia)

629 75400 6 (Emirates)

627 27921 8 (Kuwait)
```

Tutorial 08: Arrays

Exercise 1

Write and test a Java method minimum() that returns the minimum of an array of n integers.

Exercise 2

Write and test a Java method mostFrequent() that returns the most frequent element in an array of n integers.

Exercise 3

Write and test a Java method clshift() that performs a circular left shift on an array of n integers.

Exercise 4

Write and test a Java method scalarProduct() that receives 2 integer arrays of the same size and returns the scalar product of the 2 arrays.

Problem 1 (class InsertElement)

First, write a Java method **`insert`** that takes 3 parameters: 1) a non-empty integer array, an integer value to be inserted into the array, and an integer position where the new value to be inserted in the array. The method inserts the new value at the given position, pushes to the right all the elements from this position to the end of array, and returns the overflowed element value. Suppose that the user will enter a correct position (between 0 and array's length).

Second, write two java methods **`readArray(int[]a)`** and **`printArray(int[]a)`** that reads/writes the elements of an array passed in parameter from/to the standard input/output.

Finally, use the following **`main`** method to test your different methods:

```
public class InsertElement
{
   private static Scanner input = new Scanner(System.in);

   public static void main(String[] args)
   {
      int[] array = new int[10];
      System.out.print("Enter 10 elements: ");
      readArray(array);

      System.out.print("New element and its position: ");
      int number = input.nextInt();
      int position = input.nextInt();

      int oe = insert(array, number, position);
      System.out.println("Overflowed element: " + oe);

      System.out.print("Array after insert: ");
      printArray(array);

   }
}
```

Run:

```
Enter 10 elements: 9 2 4 7 6 8 0 2 1 3
New element and its position: 5 4

Overflowed element: 3
Array after insert: 9 2 4 7 5 6 8 0 2 1
```

Problem 2 (class TruncArray)

First, write a Java method **`turncate`** that takes 2 parameters: 1) a non-empty array of integers and a number of elements to truncate from each extremity. Depending on the desired number of the elements to truncate, the method replaces with 0 (zero) all the elements to be truncated from the array, moves them (the zeros) to the end of the array, and returns the number of elements moved to the beginning of the array.

Second, write two java methods **`readArray(int[]a)`** and **`printArray(int[]a)`** that reads/writes the elements of an array passed in parameter from/to the standard input/output.

Finally, use the following **`main`** method to test your different methods:

```
public class TruncArray
{
    private static Scanner input = new Scanner(System.in);

    public static void main(String[] args)
    {
        int[] array = new int[10];
        System.out.print("Enter 10 elements: ");
        readArray(array);

        System.out.print("Number of elements to truncate: ");
        int number = input.nextInt();

        int mn = truncate(array, number);
        System.out.println("Number of moves: " + mn);

        System.out.print("Array after cut: ");
        printArray(array);

    }
}
```

Run Example 1:

```
Enter 10 elements: 1 2 3 4 5 6 7 8 9 10
Number of elements to truncate: 3

Number of moves: 4
Array after cut: 4 5 6 7 0 0 0 0 0 0
```

Tutorial 09: Arrays

Exercise 1

Write a Java method reverse() that reverses the first diagonal of a square matrix of order n (maximum value of n is 10). Write a complete program to test the method and prints out the matrix before and after the modification.

Exercise 2

Write an application that calculates the product of a series of integers that are passed to method product using a variable-length argument list. Test your method with several calls, each with a different number of arguments.

Exercise 3

Write a program called **Arithmetic** that takes three command-line arguments: two integers followed by an arithmetic operator (+, -, * or /). The program shall perform the corresponding operation on the two integers and print the result. For example:

```
java Arithmetic 3 2 +
3+2=5

java Arithmetic 3 2 -
3-2=1

java Arithmetic 3 2 /
3/2=1
```

Exercise 4

You are given 2 arrays of the same dimension. Use class Arrays methods as you can to perform the following tasks:

1. Create and fill the arrays with the same values but in different order.
2. Compare the two arrays for equality.
3. Sort the two arrays and compare them again for equality again.
4. What do you observe when comparing the arrays before and after sorting?

Problem 1

Write a complete Java program (class **`FriendsArray`**) that creates a 2-dimensional array of 5x5 values of type **`boolean`** and performs the following tasks:

a. Suppose indices represent people and that the value at row **`i`**, column **`j`** of the array is **`true`** just in case **`i`** and **`j`** are friends and **`false`** otherwise. Use initializer list to instantiate and initialize your array to represent the following configuration:

```
# | 0  1  2  3  4
--+---------------
0 |    *     *  *
1 | *     *     *
2 |    *
3 | *           *
4 | *  *     *
```

(* means "friends")

b. Write some code to count and display how many pairs of friends are represented in the array. (Note that each friendship pair appears twice in the array, so in the example above there are 6 pairs of friends not 12).

Problem 2 (class CheckOrdered)

First, write a Java method **`isOrdered`** that takes as parameters a non-empty 2-dimensional array of integers. The method checks whether the elements of the array in ascendant order or not.

Second, write a java methods **`printArray(int[][]a)`** that writes the elements of an array passed in parameter to the standard output.

Finally, use the following **`main`** method to test your different methods:

```
public class CheckOrder
{
  public static void main(String[] args)
  {
    int[][] array1 = { {10,15,32,40},{60,73,81,100},{200,275,300,303} };
    int[][] array2 = {{1, 3 , 2,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};

    if (isOrdered(array1))
    {
      System.out.println("The following array elements are ordered:");
      printArray(array1);
    }

    if (isOrdered(array2))
    {
      System.out.println("The following array elements are ordered:");
      printArray(array2);
    }
  }
}
```

Run Example:

```
The following array elements are ordered:
10 15 32 40
60 73 81 100
200 275 300 303
```

Tutorial 10: Strings

Exercise 1

Run the following code and analyze its output. Explain what is the difference between equality operator (==) when used to compare strings and the equals() method.

```
int i = 30;
int j = 30;

if (i == j)
    System.out.println("i and j are equal");

String s1 = new String("SAR");
String s2 = s1;
String s3 = new String("Sar");

if (s1 == s2)
    System.out.println("s1 and s2 are same");

if (s1.equals(s2))
    System.out.println("s1 and s2 are equal by equals()");

if (s1.compareTo(s2) == 0)
    System.out.println("s1 and s2 are equal by compareTo()");


if (s1 == s3)
    System.out.println("s1 and s3 are same");

if (s1.equals(s3))
    System.out.println("s1 and s3 are equal by equals()");

if (s1.equalsIgnoreCase(s3))
    System.out.println("s1 and s3 are equal by equalsIgnoreCase()");

if (s1.compareTo(s3) == 0)
    System.out.println("s1 and s3 are equal by compareTo()");
```

Exercise 2 (FindOccurrences)

1. Define a method `int getOccurrenceNumber(String s, char c)` that returns the number of occurrences of character c in string s (use a loop and `charAt()` to compare c with the current character)
2. Define a method `void printOccurrencePositions(String s, char c)` the prints the index (position) of every occurrence of the character c in the string s (use a loop and `indexOf()`)
3. Write the following code fragment in the main method to test your methods:

```
    String s1 = new String("Lorem ipsum dolor sit amet, consectetur adipiscing
    elit. Nullam sed blandit orci, vel convallis ex. Integer ut aliquam ligula,
    a finibus ligula. Proin pharetra maximus ligula nec eleifend");

    char c = 'd';
    System.out.println("S1: " + s1);

    System.out.printf("'%c' appears '%d' times is S1\n", c,
getOccurrenceNumber(s1, c));

    System.out.printf("'%c' appears in positions: ", c);
    printOccurrencePositions(s1, c);
```

Exercise 3

Using alternatively the methods regionMatches and endsWith from the class String, define a method that takes a site name and returns its domain:

- if the site name ends with ".com" the domain is Commercial
- if the site name ends with ".edu" the domain is Educational
- if the site name ends with ".sa" the domain is Saudi Arabia
- if the site name ends with ".info" the domain is Informational

Test your method with a main method by asking the user to enter a site name, then pass it to the method, and print out the returned value.

Problem (class FindTags)

Write a Java method **`find()`** that receives as parameter a string representing a text in a specific format and prints all the tags found in the text and their number. A tag is delimited by '<' and '>' and you have to consider all possible positions of the tags: at the beginning of the text, at the end of the text, and inside the text.

To test your method, write a **`main()`** method that enables the user to input a string (using **`nextLine()`** of the **`Scanner`**), removes eventual extra spaces from it, and sends it to **`find()`**.

Run Example 1:

```
Enter a string: <html><head><title>CS140 Home Page<body><h1>Course
Description <p>
Tags are:
      -html
      -head
      -title
      -body
      -h1
      -p
6 tag(s) was(were) found.
```

Run Example 2:

```
Enter a string: CS140 Homepage & Course Description
Tags are:

Sorry! No tag found
```

Tutorial 11: Files

Exercise 1

Consider the file named `file.txt` which contains personal information (id, name, salary):

```
A123 Anna 3000
A234 Alex 4000
A986 Jame 5000
```

Write a java application that allows the user to enter an ID of a person and additional salary, then it will update the file.

The program output will look like:

```
Enter ID: A123
Enter additional salary: 2000
```

The content of the file will be (after running the program):

```
A123 Anna 5000
A234 Alex 4000
A986 Jame 5000
```

Exercise 2

Consider the file named `results.txt` which contains students information (id, name, marks):

```
100 Fahd 93
101 Omar 50
102 Anas 58
```

Complete the following program that reads a file (where the file name is provided to the main method as parameter) that contains the students' information, adds 2 to the total marks (without updating the file), and prints student information and whether is passed or not.

```
public class ProcessGrades {
    public static void main(String[] args) throws Exception {
        String fileName = args[0];
        Scanner s = new Scanner(Paths.get(fileName));
        while(s.hasNext()) {
            // add you code here
        }
    }
}
```

You must pass the name of the file as option while running the file. So you can run your program using a command line similar to: `java ProcessGrades results.txt`

It is supposed to have the following output (after running the program):

```
100 Fahd 95.00 Passed
101 Omar 52.00 Failed
102 Anas 60.00 Passed
```