# Exercise: Java Basics

This exercise is part of the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

# Part I: Data Types and Methods

## Variable in Hexadecimal Format

Write a program that reads a number in **hexadecimal format** convert it to **decimal format** and prints it.

|  |  |
| --- | --- |
| **Input** | **Output** |
| FE | 254 |
| 37 | 55 |
| 10 | 16 |

### Hints

* Use [**Integer.parseInt(string, base)**](http://www.tutorialspoint.com/java/number_parseint.htm).

## Boolean Variable

Write a program that reads a **string**, converts it to **Boolean** variable and **prints** “**Yes**”if the variable is true and “**No**” if the variable is false.

|  |  |
| --- | --- |
| **Input** | **Output** |
| True | Yes |
| False | No |

### Hints

* You can parse Booleans.

## Reverse Characters

Write a program to ask the user for **3 letters** and print them in **reversed order**.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| A  B  C | CBA |  | x  Y  z | zYx |  | G  g  n | ngG |

## Vowel or Digit

Create a program to check if given symbol is **digit**, **vowel** or any **other symbol**.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| a | vowel |  | 9 | digit |  | g | other |

## Integer to Hex and Binary

Create a program to convert a **decimal number** to **hexadecimal** and **binary** number and print it.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 10 | A  1010 |  | 420 | 1A4  110100100 |  | 256 | 100  100000000 |

### Hints

* There are methods, that convert Integer to Hex and Binary.

# Part II: Arrays

## Compare Char Arrays

Compare two char arrays lexicographically (letter by letter).

Print the them in alphabetical order, each on separate line.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| a b c  d e f | abc  def |
| p e t e r  a n n i e | annie  peter |
| a n n i e  a n | an  annie |
| a b  a b | ab  ab |

### Hints

* Compare the first letter of arr1[] and arr2[], if equal, compare the next letter, etc.
* If all letters are equal, the smaller array is the **shorter**.
* If all letters are equal and the array lengths are the same, the arrays are **equal**.

## Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 1 1 2 3 3 **2 2 2** 1 | 2 2 2 |
| **1 1 1** 2 3 1 3 3 | 1 1 1 |
| **4 4 4 4** | 4 4 4 4 |
| 0 **1 1** 5 2 2 6 3 3 | 1 1 |

### Hints

* Start with the sequence that consists of the first element: start=0, len=1.
* Scan the elements from left to right, starting at the second element: pos=1…n-1.
  + At each step compare the current element with the element on the left.
    - Same value 🡪 you have found a sequence longer by one 🡪 len++.
    - Different value 🡪 start a new sequence from the current element: start=pos, len=1.
  + After each step remember the sequence it is found to be longest at the moment: bestStart=start, bestLen=len.
* Finally, print the longest sequence by using bestStart and bestLen.

## Max Sequence of Increasing Elements

Write a program that finds the **longest increasing subsequence** in an array of integers. The longest increasing subsequence is a **portion of the array** (subsequence) that is strongly **increasing** and has the **longest possible length**. If several such subsequences exist, find the left most of them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 **2 3 4** 2 2 4 | 2 3 4 |
| 4 5 **1 2 3 4 5** | 1 2 3 4 5 |
| **3 4 5 6** | 3 4 5 6 |
| **0 1** 1 2 2 3 3 | 0 1 |

### Hints

* Use the same algorithm like in the previous problem (Max Sequence of Equal Elements).

## Most Frequent Number

Write a program that finds the **most frequent number** in a given sequence of numbers.

* Numbers will be in the range [0…65535].
* In case of multiple numbers with the same maximal frequency, print the left most of them.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Output** |
| **4** 1 1 **4** 2 3 **4 4** 1 2 **4** 9 3 | 4 | The number **4** is the most frequent (occurs 5 times) |
| **2 2 2 2** 1 **2 2 2** | 2 | The number **2** is the most frequent (occurs 7 times) |
| **7 7 7** 0 2 2 2 0 10 10 10 | 7 | The numbers **2**, **7** and **10** have the same maximal frequence (each occurs 3 times). The leftmost of them is **7**. |

## Index of Letters

Write a program that creates an array containing all letters from the alphabet (**a**-**z**). Read a lowercase word from the console and print the **index of each of its letters in the letters array**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| abcz | a -> 0  b -> 1  c -> 2  z -> 25 |
| softuni | s -> 18  o -> 14  f -> 5  t -> 19  u -> 20  n -> 13  i -> 8 |

## Equal Sums

Write a program that determines if there **exists an element in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **“no”** if there is no such index.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 3 | 2 | At a[2] -> left sum = 3, right sum = 3  a[0] + a[1] = a[3] |
| 1 2 | no | At a[0] -> left sum = 0, right sum = 2  At a[1] -> left sum = 1, right sum = 0  No such index exists |
| 1 | 0 | At a[0] -> left sum = 0, right sum = 0 |
| 1 2 3 | no | No such index exists |
| 10 5 5 99 3 4 2 5 1 1 4 | 3 | At a[3] -> left sum = 20, right sum = 20  a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] |

# Part IV: Strings, Maps and Stream API

## Reverse String

Write a program that reads a string from the console, **reverses** its letters and prints the result back at the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| sample | elpmas |
| 24tvcoi92 | 29iocvt42 |

### Hints

* **Variant I**: convert the string to **char array**, **reverse** it, then convert it to **string** again.
* **Variant II**: print the letters of the string in back direction (from the last to the first) in a for-loop.

## Fit String in 20 Chars

Write a program that **reads** from the console a string and **fits the string in 20 characters** as follows:

* If the string has **less than 20 characters**, append some ‘\*’ until it gets length of exactly 20 characters.
* If the string length is **more than 20 characters**, discard all characters after the first 20.

**Print** the result string at the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Welcome to SoftUni! | Welcome to SoftUni!\* |
| A "regular expression" (abbreviated regex or regexp) is a sequence of characters that forms a search pattern. | A "regular expressio |
| C# | C#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

### Hints

* If string length < 20, use PadRight(20, '\*').
* If string length > 20, use Substring(0, 20).

## Censor Email Address

You have some text that contains your email address. You are sick of spammers, so you want to **hide** it. You decide to **censor** your email: to **replace all characters** in it with asterisks ('\*') **except the domain**.

Assume your email address will always be in format **[username]@[domain]**. You need to replace the username with asterisks of equal number of letters and keep the domain unchanged.

### Input

* The first line holds your **email** address.
* The second line holds a **text** where the email should be censored.

### Examples

|  |
| --- |
| **Input** |
| pesho.peshev@email.bg  My name is Pesho Peshev. I am from Sofia, my email is: pesho.peshev@email.bg (not pesho.peshev@email.com). Test: pesho.meshev@email.bg, pesho.peshev@email.bg |
| **Output** |
| My name is Pesho Peshev. I am from Sofia, my email is: \*\*\*\*\*\*\*\*\*\*\*\*@email.bg (not pesho.peshev@email.com). Test: pesho.meshev@email.bg, \*\*\*\*\*\*\*\*\*\*\*\*@email.bg |

### Hints

In order to accomplish the task, you may find these steps useful:

* **Split** the email into two parts – **username** and **domain**.
* Create the **replacement** string by duplicating the '\*' character username.length and appending '@' and the **domain**.
* **Replace** all occurrences of your **email** with the **replacement string**.

## URL Parser

Write a program that **parses an URL** given in the following format:

[protocol]://[server]/[resource]

The parsing extracts its parts: protocol, server and resource.

* The **[server]** part is mandatory.
* The **[protocol]** and **[resource]** parts are optional.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| http://www.abc.com/video | [protocol] = "http"  [server] = "www.abc.com"  [resource] = "video" |
| https://www.softuni.bg/Resources/Materials | [protocol] = "https"  [server] = "www.softuni.bg"  [resource] = "Resources/Materials" |
| ftp://www.su.us/TestResource | [protocol] = "ftp"  [server] = "www.su.us"  [resource] = "TestResource" |
| https://softuni.bg | [protocol] = "https"  [server] = "softuni.bg"  [resource] = "" |
| www.nakov.com | [protocol] = ""  [server] = "www.nakov.com"  [resource] = "" |

### Hints

* Find the leftmost occurrence of “://” in the input URL.
  + If **found**, the left side holds the **protocol**, the right side holds the **server + resource**.
  + If **not found**, the protocol is missing, the input string holds **server + resource** only.
* After the “protocol” part is removed from the input URL, find the leftmost occurrence of “/”.
  + If **found**, the left side holds the **server**, the right side holds the **resource**.
  + If **not found**, the resource is missing, the whole string holds the **server**.

## Change to Uppercase

We are given a text. Write a program that **modifies the casing of letters to uppercase** at all places **in the text surrounded** **by** <upcase> **and** </upcase> **tags**. Tags cannot be nested.

### Example

|  |
| --- |
| **Input** |
| Welcome to the **<upcase>Software University</upcase>**. Learn **<upcase>computer programming</upcase>** and start a **<upcase>job</upcase>** in a software company. |
| **Output** |
| Welcome to the **SOFTWARE UNIVERSITY**. Learn **COMPUTER PROGRAMMING** and start a **JOB** in a software company. |

### Hints

* You may find the position of the first <upcase> and the first </upcase>, delete the text between and insert the uppercase version of the text without the tags at the position of <upcase>.
* Repeat the above until no more <upcase> and </upcase> tags are found in the text.

## Phonebook

Write a program that receives some info from the console about **people** and their **phone numbers**. Each **entry** should have just **one name** and **one number** (both of them strings).

On each line you will receive some of the following commands:

* **A {name} {phone}** – adds entry to the phonebook. In case of trying to add a name that is already in the phonebook you should change the existing phone number with the new one provided.
* **S {name}** – searches for a contact by given name and prints it in format "**{name} -> {number}**". In case the contact isn't found, print "**Contact {name} does not exist.**".
* **END** – stop receiving more commands.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| A Nakov 0888080808  S Mariika  S Nakov  END | Contact Mariika does not exist.  Nakov -> 0888080808 |
| A Nakov +359888001122  A RoYaL(Ivan) 666  A Gero 5559393  A Simo 02/987665544  S Simo  S simo  S RoYaL  S RoYaL(Ivan)  END | Simo -> 02/987665544  Contact simo does not exist.  Contact RoYaL does not exist.  RoYaL(Ivan) -> 666 |
| A Misho +359883123  A Misho 02/3123  S Misho  END | Misho -> 02/3123 |

### Hints

* **Parse the commands** by splitting by space. Execute the commands until “**END**” is reached.
* Store the **phonebook entries** in LinkedHashMap<String, String> with key **{name}** and value **{phone number}**.

## Phonebook Upgrade

**Add functionality to the phonebook** from the previous task to **print all contacts ordered lexicographically** when receive the command “ListAll”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| A Nakov +359888001122  A RoYaL(Ivan) 666  A Gero 5559393  A Simo 02/987665544  ListAll  END | Gero -> 5559393  Nakov -> +359888001122  RoYaL(Ivan) -> 666  Simo -> 02/987665544 |

### Hints

* **Variant I (slower):** Sort all entries in the dictionary by key and print them.
* **Variant II (faster):** Keep the entries in more appropriate data structure that will keep them in sorted order for better performance.

# Part V: Classes and Objects

## Count Working Days

Write a program that **reads two dates** in formatdd-MM-yyyy and prints the **number of working days** between these two dates **inclusive**. Non-working days are:

* All days that are **Saturday** or **Sunday**.
* All days that are **official holidays** in Bulgaria:
  + New Year Eve (**1 Jan**)
  + Liberation Day (**3 March**)
  + Worker’s day (**1 May**)
  + Saint George’s Day (**6 May**)
  + Saints Cyril and Methodius Day (**24 May**)
  + Unification Day (**6 Sept**)
  + Independence Day (**22 Sept**)
  + National Awakening Day (**1 Nov**)
  + Christmas (**24**, **25** and **26 Dec**)

All days not mentioned above are **working** and should count.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 11-04-2016  14-04-2016 | 4 |
| 11-04-2016  22-04-2016 | 10 |
| 20-12-2015  31-12-2015 | 7 |

### Hints

* Read **start date** and **end date** from Console.
* **Create** two objects of type Date – startDate and endDate.
* Create an **array of type** Date and add **all official holidays** in it.
* Loop from startDate to endDate. Add **1 day** at each iteration.
* Get the **current da**y in the loop and check whether is **Saturday**, **Sunday** or it is **contained** **in the holidays array**. If it is not, increment the workingDaysCounter.

## Advertisement Message

Write a program that **generate random fake advertisement message** to extol some product. The messages must consist of 4 parts: laudatory **phrase** + **event** + **author** + **city**. Use the following predefined parts:

* **Phrases** – {“Excellent product.”, “Such a great product.”, “I always use that product.”, “Best product of its category.”, “Exceptional product.”, “I can’t live without this product.”}
* **Events** – {“Now I feel good.”, “I have succeeded with this product.”, “Makes miracles. I am happy of the results!”, “I cannot believe but now I feel awesome.”, ”Try it yourself, I am very satisfied.”, “I feel great!”}
* **Author** – {“Diana”, “Petya”, “Stella”, “Elena”, “Katya”, “Iva”, “Annie”, “Eva”}
* **Cities** – {“Burgas”, “Sofia”, “Plovdiv”, “Varna”, “Ruse”}

The format of the output message is: **{phrase} {event} {author} – {city}**.

As an input you take the **number of messages** to be generated. Print each random message at a separate line.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | Such a great product. Now I feel good. Elena – Ruse  Excelent product. Makes miracles. I am happy of the results! Katya – Varna  Best product of its category. That makes miracles. Eva - Sofia |

### Hints

* Hold the phrases, events, authors and towns in 4 arrays of strings.
* Create Random object and **generate** **4 random numbers** each in its range:
  + phraseIndex 🡪 ­[0, phrases.Length)
  + eventIndex 🡪 [0, events.Length)
  + authorIndex 🡪 [0, authors.Length)
  + townIndex 🡪 [0, towns.Length)
* Get one **random element** from each of the four arrays and **compose a message** in the required format.

## Intersection of Circles

Create class Circle with properties Center and Radius. The center is a point with coordinates X and Y (make a class Point). Write a method bool Intersect(Circle c1, Circle c2) that tells whether the two given circles **intersect or not**. Write a program that tells if two circles intersect.

The input lines will be in format: **{X} {Y} {Radius}**. Print as output “Yes” or “No”.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Visualization** |
| 4 4 2  8 8 1 | No |  |
| 3 3 2  4 3 6 | Yes |  |
| 1 1 4  4 2 5 | Yes |  |

### Hints

* Calculate d = **distance between the circle centers**.
* If the d ≤ r1 + r2 (the sum of radiuses**) 🡪** the circles **intersect** (or one of the circles is inside the other or the circles have one common point when d = r1 + r2).
* If the d > r1 + r2 **🡪** the circles do **not intersect** (they have not common shared point).

## Average Grades

Define a class Student, which holds the following information about students: **name**, **list of grades** and **average grade** (calculated property, read-only). A single grade will be in range [2…6], e.g. 3.25 or 5.50.

Read a **list of students** and print the students that have **average grade ≥ 5.00** ordered **by name** (ascending), then by **average** **grade** (descending). Print the student name and the calculated average grade.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Ivan 3  Todor 5 5 6  Diana 6 5.50 | Diana -> 5.75  Todor -> 5.33 |
| 6  Petar 3 5 4 3 2 5 6 2 6  Mitko 6 6 5 6 5 6  Gosho 6 6 6 6 6 6  Ani 6 5 6 5 6 5 6 5  Iva 4 5 4 3 4 5 2 2 4  Ani 5.50 5.25 6.00 | Ani -> 5.58  Ani -> 5.50  Gosho -> 6.00  Mitko -> 5.67 |

### Hints

* Create class Student with properties Name (string), Grades (double[]), and property AverageGrade (calculated by LINQ as Grades.Average(), read-only).
* Make a **list of students** and filter **with** STREAMall students that has average grade >= 5.00.
* Print the filtered students **ordered by name** in ascending order, then by **average grade** in descending order.

## Book Library

To model a **book library**, define classes to hold a **book** and a **library**. The library must have a **name** and a **list of books**. The books must contain the **title**, **author**, **publisher**, **release date**, **ISBN-number** and **price.**

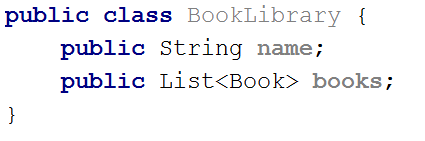
Read a **list of books**, add them to the library and print the **total sum of prices by author**,ordered **descending by price** and **then by author’s name lexicographically**.

Books in the input will be in format **{title} {author} {publisher} {release date} {ISBN} {price}**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00  Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25  HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50  HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00  AC OBowden PenguinBooks 20.11.2009 0395082555 14.00 | Tolkien -> 40.25  JKRowling -> 35.50  OBowden -> 14.00 |

### Hints

* Create classes Book and Library with all the mentioned above properties: 
* **Create** an object of type Library.
* **Read the input** and create a Bookobject for each book in the input.
* Create a **STREAM** query that will **sum the prices by author**, **order the results** as requested.
* **Print** the results.

## Book Library Modification

Use the classes from the previous problem and make a program that **read a list of books** and **print all titles** **released after given date** ordered **by date** and then **by** **title lexicographically**. The date is given if format “**day-month-year**” at the last line in the input.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00  Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25  HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50  HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00  AC OBowden PenguinBooks 20.11.2009 0395082555 14.00  30.07.1954 | HP1 -> 26.06.1997  HP7 -> 21.07.2007  AC -> 20.11.2009 |