# Lab: Objects and Classes

Problems for exercises and homework for the ["Programming Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3448/programming-fundamentals-with-java-january-2022)

You can check your solutions in [Judge.](https://judge.softuni.bg/Contests/1319)

# Using the Built-in Java Classes

## Randomize Words

You are given a **list of words in one line**. **Randomize their order** and print each word on a separate line.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Welcome to SoftUni and have fun learning programming | learning  Welcome  SoftUni  and  fun  programming  have  to | The order of the words in the output will be different after each program execution. |
| Java is the best programming language | the  programming  best  language  is  Java |  |

### Hints

* **Split** the input string (by space) and create an **array of words.**
* Create a random number generator - an object rnd of type **Random.**
* In a **for-loop exchange each number** at positions 0, 1, …, words.Length-1 by a number at a **random. position**. To generate a random number in rangeuse **rnd.**nextInt(words.length).
* Print each word in the array on a new line.

## Sum Big Numbers

You will receive two numbers **(0 to 1050),** print their sum.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 923847238931983192462832102  934572893617836459843471846187346 | 934573817465075391826664309019448 |
| 4  100 | 104 |

### Hints

Use the class [**BigInteger**](https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html)

Import the namespace "java.math.BigInteger":



Use the type BigInteger to read the numbers and calculate their sum:



## Big Factorial

You will receive **N** - the number in the range **[0 - 1000]**. Calculate the **Factorial** of **N** and print the result.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 | 120 |
| 50 | 30414093201713378043612608166064768844377641568960512000000000000 |

# Defining Simple Classes

## Songs

Define a class Song, which holds the following information about songs: **Type List**, **Name,** and **Time**.

On the first line, you will receive the **number of songs** **-** **N**.

On the **next N-lines** you will be receiving data in the following format: "{typeList}\_{name}\_{time}"**.**

On the last line, you will receive "**Type List"** / **"all".** Print only the **names of the songs** which are from that **Type List** / **All songs**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  favourite\_DownTown\_3:14  favourite\_Kiss\_4:16  favourite\_Smooth Criminal\_4:01  favourite | DownTown  Kiss  Smooth Criminal |
| 4  favourite\_DownTown\_3:14  listenLater\_Andalouse\_3:24  favourite\_In To The Night\_3:58  favourite\_Live It Up\_3:48  listenLater | Andalouse |
| 2  like\_Replay\_3:15  ban\_Photoshop\_3:48  all | Replay  Photoshop |

### Solution

Define class Song with fields: **Type List**, **Name,** and **Time**:



Define getters and setters: use keys **ALT + INS** and generate Getter and Setter:



Read the input lines, make the collection, and store the data:



Finally, read your last line – **Type List** and print the result:



You can use **Stream API** to filter the collection:



## Students

Define a class Student, which holds the following information about students: first name, last name, age, and hometown.

Read the list of students until you receive the "**end**" command. After that, you will receive a city name. Print only students which are from the given city, in the following format: **"{firstName} {lastName} is {age} years old"**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| John Smith 15 Sofia  Peter Ivanov 14 Plovdiv  Linda Bridge 16 Sofia  Simon Stone 12 Varna  end  Sofia | John Smith is 15 years old  Linda Bridge is 16 years old |
| Anthony Taylor 15 Chicago  David Anderson 16 Washington  Jack Lewis 14 Chicago  David Lee 14 Chicago  end  Chicago | Anthony Taylor is 15 years old  Jack Lewis is 14 years old  David Lee is 14 years old |

### Solution

Define a class student with the following properties: **firstName**, **lastName**, **age,** and **city**:



Generate constructor in class Student**: ALT + INSERT**



Read a list of students.



Read a city name and print only the students which are from the given city.

You can filter the students with the stream.



## Students 2.0

Use the class from the previous problem. If you receive a student which already exists (first name and last name should be **unique**) overwrite the information.

### Hints

Check if the given student already exists:





Overwrite the student information.

First, we have to find the existing student:





Finally, we have to overwrite the information:



We can use **Stream API** as well:



findFirst returns the first occurrence or null.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| John Smith 15 Sofia  John Smith 16 Sofia  Linda Bridge 17 Sofia  Simon Stone 12 Varna  end  Sofia | John Smith is 16 years old  Linda Bridge is 17 years old |
| J S 3 S  Peter Ivanov 14 P  P J 104 S  J P 61 S  Simon Stone 12 Varna  Simon Sone 12 Varna  end  Varna | Simon Stone is 12 years old  Simon Sone is 12 years old |