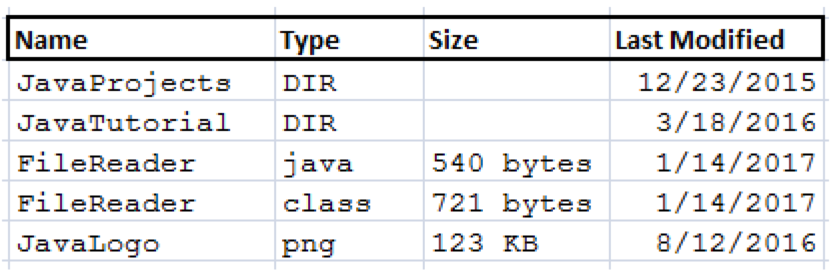
* Exercise #1: File I/O - Directory Listing

Write a program to print the content of a given directory to a text file in the following format:



Name	Type	Size	Last Modified
JavaProjects	DIR		12/23/2015
JavaTutorial	DIR		3/18/2016
FileReader	java	540 bytes	1/14/2017
FileReader	class	721 bytes	1/14/2017
JavaLogo	png	123 KB	8/12/2016

The column **Type** shows the extension of the files, or DIR if a file is a directory.

The column **Size** shows the size of the file in bytes, KB, MB, GB and so on. If a file is a directory, the size is blank.

The column **Last Modified** shows the last modification time of a file in `mm/dd/YYYY` format.

Run this program from command line like this:

```
java PrintDir < dir_path >
```

For example:

```
java PrintDir /home/john/Java
```

The result is a text file having the same name with the directory created, e.g. Java.txt - this file contains the content of the directory in the format mentioned above.

* Exercise #2: Serialization

Given the following class:

```
import java.util.*;

public class Programmer {

      private String name;

      private String email;

      private Date birthday;

      private List languages;

}
```

Let update this class like this:

- Implement getter and setter methods for all fields.

- Write an empty constructor.

- Write second constructor that takes arguments for all fields.

And write a program called `ProgrammerWriter` that asks the programmer's information from the user in command line interface, in the following manner (the underlined text is sample input):

```
- How many programmers do you want to record? 3
```

For each programmer, ask the following information:

```
- Name: John

- E-mail: john@gmail.com

- Birthday (mm/dd/YYY): 10/11/1990

- Programming languages: Java, C, C++, PHP
```

Note that the programming languages in the input must be separated by commas, and are store as List of String.

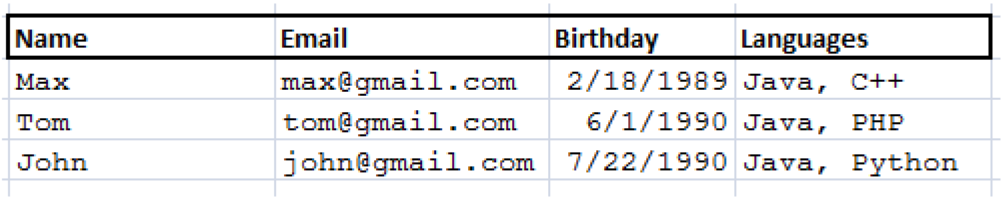
Finally the program asks for the file name:

```
      - Store in file name: programmers.dat
```

Then the program saves all the programmer's information in the specified file.

* Exercise #3: De-serialization

Write a program called `ProgrammerReader` that reads the programmer's information stored by the `ProgrammerWriter` program. The program prints the information to the standard console in the following format:



Name	Email	Birthday	Languages
Max	max@gmail.com	2/18/1989	Java, C++
Tom	tom@gmail.com	6/1/1990	Java, PHP
John	john@gmail.com	7/22/1990	Java, Python

Run this program like this:

```
      java ProgrammerReader < file_path >
```

For example:

```
java ProgrammerReader < file_path >
```

* Exercise #4: Compression

Update the `ProgrammerWriter` program in order to save data in compressed format (ZIP file).

Hint: Compress the file generated by the program in the exercise #2 to a ZIP file (e.g. `Programmers.zip`), then remove the original file.

* Exercise #5: Decompression

Update the `ProgrammerReader` program (exercise #3) in order to read the compressed file generated by the program in the exercise #4.

* Exercise #6: Random Access File

Using random access file to write a program that reads metadata of a MP4 file. Run the program like this:

```
       java Mp4Parser < mp4_file >
```

For example:

```
java Mp4Parser JavaIOTutorial.mp4
```

Then the program prints out the following information:

```
     - Length: 00:15:32
- Frame width: 1920
- Frame height: 1080
- Frame rate: 29 frames/second
```

Hint: You need to study the MP4 file format first.