# Lab: Encapsulation

This document defines the lab for the ["Java Advanced" course @ Software University](https://softuni.bg/modules/59/java-advanced). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/1535/Encapsulation-Lab).

<https://www.youtube.com/watch?v=pJzYP253_K4&t=7444s>

<https://softuni.bg/trainings/resources/video/69701/video-24-february-2022-dimo-georgiev-java-oop-february-2022/3587>

## Sort by Name and Age

Create a class **Person**, which should have **private** fields for:

* firstName: String
* lastName: String
* age: int
* toString() - **override**

You should be able to use the class like this:

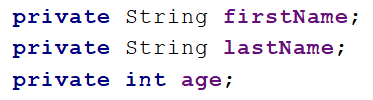
|  |
| --- |
| Main.java |
| **public static void** main(String[] args) **throws** IOException {  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  **int** n = Integer.*parseInt*(reader.readLine());   List<Person> people = **new** ArrayList<>();   **for** (**int** i = 0; i < n; i++) {  String[] input = reader.readLine().split(**" "**);  people.add(**new** Person(input[0], input[1], Integer.*parseInt*(input[2])));  }   Collections.*sort*(people, (firstPerson, secondPerson) -> {  **int** sComp = firstPerson.getFirstName().compareTo(secondPerson.getFirstName());   **if** (sComp != 0) {  **return** sComp;  } **else** {   **return** Integer.compare(firstPerson.getAge(), secondPerson.getAge());  }  });   **for** (Person person : people) {  System.***out***.println(person.toString());  } } |

### Examples

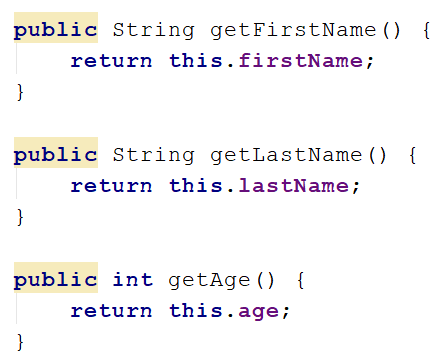
|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Angel Ivanov 65  Boris Georgiev 57  Veny Ivanov 27  Angel Harizanov 44  Boris Angelov 35 | Angel Harizanov is 44 years old.  Angel Ivanov is 65 years old.  Boris Angelov is 35 years old.  Boris Georgiev is 57 years old.  Veny Ivanov is 27 years old. |
| 4  Sara Cameron 21  John Petrovich 53  Anna Glen 21  John Alekseevich 43 | Anna Glen is 21 years old.  John Alekseevich is 43 years old.  John Petrovich is 53 years old.  Sara Cameron is 21 years old. |

### Solution

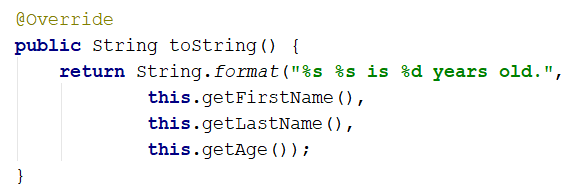
Create a **new class** and ensure **proper naming**. Define the **private** fields:



Create getters and apply them access modifiers, which are as strict as possible:



Override **toString()** method:



## Salary Increase

Read person with their **names**, **age,** and **salary**. Read **percent bonus** to every person salary. People younger **than 30** get a half bonus. Expand **Person** from the previous task. Add **salary** **field** and **getter** and **setter** with proper **access**.

New **fields** and **methods**

* salary: double
* increaseSalary(double bonus)

You should be able to use the class like this:

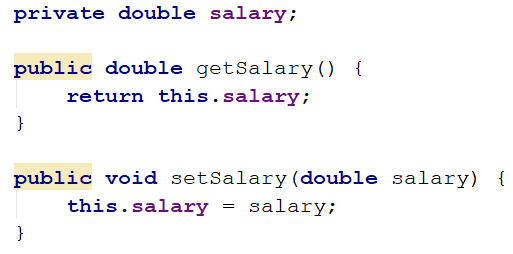
|  |
| --- |
| Main.java |
| **public static void** main(String[] args) **throws** IOException {  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  **int** n = Integer.*parseInt*(reader.readLine());  List<Person> people = **new** ArrayList<>();  **for** (**int** i = 0; i < n; i++) {  String[] input = reader.readLine().split(**" "**);  people.add(**new** Person(input[0], input[1], Integer.*parseInt*(input[2]), Double.*parseDouble*(input[3])));  }  **double** bonus = Double.*parseDouble*(reader.readLine());  **for** (Person person : people) {  person.increaseSalary(bonus);  System.***out***.println(person.toString());  } } |

### Examples

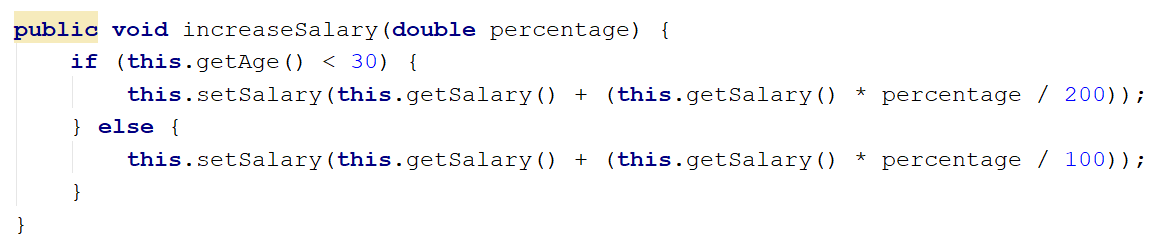
|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Angel Ivanov 65 2200  Boris Georgiev 57 3333  Veny Ivanov 27 600  Angel Harizanov 44 666.66  Boris Angelov 35 559.4  20 | Angel Ivanov gets 2640.0 leva  Boiko Georgiev gets 3999.6 leva  Veny Ivanov gets 660.0 leva  Angel Harizanov gets 799.992 leva  Boris Angelov gets 671.28 leva |
| 4  Sara Cameron 21 1200  John Petrovich 53 850.50  Anna Glen 21 1640  John Alekseevich 43 2100  13 | Sara Cameron gets 1278.0 leva  John Petrovich gets 961.065 leva  Anna Glen gets 1746.6 leva  John Alekseevich gets 2373.0 leva |

### Solution

Add a new **private** field for **salary** and proper **setters** and **getters** for it:



Add new **method**, which will **increase** salary with a bonus:



Refactor **constructor** and **toString()** method for this task.

## Validation Data

Expand Person with proper validation for every field:

* **Names must be at least 3 symbols**
* **Age must not be zero or negative**
* **Salary can't be less than 460.0**

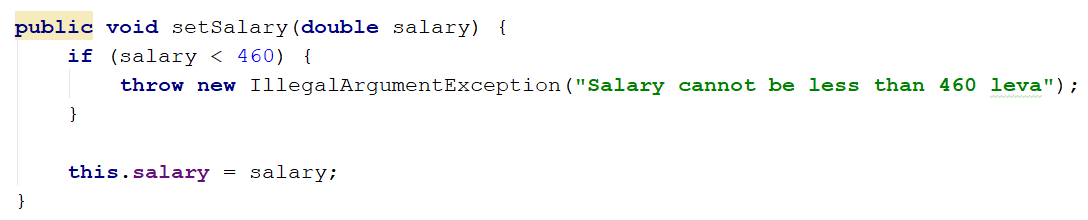
Print proper message to end-user (look at an example for messages).

Don't use **System.out.println()** in Person class.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Asen Ivanov -6 2200  B Borisov 57 3333  Ventsislav Ivanov 27 600  Asen H 44 666.66  Boiko Angelov 35 300  20 | Age cannot be zero or negative integer  First name cannot be less than 3 symbols  Last name cannot be less than 3 symbols  Salary cannot be less than 460 leva  Ventsislav Ivanov gets 660.0 leva |
| 4  Sara Cameron 21 1200  John Petrovich -53 850.50  Anna Glen 21 230  John Alekseevich 0 2100  13 | Age cannot be zero or negative integer  Salary cannot be less than 460 leva  Age cannot be zero or negative integer  Sara Cameron gets 1278.0 leva |

### Solution

Add **validation** to all **setters** in Person. Validation may look like this or something similar: 

## First and Reserve Team

Create a Team class. Add to this team all people you read. All people **younger** than 40 go on the **first team**, others go on the **reverse team**. At the end print first and reserve team sizes.

The class should have **private fields** for:

* name: String
* firstTeam: List<Person>
* reserveTeam: List<Person>

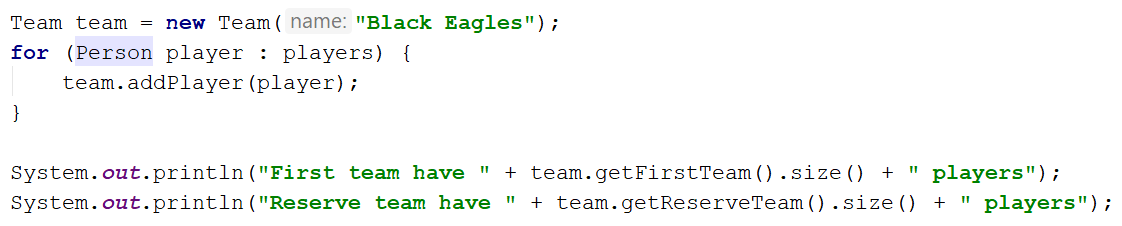
The class should have **constructors**:

* Team(String name)

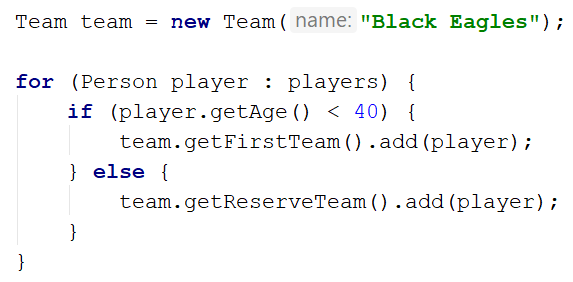
The class should also have private method for setName and **public methods** for:

* getName(): String
* addPlayer(Person person): void
* getFirstTeam(): List<Person> (Collections.unmodifiableList)
* getReserveTeam(): List<Person> (Collections.unmodifiableList)

You should be able to use the class like this:



You should **NOT** be able to use the class like this:



### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  Asen Ivanov 20 2200  Boiko Borisov 57 3333  Ventsislav Ivanov 27 600  Grigor Dimitrov 25 666.66  Boiko Angelov 35 555 | First team have 4 players  Reserve team have 1 players |
| 4  Sara Cameron 21 1200  John Petrovich 53 850.50  Anna Glen 21 475  John Alekseevich 27 2100 | First team have 3 players  Reserve team have 1 players |

### Solution

Add a new class Team. Its fields and constructor look like this:



firstTeam and reserveTeam have only **getters**:



There will be only one method, which adds players to teams:

