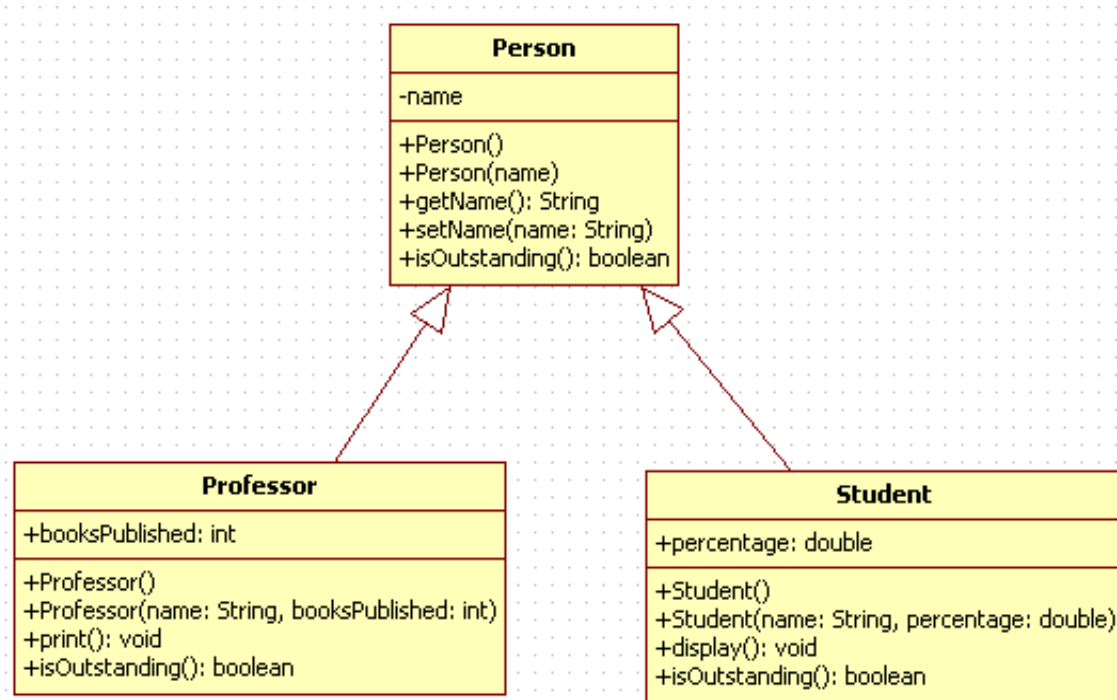Outstanding: Distinguished from others in excellence.
We judge a person if he is outstanding or not by the achievement in his/her profession

Write an application to perform the following operations:
1) Complete the entity classes shown in Class diagram
2) Implement isOutstanding( ) based on the following rules:
    a. Professor is outstanding if he has published more than 4 books
    b. Student is outstanding if his percentage is greater than or equal to 85.
3) The print ( ) method of Professor displays the name and books published by professor.
4) The display( ) method of Student displays the name and percentage of student
5) Complete the main ( ) method of PersonExample.java to implement the following:
    a. Store 5 persons in a single collection of array type (Person[ ]) , this can be a combination of students and professors
    b. Using Run-Time Type Identification print Professor and Student details.
    c. Identify all outstanding persons

2. **Frequency Count**

Count the word frequency.

Write a program to generate a frequency count of words present in a string. The program has to display the words in sorted order and its frequency.

If the input is: "java is oop and java is distributed and java is simple".

Output should be: [and = 2, distributed = 1, is = 3, …]

Hint: Method should have the signature

Map<String, Integer> getWordFrequency(String input)

3) Using REFLECTION API

The method accepts any entity object as an argument and returns an XML data as listed below:

| #	Input	Output

1	Employee employee = new Employee(230,"James", "jammy@gmail.com");	`<Employee>`
		`<employeeid>230</employeeid>`
		`<email>`jammy@gmail.com`</email>`
		`<name>James</name>`
		`</Employee>`

2	Message message = new Message("Kim", "Peter", new Date(), "Will meet you during Coffee break");	`<Message>`
		`<sentdate>Apr 01 2013</sentdate>`
		`<msgtext>Will meet you during Coffee break`
		`</msgtext>`
		`<sendername>Kim</sendername>`
		`<receivername>Peter</receivername>`
		`</Message>`

```
/**
 * @param args
 */
public static void main(String[] args) {
        Employee employee = new Employee(230, "James", "jammy@gmail.com");
        String employeeXML = generateXML(employee);
        System.out.println(employeeXML);

        Message message = new Message("Kim", "Peter", new Date(), "Will meet
you during Coffee break");
        String messageXML = generateXML(message);
        System.out.println(messageXML);
}

private static String generateXML(Object object) {
        StringBuilder builder = new StringBuilder();
            // complete the code
        return builder.toString();
}
```
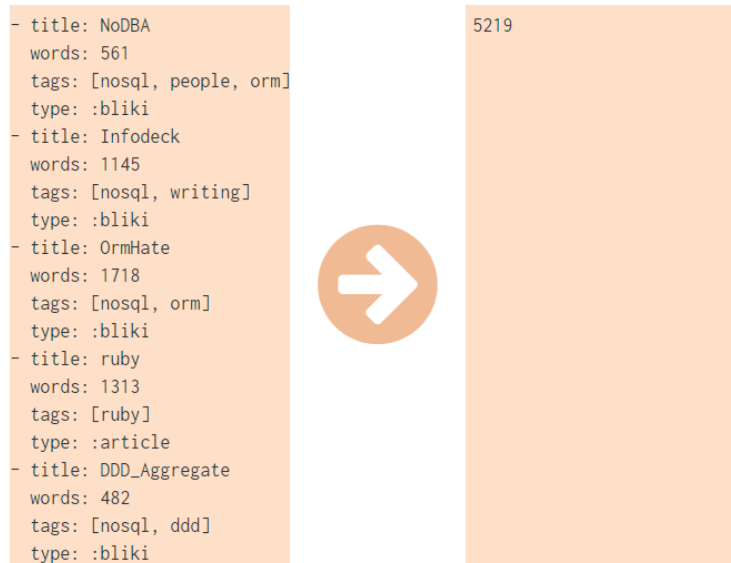
Java Collection Stream based Questions:

4. Create an Article class for the shown class diagram and perform the mentioned operations:

Article

-title: String
-words: int
-type: String
-tags: List<String>
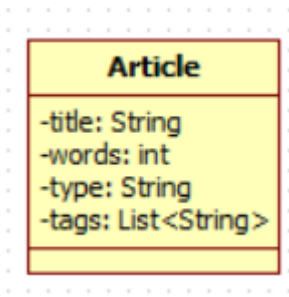
a) Getting total word counts

Create a List of articles and get a total word count for all the articles in the list

```
- title: NoDBA                              5219
  words: 561
  tags: [nosql, people, orm]
  type: :bliki
- title: Infodeck
  words: 1145
  tags: [nosql, writing]
  type: :bliki
- title: OrmHate
  words: 1718
  tags: [nosql, orm]
  type: :bliki
- title: ruby
  words: 1313
  tags: [ruby]
  type: :article
- title: DDD_Aggregate
  words: 482
  tags: [nosql, ddd]
  type: :bliki
```

b) figure out how many articles there are by each type

```
- title: NoDBA                              {bliki: 4, article: 1}
  words: 561
  tags: [nosql, people, orm]
  type: :bliki
- title: Infodeck
  words: 1145
  tags: [nosql, writing]
  type: :bliki
- title: OrmHate
  words: 1718
  tags: [nosql, orm]
  type: :bliki
- title: ruby
  words: 1313
  tags: [ruby]
  type: :article
- title: DDD_Aggregate
  words: 482
  tags: [nosql, ddd]
  type: :bliki
```

5) Radio Jockey:

A **Radio Jockey** (or **RJ**) is a person who hosts a radio talk show where the RJ selects the music to be played, or topic of discussion, by interacting with the audience; the interaction is often via telephone, but may also be online, or via email.

Radio jockeys may start working for fixed salaries, but their pay increases in proportion to their experience and number of **minutes** anchored.

Create Program.java to have the following fields:

[Date, RJ Name, Show name, Start time, End time] as listed below:

Store entries into List<Program>.
Listed below is a sample content:
2013-MAY-2,Prithvi, Good Morning, 7:00 AM, 9:00 AM
2013-MAY-2,Shilpa, Breakfast with Shilpa, 9:00 AM, 10:30 AM
2013-MAY-2, Prithvi, Bangalore Fatafat, 10:45 AM, 1:30 PM
2013-MAY-2, Pavithra, Test your knowledge, 1:35 PM, 3:00 PM
2013-MAY-2, Rakesh, Ulfat Sultan, 3:05 PM, 6:00 PM
2013-MAY-3 ,Shilpa, Good Morning, 7:00 AM, 9:00 AM
2013-MAY-3 ,Shilpa, Breakfast with Shilpa, 9:00 AM, 10:30 AM
2013-MAY-3 ,Pavithra, Bangalore Fatafat, 10:35 AM, 1:40 PM
2013-MAY-3 ,Pavithra, Test your knowledge, 1:45 PM, 3:00 PM
2013-MAY-3 ,Rakesh, Ulfat Sultan, 3:05 PM, 6:00 PM

Use Java 8 stream() apis along with high order functions to complete the assignment.

Input: Date and RJ name
Output: Total Minutes anchored by the RJ on that date

Input	Output
2013-MAY-2 Prithvi	285
2013-MAY-3 Pavithra	260
2013-MAY-3 Shilpa	210
2013-MAY-3 Rakesh	175