OOP
Lab 12.

🎯 Objectives:

- `Collections API`
- Using sets: `HashSet` and `TreeSet`
- Using maps: `HashMap, TreeMap`
- Exception handling: try with resources

Exercise 1

In this exercise you have to make a copy of the second exercise of Lab9 (English dictionary service).

a. You have to add two new dictionary implementations, one using a HashSet (`HashSetDictionary`) and another using a TreeSet (`TreeSetDictionary`) for dictionary storage (see Fig.1).

b. Complete the `DictionaryProvider` class to provide 3 types of dictionaries:

```
public class DictionaryProvider {
   public static IDictionary createDictionary( DictionaryType dtype ){
       IDictionary dictionary = null;
       switch( dtype ){
           case ARRAY_LIST: dictionary =
                            ArrayListDictionary.newInstance(); break;
           case HASH_SET: dictionary =
                            HashSetDictionary.newInstance(); break;
           case TREE_SET: dictionary =
                            TreeSetDictionary.newInstance(); break;
       }
       return dictionary;
   }
}
```

c. `Main class - main` method
Compare the three implementations by searching all the words from the bible (bible.txt). Measure the elapsed time for each method as follows:

```
long startTime = System.nanoTime();
DictionaryService service = new
                         DictionaryService(DictionaryType.ARRAY_LIST);
service.findWordsFile("bible.txt");
long endTime = System.nanoTime();
long timeElapsed = endTime - startTime;
System.out.println("Execution time in milliseconds: " +
                                              timeElapsed / 1000000);
```

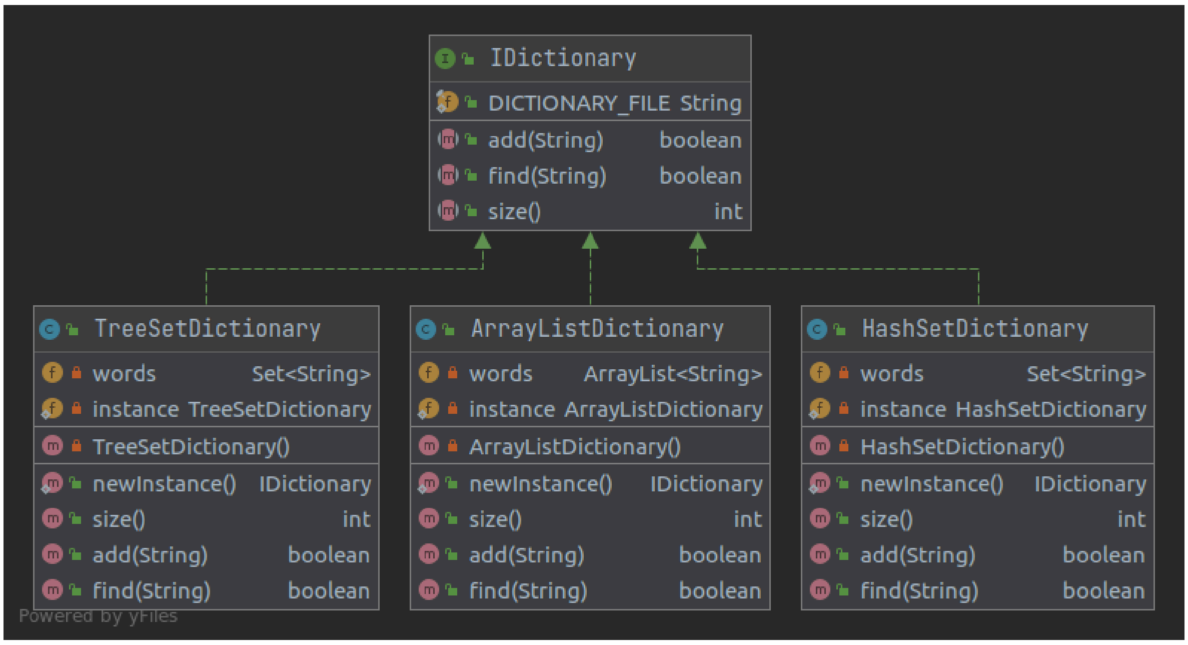

Fig.1 `IDictionary` *interface and its implementations*

Important note:

In the `findWordsFile` method you have to open a file. Use the **try with resource** exception handling in order to release the file resource properly.

Exercise 2

In this exercise, your task is to rewrite exercise 3 from Lab 11 (`Product, Storage`) in order to use `HashMap<Integer, Product>` to store the products. HashMap does not require products to be comparable!

- Compare the two solutions (Lab 10 and this one) for the `data1000000.txt` and `update1000000.txt` files.
- Which solution is better and why?

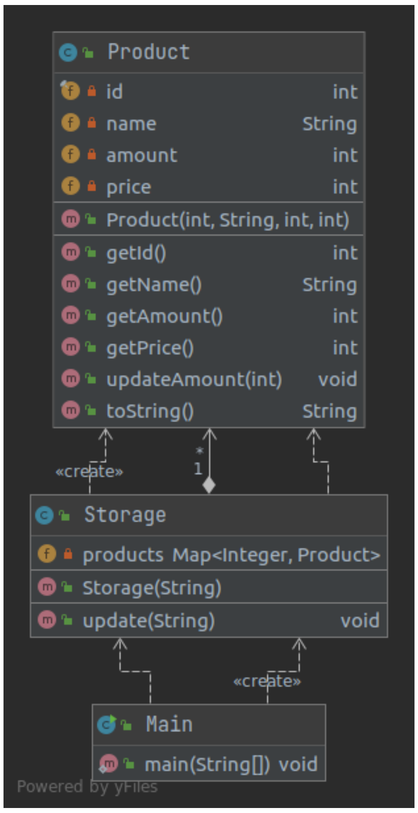

Fig.2 Product and Storage classes

Important note:
Use the **try with resource** exception handling in order to release the file resource properly.

Exercise 3

Baccalaureate (BAC)

You have to simulate a simplified baccalaureate examination. Each enrolled student has a unique ID - a five-digit integer, a first name and a last name (`nevek1.txt`).

The students have to take an examination for each of the following three subjects: Hungarian language, Romanian language and mathematics. The marks for each subject are stored in separate input files (`magyar.txt`, `roman.txt`, `matek.txt`). Each line in these files contains an identifier and a mark.

Design and implement an object-oriented application which:

- Read the data.
- Computes the average mark for each enrolled student. A student passes the exam if and only if the student has passed each subject's exam (subject's mark >= 5) and the three subjects' average is at least 6 (average >= 6).
- Prints the number of students who have successfully passed the BAC exam.
- Prints the list of students in alphabetical order who have failed the bAC exam.

Data can be downloaded from here.
Constraints:

- Use at least 2 classes (Main class not included), for example: `Student` and `Bac`.
- The processing should be optimal, therefore you should use an appropriate collection with efficient searching.

If you worked correctly, 12,400 out of a total of 17,200 students passed the Bac.