Lab #7: Set

The main aim of the lab is to get familiar with some methods in **Set**.

==========

Task 1: Word Count

==========

> This task aim at using **implementations of Set** to write a program for counting words in a text file. Methods serving for such a program are defined in **MyWordCount.java** as follows:

```
public class MyWordCount {
// public static final String fileName = "data/hamlet.txt";
	public static final String fileName = "data/fit.txt";

	private List<String> words = new ArrayList<>();

	public MyWordCount() {
		try {
			this.words.addAll(Utils.loadWord(fileName));
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
	}
// Returns a set of WordCount objects that represents the number of times
each //unique token appears in the file
	// data/hamlet.txt (or fit.txt)
	// In this method, we do not consider the order of tokens.
	public List<WordCount> getWordCounts() {
		// TODO
		return null;
	}

	// Returns the words that their appearances are 1, do not consider
duplidated words

	public Set<String> getUniqueWords() {
		// TODO
		return null;
	}
// Returns the words in the text file, duplicated words appear once in the
result
	public Set<String> getDistinctWords() {
		// TODO
		return null;
	}

// Prints out the number of times each unique token appears in the file
	// data/hamlet.txt (or fit.txt) according to ascending order of
tokens
	// Example: An - 3, Bug - 10, ...
	public Set<WordCount> exportWordCounts() {
		// TODO
		return null;
```

```
	}

// Prints out the number of times each unique token appears in the file
// data/hamlet.txt (or fit.txt) according descending order of occurences
	// Example: Bug - 10, An - 3, Nam - 2.
	public Set<WordCount> exportWordCountsOrderByOccurence() {
		// TODO
		return null;
	}

// delete words begining with the given pattern (i.e., delete words begin
// with 'A' letter
	public Set<WordCount> filterWords(String pattern) {
		// TODO
		return null;
	}

}
```

Consider to use ***appropriate implementations of Set*** to develop above methods.

==========

Task 2: Collections and Predicate

==========

In this task, we try to use the predefined interface **Predicate** in java.util.function package to implement methods in the **MyPredicates.java** class. Aside from that, the main point of the task is to work with predicates and to see how they are useful in generic programming.

The Predicate interface has the following form:

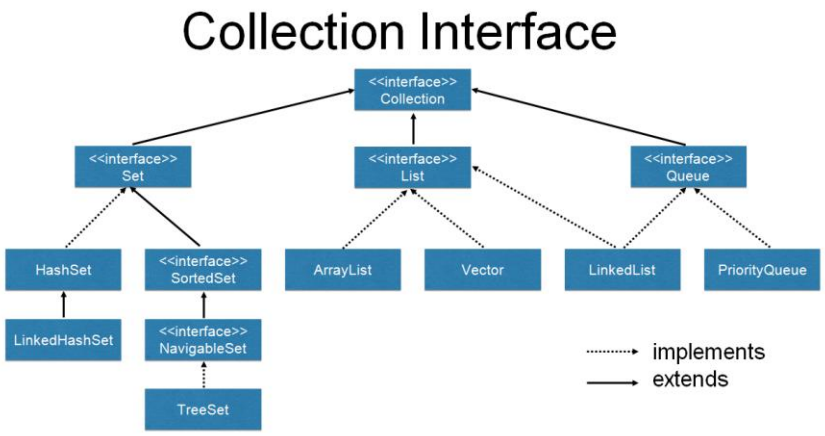
```
public interface Predicate<T> {
  public boolean test(T obj);
}
```

```
// Remove every object, obj, from coll for which p.test(obj)
	// is true. (This does the same thing as
coll.removeIf(p).)
	public static <T> void remove(Collection<T> coll,
Predicate<T> p) {
		// TODO
	}
	// Remove every object, obj, from coll for which
	// pr.test(obj) is false. (That is, retain the
	// objects for which the predicate is true.)
	public static <T> void retain(Collection<T> coll,
Predicate<T> p) {
		// TODO
	}
```

```
	// Return a Set that contains all unique objects, obj,
	// from the collection, coll, such that p.test(obj)
	// is true.
	public static <T> Set<T> collect(Collection<T> coll,
Predicate<T> p) {
		// TODO
		return null;
	}
	// Return the index of the first item in list
	// for which the predicate is true, if any.
	// If there is no such item, return -1.
	public static <T> int find(Collection<T> coll,
Predicate<T> p) {
		// TODO
		return -1;

	}
```

Notice that, the class diagram of classes and interfaces in Java Collection Framework is described in the following figure:



In this task, to test the implemented methods the parameter **coll** could be implementations of Set and List interfaces such as **HashSet**, **LinkedHashSet**, **TreeSet**, and **ArrayList**, **LinkedList**.

For simplicity, the objects of **coll** could be integers. Then, we can expand the problem by using coll as a collection of Student objects (Student class was defined in the previous Lab).

Here is an example of **EvenPredicate.java** for dealing with even integers.

```
public class Even implements Predicate<Integer> {
	public boolean test(Integer i) {
		return (i % 2 == 0);
	}
}
```