Generics

Programming Fundamentals 2

Goals

- ✯ Implementation of an interface.
- ✯ Implementation of a class with a generic type.
- ✯ Sorting a list of objects.
- ✯ **Relevant videos**:
  - Parametric polymorphism.
  - Casting polymorphism in-depth.
  - (Almost) everything is object.

Deliverables

1. The code on your Github repository generated by clicking here: `https://classroom.github.com/a/EfBiKd9F`
2. **Reviewer**: Pierre Talbot (ptal on Github).
3. **Automated testing** using JUnit and unit testing.

Exercise 1 – Listing everything

After successfully implementing a dynamic array and a linked list, you noticed a user manipulates both structures using similar methods. Indeed, both dynamic arrays and linked lists represent the same abstract concept of a *list data structure*, *i.e.*, a collection of elements. The methods you have implemented so far are methods manipulating a list, regardless of its kind.

1. We provide an interface `List` in the file `List.java` abstracting the concept of a list. Add to this interface every method (as abstract method) relevant to list manipulation: it corresponds to the methods you have implemented in classes `DynamicArray` and `LinkedList`. In the definition of the interface `List`, the type of the elements in the collection are denoted by `T`. Thus, be careful to make abstract methods depend on `T` whenever it is needed.
2. In the files `DynamicArray.java` and `LinkedList.java`, we provide two classes which implement the interface `List`. Implement every abstract method in both files you have defined in `List`. Actually, the classes `DynamicArray` and `LinkedList` should be very similar to their implementation in previous laboratories. The only difference is that here, elements contained in the list are not integers but of type `T` (and everything induced by that).
3. Write your own unit tests in order to test different instantiations of `List`. Again, you can adapt the tests of the previous laboratories so they generalize to `List`.

Exercise 2 – Sorting video games

Now, let's try our new abstract list on a collection of video games! You can rate a video game by creating an object of type `VideoGame` (class provided in the project) according to different characteristics: atmosphere, music, artistic direction, graphism, scenario and gameplay. Each of those characteristics must be rated with a score from 0 to 100 (the greater the better). For instance, if I want to rate the game "Pokémon Legends: Arceus", I can write the following:
`VideoGame pokemon = new VideoGame("Pokémon Legends: Arceus", 95, 80, 75, 10, 15, 90);`.

1. In order to sort a list of video games, you need a way to compare them! In Java, an interface `Comparable<T>` exists[1] and is used to compare an object with another object of type `T`. This interface has only one abstract method `public abstract int compareTo(T other)` which returns a negative integer if the object `this` (the object calling the method) is considered lower than the object `other`, and positive if it is considered greater. Make video games comparable by making the class `VideoGame` implements the interface `Comparable<VideoGame>`. There is not only one way to compare video games, you can choose the way which suits your preference, *e.g.*, if you give more importance to the atmosphere.
It is strongly recommended (though not required) that natural orderings be consistent with equals. For example, `obj1.compareTo(obj2) == 0` must be equivalent to `obj1.equals(obj2)`. In the example of video games, we can imagine that two video games are equal if and only if their statistics are identical. The method `compareTo` must then return `0` in that condition only.

2. In the file `Algorithm.java`, implement a method for sorting a list of any object. In the definition of the method, `T` represents the type of objects contained in the list given as an input. The only constraint of `T` is that it must be a comparable type, hence the presence of `<T extends Comparable<T>>` in the method signature.

3. Add a main function (in a new file `Generics.java`), try and test your algorithm by creating a list of video games and sorting it. Use the same function to sort a list of `Integer`.

[1] `https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`