Report P5

Daniel Varela Sánchez & Pablo Sancho Saiz

Section 1:

In the first section we were asked about creating a new list type called **SortedList** which has to keep the elements sorted always. Since it has to be compatible with the list interface, it needs to have the elements sorted and it has to be generic, it has to extend ArrayList<T>.

By declaring this class like “SortedList<T extends Comparable<T>>” we can store objects that implement the comparable of their object type.

Also we stored the different comparators so that if there’s equality we’ve got more filters to detect differences between objects. With the addCriterion we store the new comparators and sort our list of objects with them.

And also the **Person** class so that the first test works properly and gets all the information necessary which is its name and birthdate.

Section 2:

Now we had to create the **Template** class which is parameterisable with a type. We implemented the addObjects, emit and withSortingCriteria methods and since it has an empty constructor it is built by calling the add method as the exercise states.

We stored all that new information in a list of objects of class **Letter**, which is responsible to fill the text gaps,so that for the upcoming exercises our code was as generic as possible.

Section 3:

In this section we created the **Mascot** class and the addWhen and addForEach methods that create a **When** and a **Each** object respectively. We did that since the functionality is very similar to the Letter class with the emit method and with the function the lambda expression covers in the emit method.

Section 4:

Finally we created the **TimeStamper**, **UpperCaser** and **FilePersister** classes with their different behaviors using the Strategy design pattern, so all of them implement the **IOption** functional interface that has one method which has to be implemented by all of these options in their respective classes.

Class Diagram:

