Brief design note

# What is the purpose of using the Comparable and Document interface in the program? Is there another way of designing the same program functionality (a) without using Comparable and (b) without using Document? Briefly explain why or why not under each case?

**Answer:**

* Using Comparable and Document interface in the program could make it easier for ordering. In the other mean, list of objects that implement this interface can be sorted automatically by sort method of the interface.
* We’re able to use **getName()** method, which returns a **String** and use **compareTo()** method for this String. But some unnecessary exposing data risks might happen and designing a type should have compare methods

# Draw a UML design class diagram of the type hierarchy used in the program? What is an advantage of designing types in this hierarchy?

**Answer:**

* UML design class diagram of the type hierarchy used in the program:
* Advantage of designing types in this hierarchy:
  + **Modifiability**: It is easy to define new type based on the original one, new implementations of new type can be added to a program without affecting the using code.
  + **Reusability**: Subtypes either have the same behavior as their super type or extend the behavior with new operations. The type hierarchy that have the former condition are said to support abstraction by specification. What this means is that the super type’s specification holds true for all of its subtypes.
  + **Interface**: The main reason interfaces are good is that you can procedure many completely different implementations of an interface, using interfaces therefore reducing coupling and increasing flexibility. This benefit actually does not show in this assignment but it is necessary for future development.

# What is the purpose of using the SortedSet class in the program? Is it possible to develop the same program functionality without using this class? Briefly explain why or why not?

**Answer:**

* Using the SortedSet class in the program could provide a total ordering on its elements. The elements are ordered using their natural ordering (**Comparable**). The set's iterator will traverse the set in ascending element order. Several additional operations are provided to take advantage of the ordering.
* In this assignment we used **Comparable**<Customer>, we also could develop by implementing **Comparator**<Customer> in a separate class.
* In general, it's a good idea to implement Comparable<T> if there's a single "natural" sort order... otherwise (if you happen to want to sort in a particular order, but might equally easily want a different one) it's better to implement Comparator<T>. This particular situation could go either way, to be honest... but I'd probably stick with the more flexible Comparator<T> option.

# The original KEngine library can only search for text documents using keywords. What makes it possible to use this component in your program to search for Customer objects using keywords?

**Answer:**

* The original KEngine library can only search for text documents using keywords, but with **toHTMLdoc()** method which we add before . This method puts data of each Customer object into Html tags and returns a string that can be formed to a new Object data which can search for Customer object using keyword.

# Draw a complete design class diagram of the application showing the classes and their dependencies

1. **From the design class diagram, identify the implementation strategy that was used to build the application. Briefly discuss this strategy.**