Brief Discussion Regarding Home Work:

In this homework, I have worked on implementing a generic comparator which will help to sort the given list of homes based on sort field asked in either ascending or descending order. I have used Comparator Interface implementation and Reflect class to dynamically calling the getter method of Home Class.

Approach (Generic Comparator)-(GenericComparator.java)

1. Generate the method name (getter Method) by using a prefix: “get”.
2. Once name is generated Use check for null Objects in comparing and if not null get the sortField Values to compare (by using Java Reflect) and compare them to get the actual position in the list. (Added Support for SortField Datatype of: Integer, String, Double, Float, Long, Date)
3. Return the list of Homes.

As given in the home work the input parameter for the sort method is Iterable, I have also implemented Iterable interface to generate Iterable object but anyway converted it to List to give it as input to Comparator. (Homes.java)

Home Class: Defined class structure of Java(Home.java).

Steps to Execute:

1. Git Clone repository from:
2. Run SortHomes.java

I have Defined a list of Homes and Result is displayed on the console.

Pros and Cons:

1. Generic Comparator reduces implementation of Multiple Comparators if sorting field are dynamic as in our case.
2. Code is manageable as even if new field is added, that would not need much of code change in comparator unless field data is not added in the comparator.
3. Using reflection helps to dynamically call the class method using class object which is useful when you don’t know which method has to be called beforehand. I thought this fits our requirement here.
4. On the other hand, reflection has some performance issues as it resolves dynamically and hence, optimization of JVM is cannot be done.