Lab 7

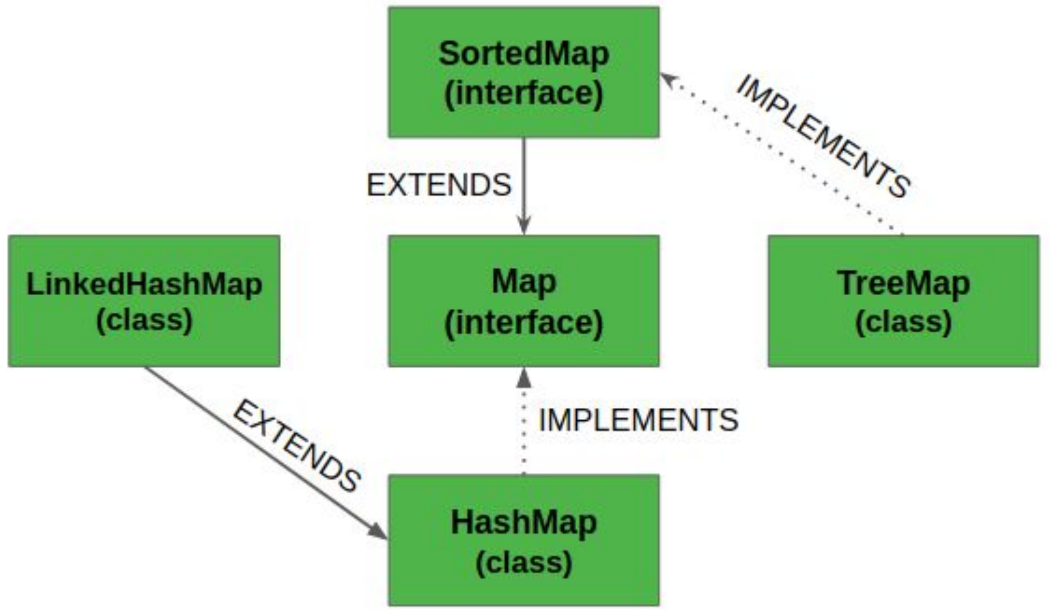
READING CSV FILES

Map

- java.util.Map is an **interface**

  Eg. **Map**<String, String> **map** = new **HashMap**<>();

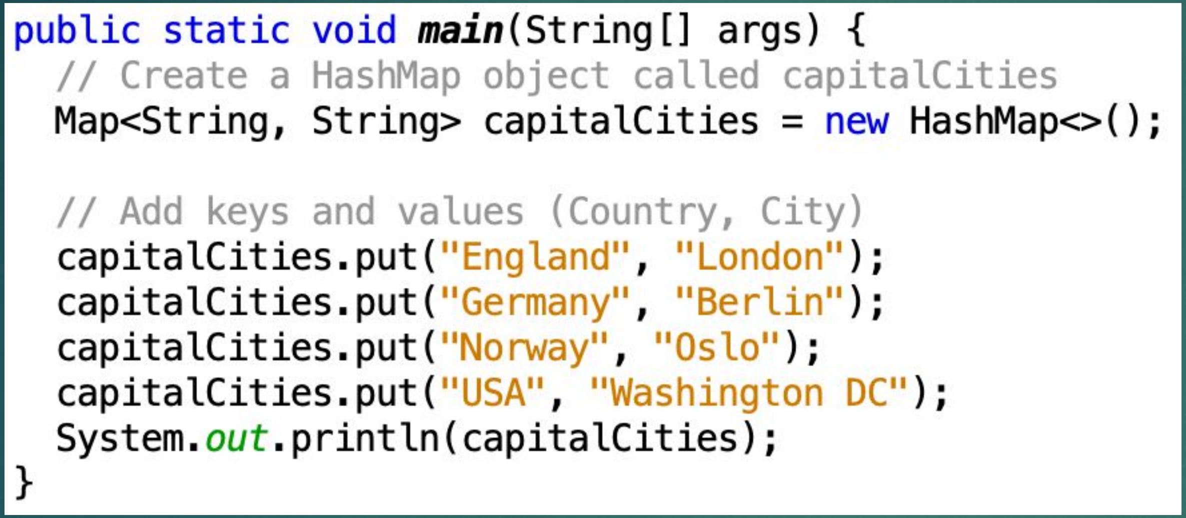
- A Map **cannot** contain **duplicate** keys and each key can map to at most one value.
- Some implementations allow null key and null value like the HashMap and LinkedHashMap, but some do not like the TreeMap.



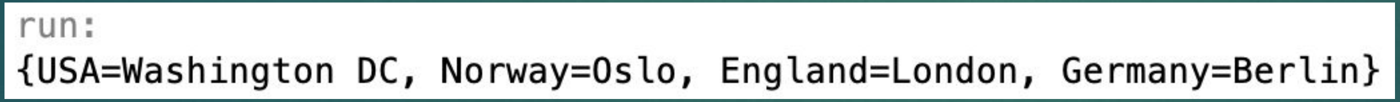
MAP Hierarchy in Java

HashMap

- A HashMap stores items in "**key/value**" pairs.



```java
public static void main(String[] args) {
  // Create a HashMap object called capitalCities
  Map<String, String> capitalCities = new HashMap<>();

  // Add keys and values (Country, City)
  capitalCities.put("England", "London");
  capitalCities.put("Germany", "Berlin");
  capitalCities.put("Norway", "Oslo");
  capitalCities.put("USA", "Washington DC");
  System.out.println(capitalCities);
}
```

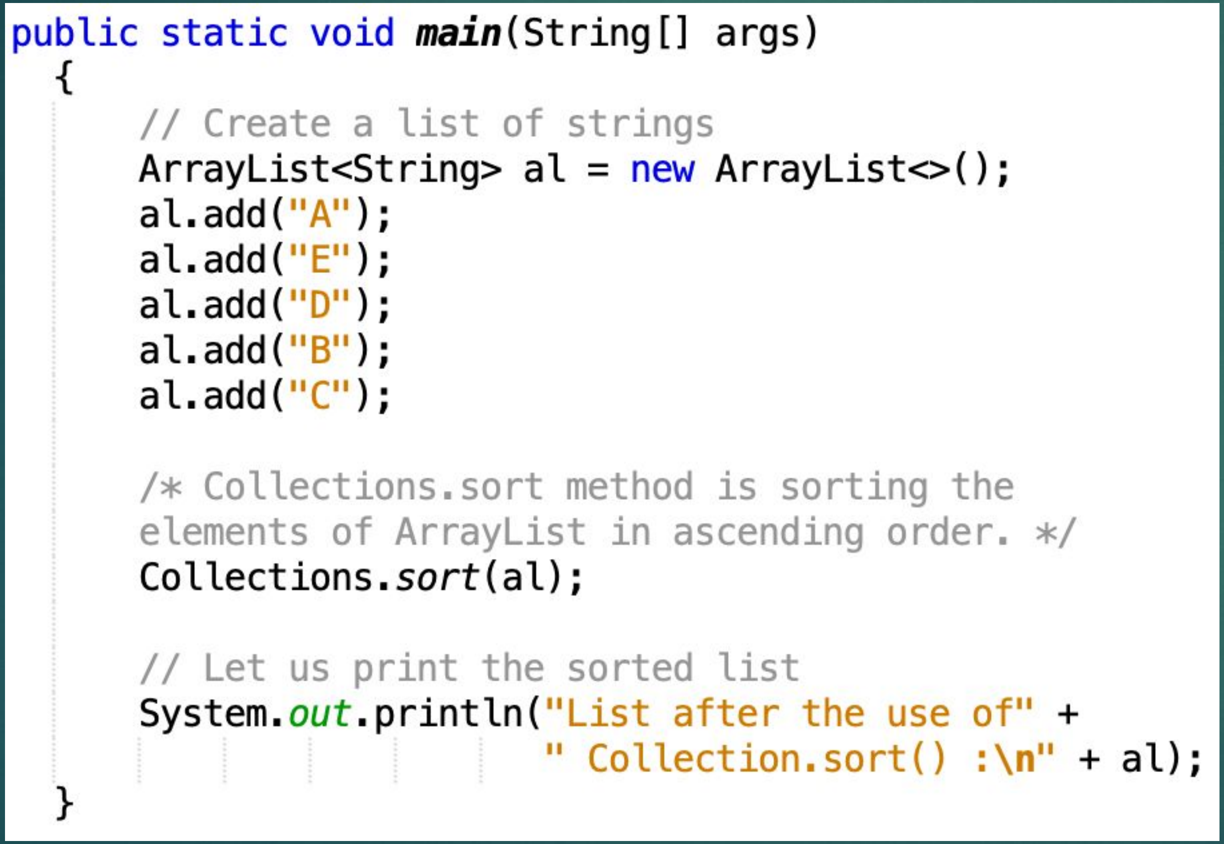


```
run:
{USA=Washington DC, Norway=Oslo, England=London, Germany=Berlin}
```

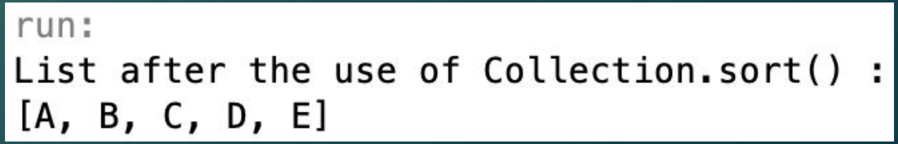
- https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

Collections.sort()

- Collections. sort(List<T> list)



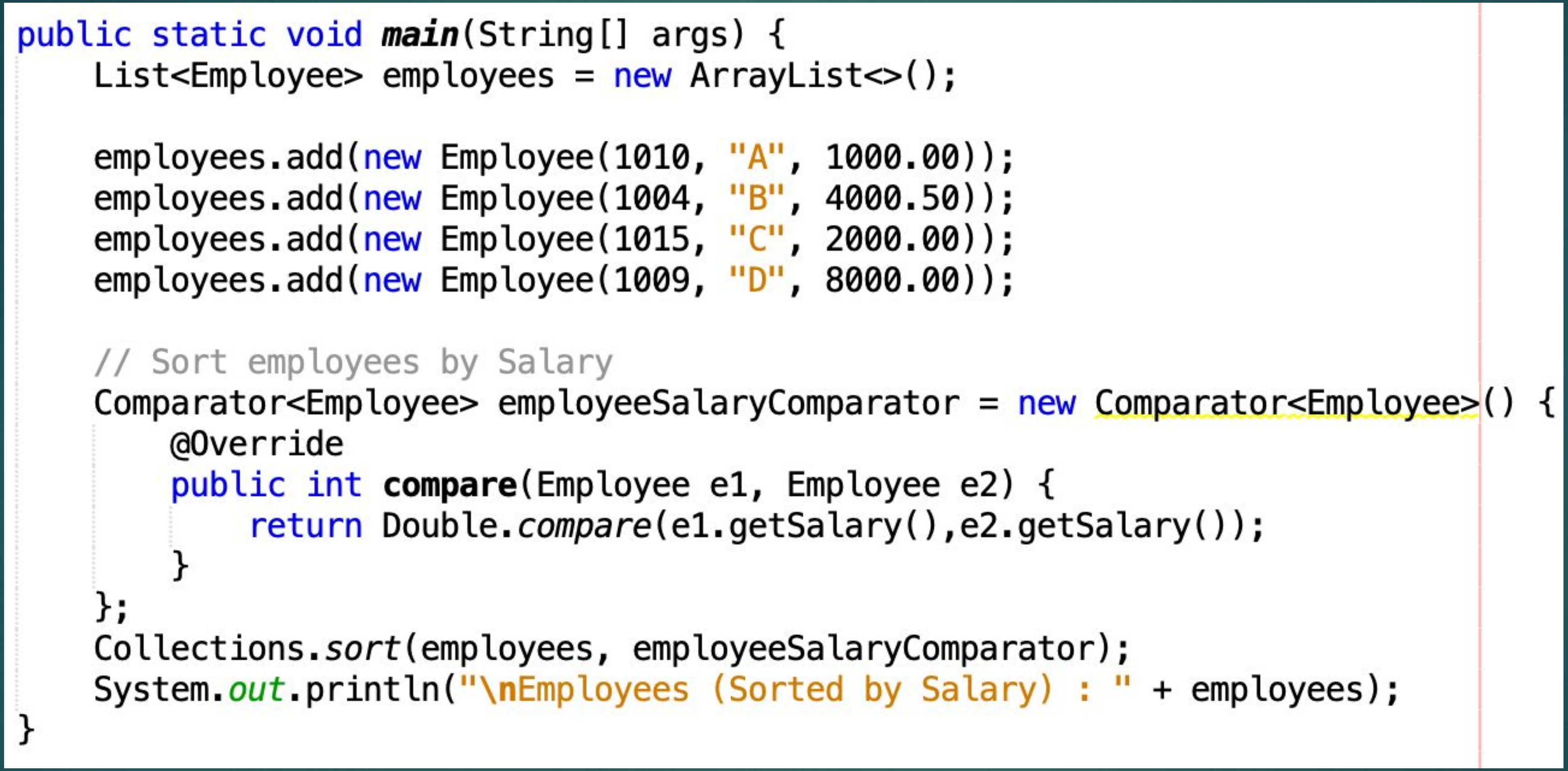
```java
public static void main(String[] args)
  {
      // Create a list of strings
      ArrayList<String> al = new ArrayList<>();
      al.add("A");
      al.add("E");
      al.add("D");
      al.add("B");
      al.add("C");

      /* Collections.sort method is sorting the
      elements of ArrayList in ascending order. */
      Collections.sort(al);

      // Let us print the sorted list
      System.out.println("List after the use of" +
                           " Collection.sort() :\n" + al);
  }
```



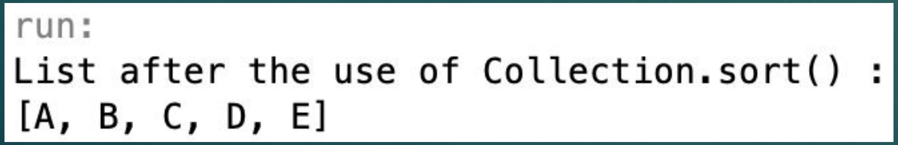
```
run:
List after the use of Collection.sort() :
[A, B, C, D, E]
```

Collections.sort()

- Collections. sort(List<T> list, Comparator<? super T> c)



```java
public static void main(String[] args) {
    List<Employee> employees = new ArrayList<>();

    employees.add(new Employee(1010, "A", 1000.00));
    employees.add(new Employee(1004, "B", 4000.50));
    employees.add(new Employee(1015, "C", 2000.00));
    employees.add(new Employee(1009, "D", 8000.00));

    // Sort employees by Salary
    Comparator<Employee> employeeSalaryComparator = new Comparator<Employee>() {
        @Override
        public int compare(Employee e1, Employee e2) {
            return Double.compare(e1.getSalary(),e2.getSalary());
        }
    };
    Collections.sort(employees, employeeSalaryComparator);
    System.out.println("\nEmployees (Sorted by Salary) : " + employees);
}
```



```
run:
List after the use of Collection.sort() :
[A, B, C, D, E]
```

Homework

- This is an individual assignment.
- Multiple git commits is required
- No direct commit on master branch

1). Find Average number of likes per comment.
2). Find the post with most liked comments.
3). Find the post with most comments.
4). Top 5 inactive users based on total posts number.
5). Top 5 inactive users based on total comments they created.
6). Top 5 inactive users overall (sum of comments, posts and likes)
7). Top 5 proactive users overall (sum of comments, posts and likes)

Due Date: Sat, Oct 31st, at 11:59 pm., on Github