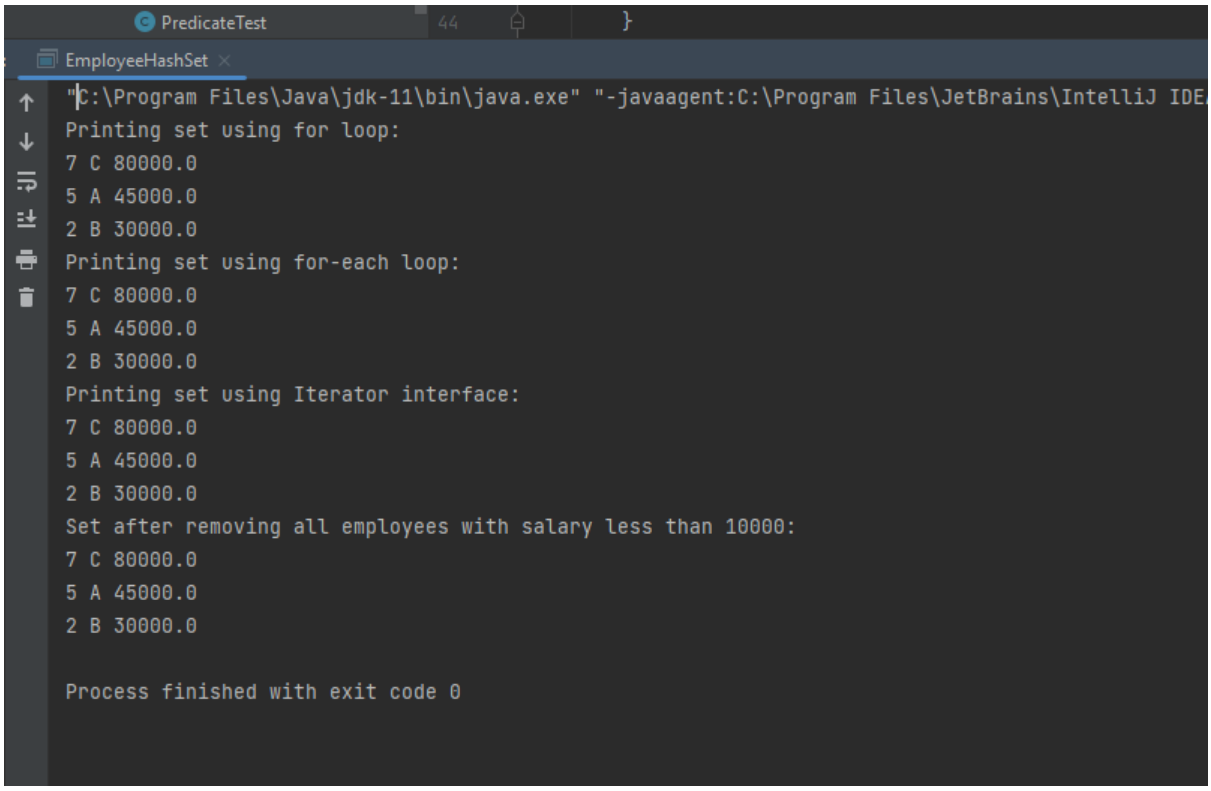
ASSIGMENT JAVA DAY13

Harshit Kushmakar| 16896

1. **Change the program created in Question – 1 & 2 of Collection Day-1 assignment to use HashSet instead of List. Record your findings and resolve issue if any.**

```
2. package assignment13;

   import assignment12.Employee;

   import java.util.HashSet;
   import java.util.Iterator;
   import java.util.Set;

   public class EmployeeHashSet {
       public static void main(String[] args) {
           Set<Employee> empSet = new HashSet<Employee>();
           empSet.add(new Employee(5,"A",45000));
           empSet.add(new Employee(2,"B",30000));
           empSet.add(new Employee(7,"C",80000));



           // Print the set using for loop
           System.out.println("Printing set using for loop:");
           for (Employee emp : empSet) {
               System.out.println(emp.getEmpID() + " " +
   emp.getEmpName() + " " + emp.getSalary());
           }

           // Print the set using for-each loop
           System.out.println("Printing set using for-each loop:");
           empSet.forEach(emp -> System.out.println(emp.getEmpID() + " "
   + emp.getEmpName() + " " + emp.getSalary()));

           // Print the set using Iterator interface
           System.out.println("Printing set using Iterator interface:");
           Iterator<Employee> empIterator = empSet.iterator();
           while (empIterator.hasNext()) {
               Employee emp = empIterator.next();
               System.out.println(emp.getEmpID() + " " +
   emp.getEmpName() + " " + emp.getSalary());
           }


           // Remove all A from the set who have salary less than 10000
           Iterator<Employee> empIterator2 = empSet.iterator();
           while (empIterator2.hasNext()) {
               Employee emp = empIterator2.next();
               if (emp.getSalary() < 10000) {
                   empIterator2.remove();
```

```
                }
            }
            System.out.println("Set after removing all employees with
    salary less than 10000:");
            empSet.forEach(emp -> System.out.println(emp.getEmpID() + " "
    + emp.getEmpName() + " " + emp.getSalary()));
        }
    }
```



2. Change the same program to use TreeSet. Sort the TreeSet on basis of

a. Employee id using Comparable Interface.

b. Employee Name using Comparator Interface.

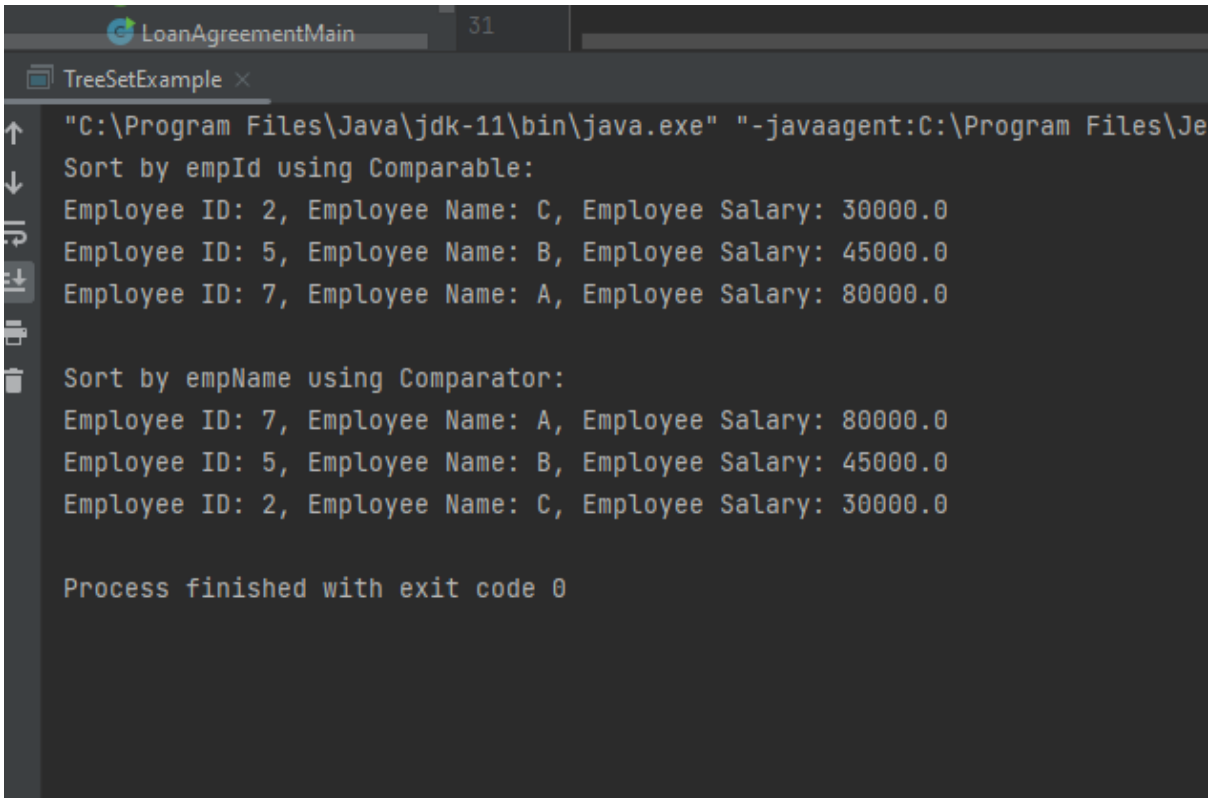
```
package assignment13;

public class Employee implements Comparable<Employee>{
    private int empId;
    private String empName;
    private double salary;

    public Employee(int empId, String empName, double salary) {
        this.empId = empId;
        this.empName = empName;
        this.salary = salary;
    }

    public int getEmpId() {
        return empId;
    }
```

```
    public String getEmpName() {
        return empName;
    }

    public double getSalary() {
        return salary;
    }

    @Override
    public int compareTo(Employee e) {
        return this.empId - e.empId;
    }
}
```

```
package assignment13;

import java.util.Comparator;
import java.util.Set;
import java.util.TreeSet;

public class EmployeeNameComparator  implements Comparator<Employee> {
    @Override
    public int compare(Employee e1, Employee e2) {
        return e1.getEmpName().compareTo(e2.getEmpName());
    }
}
```

```
package assignment13;

import java.util.Set;
import java.util.TreeSet;

public class TreeSetExample {
    public static void main(String[] args) {
        Set<Employee> employeeSet = new TreeSet<>();

        employeeSet.add(new Employee(5, "B", 45000));
        employeeSet.add(new Employee(2, "C", 30000));
        employeeSet.add(new Employee(7, "A", 80000));

        // Sort by empId using Comparable
        System.out.println("Sort by empId using Comparable:");
        for(Employee e : employeeSet) {
            System.out.println("Employee ID: " + e.getEmpId() + ", Employee
Name: " + e.getEmpName() + ", Employee Salary: " + e.getSalary());
        }

        // Sort by empName using Comparator
        Set<Employee> employeeSetByName = new TreeSet<>(new
EmployeeNameComparator());
        employeeSetByName.addAll(employeeSet);

        System.out.println("\nSort by empName using Comparator:");
        for(Employee e : employeeSetByName) {
            System.out.println("Employee ID: " + e.getEmpId() + ", Employee
```

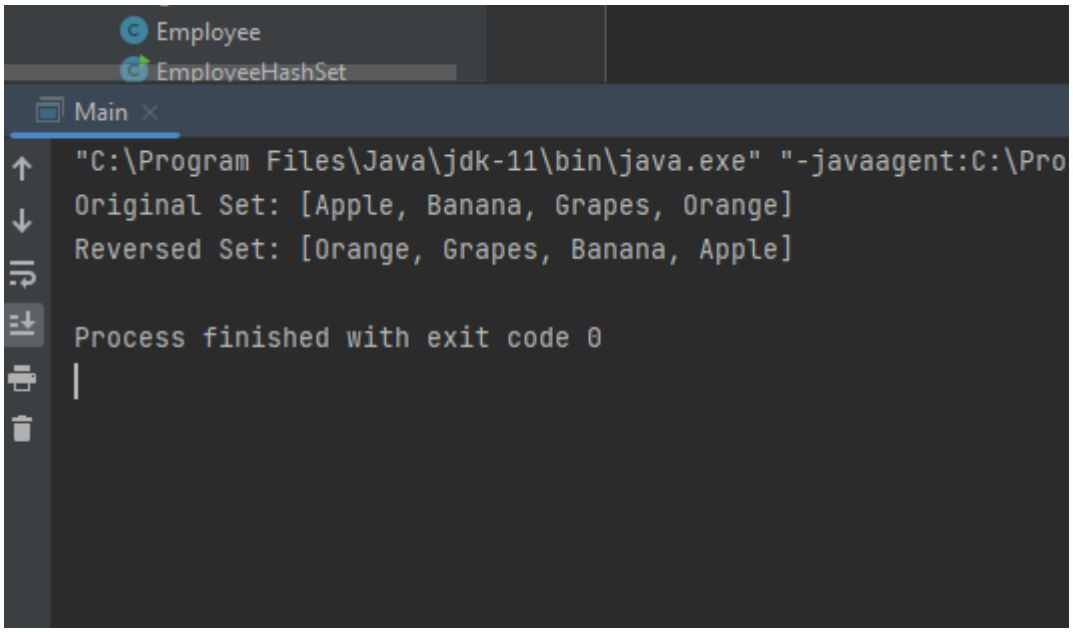
```
Name: " + e.getEmpName() + ", Employee Salary: " + e.getSalary());
        }
    }
}
```



3. In the above TreeSet, write a Java program to create a reverse order view of the elements.

```
4. package assignment13;

   import java.util.Set;
   import java.util.TreeSet;

   public class Main {

           public static void main(String[] args) {
               TreeSet<String> set = new TreeSet<>();
               set.add("Apple");
               set.add("Banana");
               set.add("Orange");
               set.add("Grapes");
```

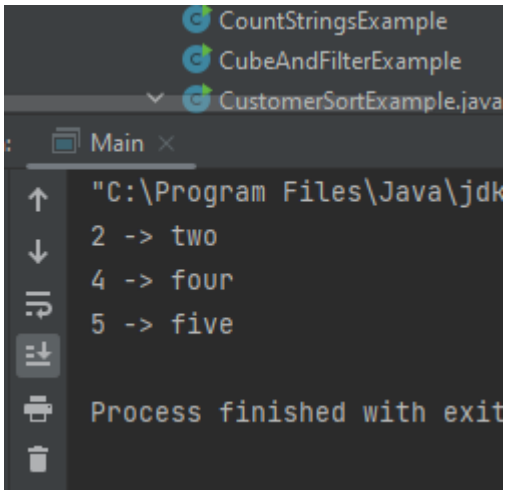
```
            System.out.println("Original Set: " + set);

            Set<String> reversedSet = set.descendingSet();
            System.out.println("Reversed Set: " + reversedSet);
        }
    }
```



4. Make a hash table (Map) that maps numbers (e.g., 2) to words (e.g, “two” or “dos”). Test it out by passing it a few numbers and printing out the corresponding words. Note: hash table keys in Java cannot be primitives; they must be objects

```
public class Main {
    public static void main(String[] args) {

        Map<Integer, String> numberWords = new HashMap<>();

        // Add some number-word pairs to the map
        numberWords.put(1, "one");
        numberWords.put(2, "two");
        numberWords.put(3, "three");
        numberWords.put(4, "four");
        numberWords.put(5, "five");


        int[] numbers = {2, 4, 5};
        for (int num : numbers) {
            String word = numberWords.get(num);
            System.out.println(num + " -> " + word);
        }
```

```
    }
}
```


CountStringsExample
CubeAndFilterExample
CustomerSortExample.java
Main
"C:\Program Files\Java\jdk
2 -> two
4 -> four
5 -> five
Process finished with exit