1. Write Java code to define List. Insert 5 floating point numbers in List, and using an iterator, find the sum of the numbers in List.

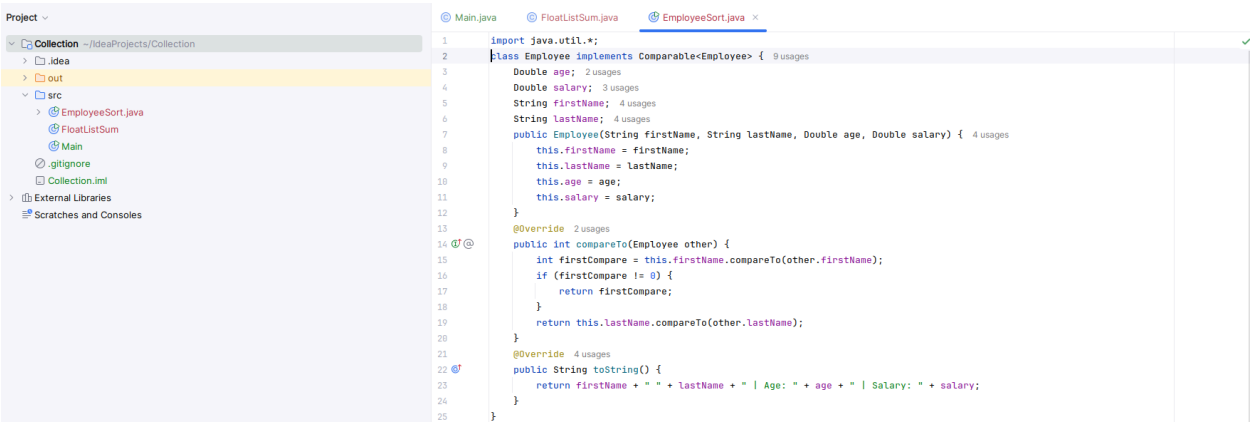
Main.java FloatListSum.java

```java
import java.util.*;

public class FloatListSum {
    public static void main(String[] args) {
        List<Float> numbers = new ArrayList<>();
        numbers.add(10.5f);
        numbers.add(20.3f);
        numbers.add(5.7f);
        numbers.add(3.9f);
        numbers.add(8.6f);

        Iterator<Float> iterator = numbers.iterator();
        float sum = 0;

        while (iterator.hasNext()) {
            sum += iterator.next();
        }

        System.out.println("The sum is: " + sum);
    }
}
```

FloatListSum

```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=32769 -Dfile.encoding=UTF-8 -classpath /home/aradhana/IdeaProjects/Collection/out/production/Collection FloatListSum
The sum is: 49.0

Process finished with exit code 0
```

2. Given the following class Employee class{ Double Age; Double Salary; String Name} Design the class in such a way that the default sorting should work on firstname and lastname. Also, Write a program to sort Employee objects based on salary using Comparator




```java
public class EmployeeSort {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee( firstName: "A",  lastName: "B",  age: 30.0,  salary: 50000.0));
        employees.add(new Employee( firstName: "c",  lastName: "D",  age: 28.0,  salary: 60000.0));
        employees.add(new Employee( firstName: "E",  lastName: "F",  age: 35.0,  salary: 55000.0));
        employees.add(new Employee( firstName: "I",  lastName: "J",  age: 32.0,  salary: 47000.0));
        Collections.sort(employees);
        System.out.println("Sorted by Name:");
        for (Employee e : employees) {
            System.out.println(e);
        }
        employees.sort(Comparator.comparingDouble( Employee emp -> emp.salary));
        System.out.println("\nSorted by Salary:");
```

Output

```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=41599 -Dfile.encoding=UTF-8 -classpath /home/aradhana/IdeaProjects/Collection/out/production/Collection EmployeeSort
Sorted by Name:
A B | Age: 30.0 | Salary: 50000.0
E F | Age: 35.0 | Salary: 55000.0
I J | Age: 32.0 | Salary: 47000.0
c D | Age: 28.0 | Salary: 60000.0

Sorted by Salary:
I J | Age: 32.0 | Salary: 47000.0
A B | Age: 30.0 | Salary: 50000.0
E F | Age: 35.0 | Salary: 55000.0
c D | Age: 28.0 | Salary: 60000.0

Process finished with exit code 0
```

3. Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. (Expected complexity O(1))

```java
import java.util.Stack;
class SpecialStack {
    Stack<Integer> stack = new Stack<>();
    Stack<Integer> minStack = new Stack<>();
    public void push(int x) {
        stack.push(x);
        if (minStack.isEmpty() || x <= minStack.peek()) {
            minStack.push(x);
        } else {
            minStack.push(minStack.peek());
        }
    }
    public int pop() {
        if (stack.isEmpty()) {
            throw new RuntimeException("Stack is empty");
        }
        minStack.pop();
        return stack.pop();
    }
    public boolean isEmpty() {
        return stack.isEmpty();
    }
    public boolean isFull() {
        return false;
    }
    public int getMin() {
        if (minStack.isEmpty()) {
            throw new RuntimeException("Stack is empty");
        }
        return minStack.peek();
    }
}
public class SpecialStackDemo{
    public static void main(String[] args) {
        SpecialStack s = new SpecialStack();

        s.push( x: 10);
        s.push( x: 20);
        s.push( x: 5);
        s.push( x: 8);
```

```java
public class SpecialStackDemo{
    public static void main(String[] args) {
        SpecialStack s = new SpecialStack();

        s.push( x: 10);
        s.push( x: 20);
        s.push( x: 5);
        s.push( x: 8);

        System.out.println("Current Minimum: " + s.getMin());

        s.pop();
        s.pop();

        System.out.println("Current Minimum after popping: " + s.getMin());
    }
}
```

Output

```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=39161 -Dfile.encoding=UTF-8 -classpath /home/aradhana/IdeaProjects/Collection/out/production/Collection SpecialStackDemo
Current Minimum: 5
Current Minimum after popping: 10

Process finished with exit code 0
```

4. Create class Employee with attributes name,age,designation and use instances of these class as keys in a Map and their salary as value

```
Collection ~/IdeaProjects/Collection
.idea
out
src
EmployeeMapExample.java
EmployeeSort.java
FloatListSum
Main
SpecialStackDemo.java
.gitignore
Collection.iml
External Libraries
Scratches and Consoles
```

```java
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

class Employe{  11 usages  new *
    String name;  5 usages
    int age;  5 usages
    String designation;  5 usages

    Employe (String name, int age, String designation) {  3 usages  new *
        this.name = name;
        this.age = age;
        this.designation = designation;
    }

    public boolean equals(Object o) {  no usages  new *
        if (this == o) return true;
        if (!(o instanceof Employe)) return false;
        Employe e = (Employe) o;
        return age == e.age &&
                Objects.equals(name, e.name) &&
                Objects.equals(designation, e.designation);
    }

    public int hashCode() {  no usages  new *
        return Objects.hash(name, age, designation);
    }

    public String toString() {  9 usages  new *
        return name + " | " + age + " | " + designation;
    }
}
```

```java
public class EmployeeMapExample {  new *
    public static void main(String[] args) {  new *
        Map<Employe, Double> employeeMap = new HashMap<>();

        Employe e1 = new Employe( name: "Aradhana",  age: 25,  designation: "Developer");
        Employe e2 = new Employe( name: "Aaru",  age: 26,  designation: "Manager");
        Employe e3 = new Employe( name: "Mishra",  age: 24,  designation: "Tester");

        employeeMap.put(e1,  v: 50000.0);
        employeeMap.put(e2,  v: 60000.0);
        employeeMap.put(e3,  v: 40000.0);

        for (Map.Entry<Employe, Double> entry : employeeMap.entrySet()) {
            System.out.println(entry.getKey() + " → Salary: " + entry.getValue());
        }
    }
}
```

EmployeeMapExample

on > src > EmployeeMapExample.java 2:1 LF UTF-8 4 spaces

Output

```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=38549 -Dfile.encoding=UTF-8 -classpath /home/aradhana/IdeaProjects/Collection/out/production/Collection EmployeeMapExample
Aradhana | 25 | Developer → Salary: 50000.0
Mishra | 24 | Tester → Salary: 40000.0
Aaru | 26 | Manager → Salary: 60000.0

Process finished with exit code 0
```

on > src > EmployeeMapExample.java 2:1 LF UTF-8 4 spaces