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.

import java.util.\*;

public class ListSumExample {

public static void main(String[] args) {

List<Float> numbers = new ArrayList<Float>();

numbers.add(2.5F);

numbers.add(4.8f);

numbers.add(1.2f);

numbers.add(3.6f);

numbers.add(5.9f);

Iterator<Float> iterator = numbers.iterator();

float sum = 0.0f;

while(iterator.hasNext()) {

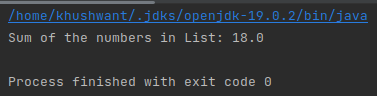
sum += iterator.next();

}

System.*out*.println("Sum of the numbers in List: " + sum);

}

}



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.

class Employee implements Comparable<Employee> {

private Double age;

private Double salary;

private String name;

public Employee(Double age, Double salary, String name) {

this.age = age;

this.salary = salary;

this.name = name;

}

public Double getAge() {

return age;

}

public Double getSalary() {

return salary;

}

public String getName() {

return name;

}

@Override

public int compareTo(Employee otherEmployee) {

return this.name.compareTo(otherEmployee.getName());

}

}

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<Employee> employees = new ArrayList<>();

employees.add(new Employee(25.0, 2500.0, "John Doe"));

employees.add(new Employee(30.0, 3000.0, "Alice Smith"));

employees.add(new Employee(35.0, 3500.0, "Bob Johnson"));

employees.add(new Employee(40.0, 4000.0, "Charlie Brown"));

System.*out*.println("Before sorting:");

for (Employee emp : employees) {

System.*out*.println(emp.getName() + " - " + emp.getSalary());

}

Collections.*sort*(employees);

System.*out*.println("After sorting:");

for (Employee emp : employees) {

System.*out*.println(emp.getName() + " - " + emp.getSalary());

}

Comparator<Employee> salaryComparator = new Comparator<Employee>() {

@Override

public int compare(Employee o1, Employee o2) {

return o1.getSalary().compareTo(o2.getSalary());

}

};

Collections.*sort*(employees, salaryComparator);

System.*out*.println("Sorting base on salary:");

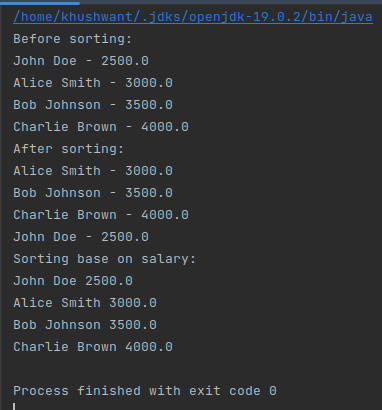
for (Employee employee : employees) {

System.*out*.println(employee.getName() + " " + employee.getSalary());

}

}

}



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))

import java.util.Stack;

public class SpecialStack {

private Stack<Integer> stack;

private Stack<Integer> minStack;

public SpecialStack() {

stack = new Stack<>();

minStack = new Stack<>();

}

public void push(int value) {

stack.push(value);

if (minStack.isEmpty() || value <= minStack.peek()) {

minStack.push(value);

}

}

public int pop() {

int value = stack.pop();

if (value == minStack.peek()) {

minStack.pop();

}

return value;

}

public boolean isEmpty() {

return stack.isEmpty();

}

public boolean isFull() {

return false;

}

public int getMin() {

return minStack.peek();

}

}

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

import java.util.HashMap;

public class Employee {

String name = "";

int Age = 0;

String Designation = "";

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return Age;

}

public void setAge(int age) {

Age = age;

}

public String getDesignation() {

return Designation;

}

public void setDesignation(String designation) {

Designation = designation;

}

public Employee(String name, int age, String designation) {

this.name = name;

Age = age;

Designation = designation;

}

@Override

public String toString() {

return

"name = " + name +"\n"+

"Age = " + Age +"\n"+

"Designation = " + Designation +"\n"+

"Salary ";

}

public static void main(String[] args) {

HashMap<Employee,Integer> map = new HashMap<>();

map.put(new Employee("Khushwant",23,"Trainee"),15000);

System.*out*.println(map);

}

}

