package employeeManagementSystem;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.Scanner;

import java.util.stream.Stream;

public class EmployeeManagementSystem {

public static void main(String[] args) {

EmployeeManager employeeManager = new EmployeeManager();

employeeManager.startUp();

}

}

class EmployeeManager{

private Employee employee;

public EmployeeManager() {

employee = new Employee();

}

public void startUp(){

Scanner input = new Scanner(System.in);

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\t\t\t------- Welcome to Mahin IT Employee Management System --------");

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("Enter Admin Email: ");

String adminEmail = input.nextLine();

System.out.print("Enter Admin Pssword: ");

String adminPassword = input.nextLine();

System.out.println();

int counterLoginTrial = 0;

while(counterLoginTrial<2) {

if ((adminEmail.equalsIgnoreCase("mahin@gmail.com")) && (adminPassword.equals("mahin"))) {

boolean flag = true;

while (flag) {

System.out.println("\tMenu: ");

System.out.println("\t1. Add Employee: ");

System.out.println("\t2. Show Employee: ");

System.out.println("\t3. Search Employee by Id: ");

System.out.println("\t4. Show Maximum Salary Holder info: ");

System.out.println("\t5. Show Minimum Salary Holder info: ");

System.out.println("\t6. Filtering on Employee's Salary: ");

System.out.println("\t7. Total Employee: ");

System.out.println("\t8. Employee Bonus Calculator: ");

System.out.println("\t0. Exit: ");

System.out.print("\n\tEnter Option: ");

int option = input.nextInt();

System.out.println();

switch (option) {

case 1:

addingEmployeeInfoInManager();

continue;

case 2:

showingEmployeeInfoInManager();

continue;

case 3:

searchingEmployeeInfoInManager();

continue;

case 4:

showMaxSalaryInfoInManager();

continue;

case 5:

showMinSalaryInfoInManager();

continue;

case 6:

filterEmployeeInfoInManager();

continue;

case 7:

totalEmployee();

continue;

case 8:

bonusCalculatorForEmployee();

continue;

default :

System.out.print("Operation End. Thanks for Using");

System.exit(option);

}

}

}

else {

System.out.println("Invalid Information. Enter Correct one. ");

System.out.print("Enter Admin Name: ");

adminEmail = input.nextLine();

System.out.print("Enter Admin Pssword: ");

adminPassword = input.nextLine();

System.out.println();

counterLoginTrial++;

if(counterLoginTrial == 3) {

System.out.println("Currently You are blocked");

break;

}

}

}

System.out.println("Try again later with valid info.");

System.out.println("Thanks.");

input.close();

}

public void addingEmployeeInfoInManager() {

employee.addEmployeeToEmployeeList();

System.out.println("Successfully Added all the Employee in Employee list.\n\n");

}

public void showingEmployeeInfoInManager() {

System.out.println("Employee List: ");

employee.showEmployeeFromEmployeeList();

}

public void searchingEmployeeInfoInManager() {

System.out.print("Enter a job Id to Search: ");

employee.searchEmployeeFromEmployeeList();

}

public void showMaxSalaryInfoInManager() {

System.out.println("Max Salary Holder Employee Info in details: ");

employee.showMaxSalaryHolderInfoFromEmployeeList();

}

public void showMinSalaryInfoInManager() {

System.out.println("Min Salary Holder Employee Info in details: ");

employee.showMinSalaryHolderInfoFromEmployeeList();

}

public void filterEmployeeInfoInManager() {

employee.filteringEmployeeList();

}

public void totalEmployee() {

employee.totalEmployeeCounter();

}

public void bonusCalculatorForEmployee() {

employee.bonusCalculator();

}

}

class EmployeeFiles{

public static final String employeeInfoCenter = "D:\\C & Java Programming\\All Java Code Volt\\Java-Programming\\Java\\B Summer2021 Java Lab\\src\\employeeManagementSystem";

public static final String employeeList = employeeInfoCenter + "\\Employee List.txt";

public EmployeeFiles(){

}

public static void addEmployeeInfoInFile(Employee[] employee, String listName) {

try {

FileWriter myfile = new FileWriter(listName, true);

for (int i = 0; i < employee.length; i++) {

myfile.write(employee[i].toString() + "\n");

//myfile.write(employee[i].getName() + " " + String.valueOf(employee[i].getAge()) + " " + String.valueOf(employee[i].getId()) + " " + String.valueOf(employee[i].getSalary()) + " " + employee[i].getDesignation() + "\n");

}

myfile.close();

} catch(IOException e) {

System.out.println("Writer File is not Working");

}

}

public static void showEmployeeInfoFromFile(String listName) {

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

String line;

while((line = br.readLine()) != null) {

String name = line.split(" ")[0] + " " + line.split(" ")[1];

String age = line.split(" ")[2] + " " + line.split(" ")[3];

String id = line.split(" ")[4] + " " + line.split(" ")[5];

String salary = line.split(" ")[6] + " " + line.split(" ")[7];

String designation = line.split(" ")[8] + " " + line.split(" ")[9];

System.out.println(name + " " + age + " " + id + " " + salary + " " + designation);

}

fr.close();

br.close();

System.out.println("\n");

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

}

public static void searchEmployeeInfoFromFile(String listName, String searchId) {

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

System.out.println("Searching Result: ");

String line = null;

String name, age, id, salary, designation;

name=age=id=salary=designation = null;

boolean isExists = false;

while((line = br.readLine()) != null) {

name = line.split(" ")[0] + " " + line.split(" ")[1];

age = line.split(" ")[2] + " " + line.split(" ")[3];

id = line.split(" ")[4] + " " + line.split(" ")[5];

salary = line.split(" ")[6] + " " + line.split(" ")[7];

designation = line.split(" ")[8] + " " + line.split(" ")[9];

if(line.split(" ")[5].equalsIgnoreCase(searchId)) {

isExists = true;

break;

}

}

if(isExists == true) {

System.out.println(name + " " + age + " " + id + " " + salary + " " + designation);

}

else

System.out.println("Sorry, Employee not found in Employee List.");

fr.close();

br.close();

System.out.println("\n");

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

}

public static void searchEmployeeMaxSalaryInfoFromFile(String listName) {

ArrayList<Employee> employeeList = new ArrayList<Employee>();

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

String line;

while((line = br.readLine()) != null) {

String name = line.split(" ")[1];

int age = Integer.parseInt(line.split(" ")[3]);

int id = Integer.parseInt(line.split(" ")[5]);

int salary = Integer.parseInt(line.split(" ")[7]);

String designation = line.split(" ")[9];

Employee emp = new Employee(name, age, id, salary, designation);

employeeList.add(emp);

}

// Sorting on salary property

Collections.sort(employeeList, EmployeeDataHelper.employeeSalary);

// Print Maximum Salary Holder Info

int maximumSalaryHoldePosition = employeeList.size()-1;

System.out.println(employeeList.get(maximumSalaryHoldePosition).toString());

fr.close();

br.close();

System.out.println("\n");

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

}

public static void searchEmployeeMinSalaryInfoFromFile(String listName) {

ArrayList<Employee> employeeList = new ArrayList<Employee>();

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

String line;

while((line = br.readLine()) != null) {

String name = line.split(" ")[1];

int age = Integer.parseInt(line.split(" ")[3]);

int id = Integer.parseInt(line.split(" ")[5]);

int salary = Integer.parseInt(line.split(" ")[7]);

String designation = line.split(" ")[9];

Employee emp = new Employee(name, age, id, salary, designation);

employeeList.add(emp);

}

// Sorting on salary property

Collections.sort(employeeList, EmployeeDataHelper.employeeSalary);

// Print Minimum Salary Holder Info

System.out.println(employeeList.get(0).toString());

fr.close();

br.close();

System.out.println("\n");

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

}

public static void filterEmployeeInfoFromFile(String listName, int lowRange, int highRange) {

ArrayList<Employee> employeeList = new ArrayList<Employee>();

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

String line;

while((line = br.readLine()) != null) {

String name = line.split(" ")[1];

int age = Integer.parseInt(line.split(" ")[3]);

int id = Integer.parseInt(line.split(" ")[5]);

int salary = Integer.parseInt(line.split(" ")[7]);

String designation = line.split(" ")[9];

Employee emp = new Employee(name, age, id, salary, designation);

employeeList.add(emp);

}

// After getting filtering Print Higher and Lower salary range employee's Info

Stream<Employee> filteredEmployeeList = employeeList.stream().filter(e -> (e.getSalary()>=lowRange) && (e.getSalary()<=highRange));

System.out.println("\nFiltering Result of " + lowRange + " to " + highRange + ": ");

filteredEmployeeList.forEach(employee -> System.out.print(employee.toString() + "\n"));

fr.close();

br.close();

System.out.println("\n");

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

}

public static int countEmployeeNumberFromFile(String listName) {

int employeeCounter = 0;

try {

FileReader fr = new FileReader(listName);

BufferedReader br = new BufferedReader(fr);

String line;

while((line = br.readLine()) != null) {

employeeCounter++;

}

fr.close();

br.close();

} catch(IOException e) {

System.out.println("File Reader is not Working");

}

return employeeCounter;

}

}

class Address{

private int houseNo;

private int roadNo;

private String area;

public Address() {

}

public Address(int houseNo, int roadNo, String area) {

this.houseNo = houseNo;

this.roadNo = roadNo;

this.area = area;

}

public int getHouseNo() {

return houseNo;

}

public void setHouseNo(int houseNo) {

this.houseNo = houseNo;

}

public int getRoadNo() {

return roadNo;

}

public void setRoadNo(int roadNo) {

this.roadNo = roadNo;

}

public String getArea() {

return area;

}

public void setArea(String area) {

this.area = area;

}

@Override

public String toString() {

return "Address [House No: " + houseNo + ", Road No: " + roadNo + ", Area: " + area + "]";

}

}

abstract class Person{

private String name;

private int age;

private Address address;

public Person() {

}

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public Person(String name, int age, Address address) {

this.name = name;

this.age = age;

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public abstract void bonusCalculator();

public abstract void totalEmployeeCounter();

@Override

public String toString() {

return "Person [name=" + name + ", age=" + age + ", address=" + address + "]";

}

}

class Contact{

private String mobileNo;

private String email;

public Contact() {

}

public Contact(String mobileNo, String email) {

this.mobileNo = mobileNo;

this.email = email;

}

public String getMobileNo() {

return mobileNo;

}

public void setMobileNo(String mobileNo) {

this.mobileNo = mobileNo;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

public String toString() {

return "Contact [Mobile No: " + mobileNo + ", Email: " + email + "]";

}

}

class Employee extends Person{

private int id;

private int salary;

private String designation;

private Contact contact;

public Employee () {

}

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

super(name, age);

this.id = id;

this.salary = salary;

this.designation = designation;

}

public Employee(String name, int age, int jobId, int salary, String designation, Contact contact) {

super(name, age);

this.id = jobId;

this.salary = salary;

this.designation = designation;

this.contact = contact;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public void addEmployeeToEmployeeList(){

Scanner input = new Scanner(System.in);

EmployeeDataHelper employeeData = new EmployeeDataHelper();

System.out.print("Enter How many Employee you want to add in Employee list: ");

int numberOfEmployee = input.nextInt();

if(numberOfEmployee >= 1) {

Employee[] employee = new Employee[numberOfEmployee];

Employee[] employeeList = employeeData.employeeDataInput(employee);

// File write

EmployeeFiles.addEmployeeInfoInFile(employeeList, EmployeeFiles.employeeList);

}

}

void showEmployeeFromEmployeeList() {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataShow();

}

void searchEmployeeFromEmployeeList() {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataSearch();

}

void showMaxSalaryHolderInfoFromEmployeeList() {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataSearchMaxSallary();

}

public void showMinSalaryHolderInfoFromEmployeeList() {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataSearchMinSallary();

}

public void filteringEmployeeList( ) {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataFilter();

}

public void totalEmployeeCounter() {

EmployeeDataHelper employeeData = new EmployeeDataHelper();

employeeData.employeeDataCounter();

}

public void bonusCalculator() {

Scanner input = new Scanner(System.in);

EmployeeDataHelper employeeData = new EmployeeDataHelper();

System.out.print("Enter Employee Desination: ");

designation = input.next();

System.out.print("Enter Employee Current Salary: ");

salary = input.nextInt();

int bonusAmmount = employeeData.bonusSalaryCalculate(designation, salary);

System.out.print("Bonus ammont for a " + designation + " is: " + bonusAmmount + " BDT");

System.out.println();

System.out.print("Total Salary for a " + designation + " is: " + (bonusAmmount+salary) + " BDT");

System.out.println("\n");

}

@Override

public String toString() {

return "Name: " + super.getName() + " Age: " + super.getAge() + " EmployeeId: " + id + " Salary: " + salary + " Designation: " + designation;

}

}

class InvalidSalaryException extends Exception{

public InvalidSalaryException(String errorMessage){

super(errorMessage);

}

}

class InvalidAgeException extends Exception{

public InvalidAgeException(String errorMessage){

super(errorMessage);

}

}

interface SalaryCalculation{

public int bonusSalaryCalculate(String designation, int salary);

}

class EmployeeDataHelper implements SalaryCalculation {

Employee employeeObject;

public Employee[] employeeDataInput(Employee[] employees) {

employeeObject = new Employee();

Scanner input = new Scanner(System.in);

for (int i = 0; i < employees.length; i++) {

System.out.println("Enter Employee " + (i+1) + " details: ");

System.out.print("Enter Employee Name: ");

employeeObject.setName(input.next());

System.out.print("Enter Employee age: ");

employeeObject.setAge(input.nextInt());

try {

checkAgeValidity(employeeObject.getAge());

} catch (InvalidAgeException e) {

System.out.println(e);

}

System.out.print("Enter Employee Id: ");

employeeObject.setId(input.nextInt());

System.out.print("Enter Employee Salary: ");

employeeObject.setSalary(input.nextInt());

try {

checkSalaryValidity(employeeObject.getSalary());

} catch (InvalidSalaryException e) {

System.out.println(e);

}

System.out.print("Enter Employee Designation: ");

employeeObject.setDesignation(input.next());

employees[i] = new Employee(employeeObject.getName(), employeeObject.getAge(), employeeObject.getId(), employeeObject.getSalary(), employeeObject.getDesignation());

System.out.println();

}

return employees;

}

public void checkSalaryValidity(int salary) throws InvalidSalaryException{

if(salary<0)

new InvalidSalaryException("Salary shouldn't be Negative");

}

public void checkAgeValidity(int salary) throws InvalidAgeException{

if(salary<0)

new InvalidAgeException("Age shouldn't be less then 20 and not more then 60");

}

public void employeeDataShow() {

EmployeeFiles.showEmployeeInfoFromFile(EmployeeFiles.employeeList);

}

public void employeeDataSearch() {

Scanner input = new Scanner(System.in);

String searchEmployeeId = input.next();

EmployeeFiles.searchEmployeeInfoFromFile(EmployeeFiles.employeeList, searchEmployeeId);

}

public void employeeDataSearchMaxSallary() {

EmployeeFiles.searchEmployeeMaxSalaryInfoFromFile(EmployeeFiles.employeeList);

}

public void employeeDataSearchMinSallary() {

EmployeeFiles.searchEmployeeMinSalaryInfoFromFile(EmployeeFiles.employeeList);

}

void employeeDataFilter( ) {

Scanner input = new Scanner(System.in);

// Salary Info Range

System.out.print("Enter Lower range for filtering: ");

int lowRange = 0;

lowRange = input.nextInt();

System.out.print("Enter Higher range for filtering: ");

int highRange = 0;

highRange = input.nextInt();

EmployeeFiles.filterEmployeeInfoFromFile(EmployeeFiles.employeeList, lowRange, highRange);

}

void employeeDataCounter() {

System.out.print("Total Employee: " + EmployeeFiles.countEmployeeNumberFromFile(EmployeeFiles.employeeList));

System.out.println("\n");

}

public static Comparator<Employee> employeeSalary = new Comparator<Employee>() {

public int compare(Employee e1, Employee e2) {

int salary1 = e1.getSalary();

int salary2 = e2.getSalary();

return salary1-salary2; // For ascending order

}

};

@Override

public int bonusSalaryCalculate(String designation, int salary) {

int bonousSalaryTotal = 0;

if(designation.trim().equalsIgnoreCase("JuniorDeveloper")) {

bonousSalaryTotal = (int)(salary\*0.10);

}

else if(designation.trim().equalsIgnoreCase("SeniorDeveloper")) {

bonousSalaryTotal = (int)(salary\*0.20);

}

else if(designation.trim().equalsIgnoreCase("TeamLead")) {

bonousSalaryTotal = (int)(salary\*0.40);

}

else if(designation.trim().equalsIgnoreCase("ProductManager")) {

bonousSalaryTotal = (int)(salary\*0.50);

}

else if(designation.trim().equalsIgnoreCase("CEO")) {

bonousSalaryTotal = (int)(salary\*0.70);

}

return bonousSalaryTotal;

}

}