**ASSIGNMENTS**

**1. Based On Classes and Object:**

public class Main {

public static void main(String[] args) {

System.out.println("Hi Everyone!");

Employee emp= new Employee();

System.out.print("My name is "+emp.firstName+" "+emp.lastName);

System.out.print(", my employee ID is " +emp.employeeId+", my email ID is "+emp.emailId);

System.out.print(" and my current CTC is " +emp.salary+'.');

}

}

class Employee{

int employeeId=817379;

String firstName="Notakaru";

String lastName="Thulasiram";

String emailId="NT00817379@TechMahindra.com";

float salary=350000f;

}

**OUTPUT:**

Hi Everyone!

My name is Notakaru Thulasiram, my employee ID is 817379, my email ID is NT00817379@TechMahindra.com and my current CTC is 350000.0

**2.Based On Array:**

import java.util.\*;

class Main{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

int size;

//taking input from user for size

System.out.print("Enter array size : ");

size = sc.nextInt();

int[] numArray = new int[size];

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

{

numArray[i] = i+1;

displayForLoop(numArray);

displayForEachLoop(numArray);

findSumOfArray(numArray);

findAvgOfArray(numArray);

}

public static void displayForLoop(int[] arr)

{

System.out.println("Display Array Elements through for-loop : ");

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

{

System.out.print(arr[i]+",");

}

System.out.println();

}

public static void displayForEachLoop(int[] arr)

{

System.out.println("Display Array Elements through for-each loop : ");

for(int num : arr)

{

System.out.print(num+",");

}

System.out.println();

}

public static void findSumOfArray(int[] arr)

{

System.out.println("calculating sum of all elements in array : ");

int sum=0;

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

{

sum += arr[i];

}

System.out.println("Sum of Elements : "+sum);

}

public static void findAvgOfArray(int[] arr)

{

System.out.println("calculating average of all elements in array : ");

int sum=0;

float avg=0.0f;

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

{

sum += arr[i];

}

avg = (float)sum/arr.length ;

System.out.println("Average of elements : "+avg);

}

}

**OUTPUT:**

Enter array size : 10

Display Array Elements through for-loop :

1,2,3,4,5,6,7,8,9,10,

Display Array Elements through for-each loop :

1,2,3,4,5,6,7,8,9,10,

calculating sum of all elements in array :

Sum of Elements : 55

calculating average of all elements in array :

Average of elements : 5.5

**3.Based on Array Count repeating String:**

public class Main {

public static void main(String[] args) {

String[] strArray = {"abc", "dce", "xyz", "abc", "xyz"};

for(String str: getUniqueItems(strArray)) {

if (str == null) continue; // ignore uninitialized values.

System.out.println(str + " repeated: " + count(strArray, str) + " times");

}

}

// checks whether a string exists in array.

static boolean exists(String[] arr, String target) {

for(String str: arr) {

if(str == null) continue; // skip uninitialized slots.

if(str.equals(target)) return true;

}

return false;

}

// returns only unique items in array.

static String[] getUniqueItems(String[] arr) {

String[] uniqueItems = new String[arr.length];

int len = 0;

for(String str: arr)

if (exists(uniqueItems, str)) continue;

else uniqueItems[len++] = str;

return uniqueItems;

}

// count the number of times a string is repeated in array.

static int count(String[] arr, String target) {

int count = 0;

for(String str: arr)

if(str.equals(target)) count++;

return count;

}

}

**OUTPUT:**

abc repeated: 2 times

dce repeated: 1 times

xyz repeated: 2 times

**4.Store Employee Object in Array and display :**

public class Main {

public static void main(String[] args) {

Employee[] employees = new Employee[2];

employees[0] = new Employee(1, "Thulasi", "ram", "Thulasi@ram.com");

employees[1] = new Employee(2, "Hari", "priya", "Hari@priya.com");

for(Employee e: employees) {

e.display();

}

}

}

class Employee {

int id;

String fname;

String lname;

String email;

Employee(int id, String fname, String lname, String email) {

this.id = id;

this.fname = fname;

this.lname = lname;

this.email = email;

}

void display() {

System.out.println("name: " + fname + " " + lname + ", email: " + email + ", id: " + id);

}

**OUTPUT:**

name: thulasi ram, email: thulasi@ram.com, id: 1

name: hari priya, email: hari@priya.com, id: 2

**5.**  **Based on Array, Exception and toString() having 2 users, Admin and Normal user (objects being created before showing ui, login page/ asking to enter creds)**

**a. admin can do any operation**

**b. normal user can just view his employee details**

**c. program start with showing login ui, asking username and password**

**d. user will be authenticated, based on permission ui will be shown below option, user can enter below options.**

**If Admin:**

**i. Add Employee**

**ii. Update Employee (based on id)**

**iii. Delete Employee (based on id)**

**iv. View Employee (based on id)**

**v. view all employees**

**if Normal user:**

**i. View Own Employee Details**

**Handle Exception:**

**Throw custom exception if creds entered are wrong and terminate the program.**

import java.util.Scanner;

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

// scanner setup

Scanner sc = new Scanner(System.in);

Employee[] records = new Employee[100];

// pre existing users.

records[0] = new Employee(99, "thulasi", "ram", 46000, "ram@abc.xyz", "password1", true);

records[1] = new Employee(100, "hari", "priya", 45000, "kumar@abc.xyz", "password2", false);

// start interactive session

// authenticate first.

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

System.out.print("Enter you email: ");

String loginEmail = sc.next();

System.out.print("Enter your password: ");

String loginPassword = sc.next();

// authenticate

Employee loggedInUser = null;

for (Employee emp : records) {

if (emp == null) continue; // ignore empty slots

if (emp.email.equals(loginEmail) && emp.password.equals(loginPassword)) {// found record

loggedInUser = emp;

break;

}

}

if (loggedInUser == null) { // no user found.

throw new InvalidCredsException("no such user found. exiting");

}

System.out.println("Welcome " + loggedInUser.firstName + "...");

while (true) {

System.out.print("Enter your action(add/update/remove/display/displayAll): ");

String choice = sc.next();

if (choice.equals("exit")) break; // exit program

boolean choiceAllowed = false;

for (String perm : loggedInUser.permissions) {

if (perm.equals(choice)) {

choiceAllowed = true;

break;

}

}

if (!choiceAllowed) {

System.out.println("invalid or unauthorized action.");

continue;

}

switch (choice) {

case "add":

System.out.print("Enter ID: ");

int id = sc.nextInt();

System.out.print("Enter first name: ");

String firstName = sc.next();

System.out.print("Enter last name: ");

String lastName = sc.next();

System.out.print("Enter the salary: ");

float salary = sc.nextFloat();

System.out.print("Enter email ID: ");

String email = sc.next();

System.out.print("Enter your password: ");

String password = sc.next();

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

if (records[i] == null) {

records[i] = new Employee(id, firstName, lastName, salary, email, password, false);

}

}

break;

case "remove":

System.out.println("Enter ID of employee to remove: ");

int givenId = sc.nextInt();

boolean removed = false;

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

if (records[i] == null) continue; // skip empty slots;

if (records[i].id == givenId) {

records[i] = null;

removed = true;

break;

}

}

if (removed)

System.out.println("Record removed successfully");

else

System.out.println("no such record found");

break;

case "update":

System.out.println("Enter ID of employee to update: ");

givenId = sc.nextInt();

for (Employee e : records) {

if (e == null) continue; // ignore empty slots.

if (e.id == givenId) {

System.out.println("Enter the field to update (firstName, lastName, email):");

String fieldName = sc.next();

System.out.println("Enter the field value: ");

String fieldValue = sc.next();

e.updateRecord(fieldName, fieldValue);

}

}

break;

case "display":

if (!loggedInUser.isAdmin()) loggedInUser.showDetails();

else {

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

givenId = sc.nextInt();

boolean found = false;

for (Employee e : records) {

if (e == null) continue;

if (e.id == givenId) {

e.showDetails();

found = true;

break;

}

}

if (!found)

System.out.println("the record was not found");

}

break;

case "displayAll":

for (Employee e : records) {

if (e == null) continue;

e.showDetails();

}

break;

default:

System.out.println("Invalid choice.");

}

}

}

}

class Employee {

final int id;

String firstName;

String lastName;

float salary;

String email;

String password;

boolean admin;

String[] permissions; // default permissions

Employee(int id, String firstName, String lastName, float salary,

String email, String password, boolean admin) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

this.email = email;

this.password = password;

this.admin = admin;

// if admin, give more permissions.

if (admin) {

permissions = new String[]{"add", "remove", "update", "display", "displayAll"};

} else {

permissions = new String[]{"display"};

}

}

public String toString() {

String text = "Full Name: " + this.firstName + " " + this.lastName + "\n"

+ "ID: " + this.id + "\n"

+ "email: " + this.email + "\n"

+ "Salary: " + this.salary + "\n"

+ "Admin: " + this.admin + "\n"

+ "----";

return text;

}

public void showDetails() {

System.out.println(this.toString());

}

boolean isAdmin() {

return this.admin;

}

public void updateRecord(String fieldName, String fieldValue) {

switch (fieldName) {

case "firstName":

this.firstName = fieldValue;

break;

case "lastName":

this.lastName = fieldValue;

break;

case "email":

this.email = fieldValue;

break;

default:

System.out.println("That field is not valid. no record updated");

}

}

}

class InvalidCredsException extends RuntimeException {

InvalidCredsException(String message) {

super(message);

}

}

**OUTPUT:**

**Output 1**

Authenticating...

Enter you email: awed

Enter your password: aadf

Exception in thread "main" InvalidCredsException: no such user found. exiting

at Main.main(Main.java:40)

**Output 2**

Authenticating...

Enter you email: kumar@abc.xyz

Enter your password: password2

Welcome hari...

Enter your action(add/update/remove/display/displayAll): display

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): remove

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): exit

**Output 3**

Authenticating...

Enter you email: ram@abc.xyz

Enter your password: password1

Welcome thulasi...

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: thulasi ram

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): update

Enter ID of employee to update:

100

Enter the field to update (firstName, lastName, email):

lastName

Enter the field value:

Sangakara

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari Sangakara

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

Enter ID: 102

Enter first name: new

Enter last name: user

Enter the salary: 40000

Enter email ID: newuser@abc.xyz

Enter your password: password3

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name:

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: arun Sangakara

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Full Name: new user

ID: 102

email: newuser@abc.xyz

Salary: 40000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): remove

Enter ID of employee to remove:

102

Record removed successfully

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: ram sager

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: arun Sangakara

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): exit

**6.Create threads in 2 ways, sleep the current running thread for 2 sec., change the thread name, print the current running thread name**

public class Main {

public static void main(String[] args) {

Thread t1 = new Task1();

Thread t2 = new Thread(new Task2());

t1.start();

t2.start();

}

}

class Task1 extends Thread {

public void run() {

try {

Thread.sleep(2000);

} catch (InterruptedException e) {}

Thread.currentThread().setName("newthread1name");

System.out.println("Thread name: " + Thread.currentThread().getName());

}

}

class Task2 implements Runnable {

public void run() {

try {

Thread.sleep(2000);

} catch (InterruptedException e) {}

Thread.currentThread().setName("newthread2name");

System.out.println("Thread name: " + Thread.currentThread().getName());

}

}

**OUTPUT:**

Thread name: newthread1name

Thread name: newthread2name

**7. Add Employees to collection and work with some methods like add(), addAll(), remove(), removeAll(), contains(), size(), clear() etc.,**

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

public class Main {

public static void main(String[] args) {

ArrayList<Employee> empList = new ArrayList<>(2);

Employee e1 = new Employee(1, "thulasi", "ram", "thulasi@ram.com");

Employee e2 = new Employee(2, "hari", "priya", "hari@priya.com");

// add()

empList.add(e1);

empList.add(e2);

System.out.println("after adding 2 entires:");

System.out.println(empList);

// remove()

empList.remove(1);

System.out.println("after removing emp at 1st index:");

System.out.println(empList);

// contains()

System.out.println("empList contains e1: " + empList.contains(e1));

// size();

System.out.println("empList size: " + empList.size());

// clear();

empList.clear();

System.out.println("after clearing empList: " + empList);

// addAll()

List<Employee> addList = new ArrayList<>(2);

addList.add(e1);

addList.add(e2);

empList.addAll(addList);

System.out.println("reset empList: " + empList);

// removeAll()

List<Employee> removeList = new LinkedList<>();

removeList.add(e1);

empList.removeAll(removeList);

System.out.println("after removing all items in this list -> " + removeList + " from empList, the following remain. empList: " + empList);

}

}

class Employee {

int id;

String fname;

String lname;

String email;

Employee(int id, String fname, String lname, String email) {

this.id = id;

this.fname = fname;

this.lname = lname;

this.email = email;

}

void display() {

System.out.println(toString());

}

public String toString() {

return ("{name: " + fname + " " + lname + ", email: " + email + ", id: " + id + "}");

}

}

**Output**

after adding 2 entires:

[{name: thulasi ram, email: thulasi@ram.com, id: 1}, {name: hari priya, email: hari@priya.com, id: 2}]

after removing emp at 1st index:

[{name: thulasi ram, email: thulasi@ram.com, id: 1}]

empList contains e1: true

empList size: 1

after clearing empList: []

reset empList: [{name: thulasi ram, email: thulasi@ram.com, id: 1}, {name: hari priya, email: hari@priya.com, id: 2}]

after removing all items in this list -> [{name: thulasi ram, email: thulasi@ram.com, id: 1}] from empList, the following remain. empList: [{name: hari priya, email: hari@priya.com, id: 2}]

**8) List of Employees (No duplicates Allowed) and the list can be sorted based on ID or name**

import java.util.\*;

public class Main {

public static void main(String[] args) {

LinkedHashSet<Employee> list = new LinkedHashSet<>();

Employee e1 = new Employee(2, "thulasi");

Employee e2 = new Employee(1, "ram");

Employee e3 = new Employee(3, "hari");

list.add(e1);

list.add(e2);

list.add(e2); // duplicate wont get added.

list.add(e3);

list.add(e3); // duplicate wont get added

System.out.println("employees without duplicates:");

System.out.println(list);

List<Employee> sorted = new ArrayList<>(list);

sorted.sort(new IdComparator());

System.out.println("sorted via id:");

System.out.println(sorted);

sorted.sort(new NameComparator());

System.out.println("sorted via name:");

System.out.println(sorted);

}

}

class IdComparator implements Comparator<Employee> {

@Override

public int compare(Employee o1, Employee o2) {

if (o1.id == o2.id) return 0;

else if (o1.id > o2.id) return 1;

else return -1;

}

}

class NameComparator implements Comparator<Employee> {

@Override

public int compare(Employee o1, Employee o2) {

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

}

}

class Employee {

int id;

String name;

Employee(int id, String name) {

this.id = id;

this.name = name;

}

public String toString() {

return ("{name: " + name + ", id: " + id + "}");

}

}

**Output**

employees without duplicates:

[{name: thulasi, id: 2}, {name: ram, id: 1}, {name: hari, id: 3}]

sorted via id:

[{name: ram, id: 1}, {name: thulasi, id: 2}, {name: hari, id: 3}]

sorted via name:

[{name: thulasi, id: 2}, {name: hari, id: 3}, {name: ram, id: 1}]

**9) Employees Management Functionality (problem 5) achieved by using collection framework**

import java.util.Scanner;

import java.util.ArrayList;

class Main {

public static void main(String[] args) {

// scanner setup

Scanner sc = new Scanner(System.in);

Employee[] records = new Employee[100];

// pre existing users.

records[0] = new Employee(99, "thulasi", "ram", 46000, "ram@abc.xyz", "password1", true);

records[1] = new Employee(100, "hari", "priya", 45000, "kumar@abc.xyz", "password2", false);

// start interactive session

// authenticate first.

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

System.out.print("Enter you email: ");

String loginEmail = sc.next();

System.out.print("Enter your password: ");

String loginPassword = sc.next();

// authenticate

Employee loggedInUser = null;

for (Employee emp : records) {

if (emp == null) continue; // ignore empty slots

if (emp.email.equals(loginEmail) && emp.password.equals(loginPassword)) {// found record

loggedInUser = emp;

break;

}

}

if (loggedInUser == null) { // no user found.

throw new InvalidCredsException("no such user found. exiting");

}

System.out.println("Welcome " + loggedInUser.firstName + "...");

while (true) {

System.out.print("Enter your action(add/update/remove/display/displayAll): ");

String choice = sc.next();

if (choice.equals("exit")) break; // exit program

boolean choiceAllowed = false;

for (String perm : loggedInUser.permissions) {

if (perm.equals(choice)) {

choiceAllowed = true;

break;

}

}

if (!choiceAllowed) {

System.out.println("invalid or unauthorized action.");

continue;

}

switch (choice) {

case "add":

System.out.print("Enter ID: ");

int id = sc.nextInt();

System.out.print("Enter first name: ");

String firstName = sc.next();

System.out.print("Enter last name: ");

String lastName = sc.next();

System.out.print("Enter the salary: ");

float salary = sc.nextFloat();

System.out.print("Enter email ID: ");

String email = sc.next();

System.out.print("Enter your password: ");

String password = sc.next();

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

if (records[i] == null) {

records[i] = new Employee(id, firstName, lastName, salary, email, password, false);

}

}

break;

case "remove":

System.out.println("Enter ID of employee to remove: ");

int givenId = sc.nextInt();

boolean removed = false;

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

if (records[i] == null) continue; // skip empty slots;

if (records[i].id == givenId) {

records[i] = null;

removed = true;

break;

}

}

if (removed)

System.out.println("Record removed successfully");

else

System.out.println("no such record found");

break;

case "update":

System.out.println("Enter ID of employee to update: ");

givenId = sc.nextInt();

for (Employee e : records) {

if (e == null) continue; // ignore empty slots.

if (e.id == givenId) {

System.out.println("Enter the field to update (firstName, lastName, email):");

String fieldName = sc.next();

System.out.println("Enter the field value: ");

String fieldValue = sc.next();

e.updateRecord(fieldName, fieldValue);

}

}

break;

case "display":

if (!loggedInUser.isAdmin()) loggedInUser.showDetails();

else {

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

givenId = sc.nextInt();

boolean found = false;

for (Employee e : records) {

if (e == null) continue;

if (e.id == givenId) {

e.showDetails();

found = true;

break;

}

}

if (!found)

System.out.println("the record was not found");

}

break;

case "displayAll":

for (Employee e : records) {

if (e == null) continue;

e.showDetails();

}

break;

default:

System.out.println("Invalid choice.");

}

}

}

}

class Employee {

final int id;

String firstName;

String lastName;

float salary;

String email;

String password;

boolean admin;

String[] permissions; // default permissions

Employee(int id, String firstName, String lastName, float salary,

String email, String password, boolean admin) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

this.email = email;

this.password = password;

this.admin = admin;

// if admin, give more permissions.

if (admin) {

permissions = new String[]{"add", "remove", "update", "display", "displayAll"};

} else {

permissions = new String[]{"display"};

}

}

public String toString() {

String text = "Full Name: " + this.firstName + " " + this.lastName + "\n"

+ "ID: " + this.id + "\n"

+ "email: " + this.email + "\n"

+ "Salary: " + this.salary + "\n"

+ "Admin: " + this.admin + "\n"

+ "----";

return text;

}

public void showDetails() {

System.out.println(this.toString());

}

boolean isAdmin() {

return this.admin;

}

public void updateRecord(String fieldName, String fieldValue) {

switch (fieldName) {

case "firstName":

this.firstName = fieldValue;

break;

case "lastName":

this.lastName = fieldValue;

break;

case "email":

this.email = fieldValue;

break;

default:

System.out.println("That field is not valid. no record updated");

}

}

}

class InvalidCredsException extends RuntimeException {

InvalidCredsException(String message) {

super(message);

}

}

**OUTPUT:**

**Output 1**

Authenticating...

Enter you email: awed

Enter your password: aadf

Exception in thread "main" InvalidCredsException: no such user found. exiting

at Main.main(Main.java:40)

**Output 2**

Authenticating...

Enter you email: kumar@abc.xyz

Enter your password: password2

Welcome hari...

Enter your action(add/update/remove/display/displayAll): display

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): remove

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): exit

**Output 3**

Authenticating...

Enter you email: ram@abc.xyz

Enter your password: password1

Welcome thulasi...

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: thulasi ram

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): update

Enter ID of employee to update:

100

Enter the field to update (firstName, lastName, email):

lastName

Enter the field value:

Sangakara

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari Sangakara

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

Enter ID: 102

Enter first name: new

Enter last name: user

Enter the salary: 40000

Enter email ID: newuser@abc.xyz

Enter your password: password3

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name:

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari Priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Full Name: new user

ID: 102

email: newuser@abc.xyz

Salary: 40000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): remove

Enter ID of employee to remove:

102

Record removed successfully

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: ram sager

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): exit

**10) Employee management functionality will be achieved using JDBC (Database)**

import java.util.Scanner;

import java.util.ArrayList;

import java.sql.\*;

class DB { // not the best way to do it, but oh well it works

private Connection connection;

private Statement statement;

DB(String url) throws SQLException {

connection = DriverManager.getConnection(url, "root", "");

statement = connection.createStatement();

}

ArrayList<Employee> load() throws SQLException {

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

ResultSet rs = statement.executeQuery("select \* from employees;");

while(rs.next()) {

records.add(new Employee(

rs.getInt("id"),

rs.getString("firstName"),

rs.getString("lastName"),

rs.getFloat("salary"),

rs.getString("email"),

rs.getString("password"),

rs.getBoolean("admin")

));

}

return records;

}

void persist(ArrayList<Employee> records) throws SQLException {

int result = statement.executeUpdate("truncate table employees;");

String query = "insert into employees values ";

for (Employee e: records) {

query += "( '" + e.id + "', '" + e.firstName + "', '" + e.lastName +

"', " + e.salary + ", '" + e.email + "', '" + e.password + "', " + e.admin + "),";

}

System.out.println(query);

result = statement.executeUpdate(query.substring(0, query.length() - 1));

}

}

class Main {

public static void main(String[] args) throws SQLException {

DB db = new DB("jdbc:mysql://localhost:3306/empdb");

// scanner setup

Scanner sc = new Scanner(System.in);

ArrayList<Employee> records= db.load();

// start interactive session

// authenticate first.

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

System.out.print("Enter you email: ");

String loginEmail = sc.next();

System.out.print("Enter your password: ");

String loginPassword = sc.next();

// authenticate

Employee loggedInUser = null;

for (Employee emp : records)

if (emp.email.equals(loginEmail) && emp.password.equals(loginPassword)) {// found record

loggedInUser = emp;

break;

}

if(loggedInUser == null) { // no user found.

throw new InvalidCredsException("no such user found. exiting");

}

System.out.println("Welcome " + loggedInUser.firstName + "...");

while(true) {

System.out.print("Enter your action(add/update/remove/display/displayAll): ");

String choice = sc.next();

if (choice.equals("exit")) break; // exit program

boolean chocieAllowed = false;

for (String perm: loggedInUser.permissions) {

if (perm.equals(choice)) {

chocieAllowed = true;

break;

}

}

if(!chocieAllowed) {

System.out.println("invalid or unauthorized action.");

continue;

}

switch(choice) {

case "add":

System.out.print("Enter ID: ");

int id = sc.nextInt();

System.out.print("Enter first name: ");

String firstName = sc.next();

System.out.print("Enter last name: ");

String lastName = sc.next();

System.out.print("Enter the salary: ");

float salary = sc.nextFloat();

System.out.print("Enter email ID: ");

String email = sc.next();

System.out.print("Enter your password: ");

String password = sc.next();

// add a new employee at the end of array.

records.add(new Employee(id, firstName, lastName, salary, email, password, false));

db.persist(records);

break;

case "remove":

System.out.println("Enter ID of employee to remove: ");

int givenId = sc.nextInt();

boolean removed = records.removeIf(e -> e.id == givenId);

if(removed) {

db.persist(records);

System.out.println("Record removed successfully");

}

else

System.out.println("no such record found");

break;

case "update":

System.out.println("Enter ID of employee to update: ");

givenId = sc.nextInt();

for(Employee emp: records)

if (emp.id == givenId) {

System.out.println("Enter the field to update (firstName, lastName, email): ");

String fieldName = sc.next();

System.out.println("Enter the field value: ");

String fieldValue = sc.next();

emp.updateRecord(fieldName, fieldValue);

db.persist(records);

}

break;

case "display":

if(!loggedInUser.isAdmin()) loggedInUser.showDetails();

else {

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

givenId = sc.nextInt();

boolean found = false;

for(Employee e: records)

if (e.id == givenId) {

e.showDetails();

found = true;

break;

}

if(!found)

System.out.println("the record was not found");

}

break;

case "displayAll":

for(Employee e: records) {

e.showDetails();

}

break;

default:

System.out.println("Invalid choice.");

}

}

}

}

class Employee {

final int id;

String firstName;

String lastName;

float salary;

String email;

String password;

boolean admin;

String[] permissions; // default permissions

Employee(int id, String firstName, String lastName, float salary,

String email, String password, boolean admin) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

this.email = email;

this.password = password;

this.admin = admin;

// if admin, give more permissions.

if (admin) {

permissions = new String[] {"add", "remove", "update", "display", "displayAll"};

} else {

permissions = new String[] {"display"};

}

}

public String toString() {

String text = "Full Name: " + this.firstName + " " + this.lastName + "\n"

+ "ID: " + this.id + "\n"

+ "email: " + this.email + "\n"

+ "Salary: " + this.salary + "\n"

+ "Admin: " + this.admin + "\n"

+ "----";

return text;

}

public void showDetails() {

System.out.println(this.toString());

}

boolean isAdmin() {

return this.admin ? true : false;

}

public void updateRecord(String fieldName, String fieldValue) {

switch(fieldName) {

case "firstName":

this.firstName = fieldValue;

break;

case "lastName":

this.lastName = fieldValue;

break;

case "email":

this.email = fieldValue;

default:

System.out.printf("That field is not valid. no record updated");

}

}

}

class InvalidCredsException extends RuntimeException {

InvalidCredsException(String message) {

super(message);

}

}

**OUTPUT:**

**Output 1**

Authenticating...

Enter you email: awed

Enter your password: aadf

Exception in thread "main" InvalidCredsException: no such user found. exiting

at Main.main(Main.java:40)

**Output 2**

Authenticating...

Enter you email: kumar@abc.xyz

Enter your password: password2

Welcome hari...

Enter your action(add/update/remove/display/displayAll): display

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): remove

invalid or unauthorized action.

Enter your action(add/update/remove/display/displayAll): exit

**Output 3**

Authenticating...

Enter you email: ram@abc.xyz

Enter your password: password1

Welcome thulasi...

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: thulasi ram

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): update

Enter ID of employee to update:

100

Enter the field to update (firstName, lastName, email):

lastName

Enter the field value:

Sangakara

Enter your action(add/update/remove/display/displayAll): display

Enter Employee ID: 100

Full Name: hari Sangakara

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): add

Enter ID: 102

Enter first name: new

Enter last name: user

Enter the salary: 40000

Enter email ID: newuser@abc.xyz

Enter your password: password3

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name:

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari Priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Full Name: new user

ID: 102

email: newuser@abc.xyz

Salary: 40000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): remove

Enter ID of employee to remove:

102

Record removed successfully

Enter your action(add/update/remove/display/displayAll): displayAll

Full Name: ram sager

ID: 99

email: ram@abc.xyz

Salary: 46000.0

Admin: true

----

Full Name: hari priya

ID: 100

email: kumar@abc.xyz

Salary: 45000.0

Admin: false

----

Enter your action(add/update/remove/display/displayAll): exit