**ASSIGNMENT -1:Constructor, Inheritance and Interface**

1. Design a class named "Product" with **default, and parameterized constructors**.
   1. Default constructor initializes default values
   2. the parameterized constructor allows custom initialization

Implement methods to display product details. Test the class by creating instances using each constructor to make sure everything works correctly and data is handled properly.

SOURCE CODE-

package aa;

public class Product {

String productName;

double price;

// Default constructor

public Product() {

this.productName = "Dell laptop";

this.price = 80000;

}

// Parameterized constructor

public Product(String productName, double price) {

this.productName = productName;

this.price = price;

}

public String getProductName() {

return productName;

}

public double getPrice() {

return price;

}

public void setProductName(String productName) {

this.productName = productName;

}

public void setPrice(double price) {

this.price = price;

}

public static void main(String[] args) {

Product defaultProduct = new Product();

System.out.println("Default Product Name: " + defaultProduct.getProductName());

System.out.println("Default Product Price: Rs." + defaultProduct.getPrice());

Product customProduct = new Product("Custom laptop", 100000);

System.out.println("Custom Product Name: " + customProduct.getProductName());

System.out.println("Custom Product Price: Rs." + customProduct.getPrice());

customProduct.setPrice(110000);

System.out.println("Updated Custom Product Price: Rs." + customProduct.getPrice());

}

}

package m1;

public class Constructor {

String firstname,lastname;

int age;

public Constructor() { //default

firstname="Abc";

lastname="Def";

age=19;

}

public Constructor(String firstname, String lastname, int age)//parameterised

{

this.firstname=firstname;

this.lastname=lastname;

this.age=age;

}

public static void main(String[] args) {

Constructor obj1 = new Constructor();

Constructor obj2 = new Constructor("Mno", "Pqr", 25);

System.***out***.println(obj1.firstname);

System.***out***.println(obj1.lastname);

System.***out***.println(obj1.age);

System.***out***.println(obj2.firstname);

System.***out***.println(obj2.lastname);

System.***out***.println(obj2.age);

}

}

//Constructor

**------------------------------------------------------------------------------------------------**

1. Create a base class Person with attributes name and age. Include a method display\_info() to print basic information about the person.
   1. **Single Inheritance:**

Derive a class Student from the Person class with additional attributes student\_id and major. Include a method enroll\_course() to simulate a student enrolling in a course.

* 1. **Multilevel Inheritance:**

Extend the Student class to create a new class GraduateStudent with an additional attribute research\_topic. Implement a method conduct\_research() to simulate a graduate student conducting research.

* 1. **Hierarchical Inheritance:**

Derive a class Professor from the Person class with attributes like employee\_id, department, and teaching\_subject. Implement a method conduct\_lecture() to simulate a professor conducting a lecture.

SOURCE CODE-

package per;

class Person {

protected String name;

protected int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public void display\_info() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

public static void main(String[] args) {

// Single Example

Student student = new Student("Mariyam", 20, 12345, "Computer Science");

student.display\_info();

student.enroll\_course();

System.out.println();

// Multilevel Example

GraduateStudent gradStudent = new GraduateStudent("Ayesha", 25, 54321, "Physics", "Quantum Mechanics");

gradStudent.display\_info();

gradStudent.enroll\_course();

gradStudent.conduct\_research();

System.out.println();

// Hierarchical Example

Professor professor = new Professor("John", 40, 98765, "Mathematics", "Calculus");

professor.display\_info();

professor.conduct\_lecture();

}

}

//Single Inheritance

class Student extends Person {

private int student\_id;

private String major;

public Student(String name, int age, int student\_id, String major) {

super(name, age);

this.student\_id = student\_id;

this.major = major;

}

public void enroll\_course() {

System.out.println(name + " is enrolled in a course.");

}

}

//Multilevel Inheritance

class GraduateStudent extends Student {

private String research\_topic;

public GraduateStudent(String name, int age, int student\_id, String major, String research\_topic) {

super(name, age, student\_id, major);

this.research\_topic = research\_topic;

}

public void conduct\_research() {

System.out.println(name + " is conducting research on " + research\_topic);

}

}

//Hierarchical Inheritance

class Professor extends Person {

private int employee\_id;

private String department;

private String teaching\_subject;

public Professor(String name, int age, int employee\_id, String department, String teaching\_subject) {

super(name, age);

this.employee\_id = employee\_id;

this.department = department;

this.teaching\_subject = teaching\_subject;

}

public void conduct\_lecture() {

System.out.println(name + " is conducting a lecture on " + teaching\_subject);

}

}

package m2;

class Base {

String firstName = "Shreyash";

}

class Child1 extends Base{

String lastName = "Waghmode";

}

class Child2 extends Child1{

int rollNo = 48;

}

class Child3 extends Base{

String role = "Student";

}

public class Inheritance {

public static void main(String a[])

{

Child2 obj = new Child2();

Child3 obj2 = new Child3();

System.***out***.println(obj.firstName);//Single

System.***out***.println(obj.lastName);//Single

System.***out***.println(obj.rollNo);//Multilevel

System.***out***.println(obj2.firstName +" is a " + obj2.role);//Hierarchical

}

}

//Inheritance

**------------------------------------------------------------------------------------------------**

1. **Multiple inheritance using Interface:**

Define two interfaces: CardioWorkout and StrengthTraining.

* 1. In the CardioWorkout interface, declare methods such as getWorkoutName(), getDurationMinutes(), and displayWorkoutDetails().
  2. In the StrengthTraining interface, declare methods like getEquipmentUsed(), getInstructorName(), and displayStrengthTrainingDetails().

Derived Class:

Create a class named FullBodyWorkout that implements both Cardio\_Workout and StrengthTraining. This class represents a workout program that combines cardio exercises with strength training for a comprehensive full-body workout.

SOURCE CODE-

package ab;

public interface CardioWorkout {

String getWorkoutName();

int getDurationMinutes();

void displayWorkoutDetails();

}

public interface StrengthTraining {

String getEquipmentUsed();

String getInstructorName();

void displayStrengthTrainingDetails();

}

public class FullBodyWorkout implements CardioWorkout, StrengthTraining {

private String workoutName;

private int durationMinutes;

private String equipmentUsed;

private String instructorName;

public FullBodyWorkout(String workoutName, int durationMinutes, String equipmentUsed, String instructorName) {

this.workoutName = workoutName;

this.durationMinutes = durationMinutes;

this.equipmentUsed = equipmentUsed;

this.instructorName = instructorName;

}

@Override

public String getWorkoutName() {

return workoutName;

}

@Override

public int getDurationMinutes() {

return durationMinutes;

}

@Override

public void displayWorkoutDetails() {

System.out.println("Cardio Workout Details:");

System.out.println("Workout Name: " + getWorkoutName());

System.out.println("Duration: " + getDurationMinutes() + " minutes");

}

@Override

public String getEquipmentUsed() {

return equipmentUsed;

}

@Override

public String getInstructorName() {

return instructorName;

}

@Override

public void displayStrengthTrainingDetails() {

System.out.println("\nStrength Training Details:");

System.out.println("Equipment Used: " + getEquipmentUsed());

System.out.println("Instructor Name: " + getInstructorName());

}

public static void main(String[] args) {

FullBodyWorkout fullBodyWorkout = new FullBodyWorkout("Full Body Circuit", 60, "Dumbbells", "Fitness Trainer");

fullBodyWorkout.displayWorkoutDetails();

fullBodyWorkout.displayStrengthTrainingDetails();

}

}

package m6;

interface Animal{

public void animalSound();

public void sleep();

}

class Cat implements Animal{

public void animalSound() {

System.***out***.println("the cat says:meow");

}

public void sleep() {

System.***out***.println("zzzzzzzzzzzzzzzzzzzz");

}

}

class Interface{

public static void main(String[] args) {

Cat c=new Cat();

c.animalSound();

c.sleep();

}

}

**------------------------------------------------------------------------------------------------**

**ASSIGNMENT -2 : Polymorphism,Abstraction and Encapsulation**

1. Create a base class Account featuring methods deposit() and withdraw(). These methods should be overridden by derived classes to execute specific actions tailored to different account types.
   1. **Method Overloading:**

Derive a class SavingsAccount from Account. Implement method overloading for withdraw() by introducing an additional parameter to specify the reason for withdrawal.

* 1. **Method Overriding**:

Derive a class CheckingAccount from Account. Implement method overriding for withdraw() within the CheckingAccount class, introducing overdraft protection.

* 1. **Polymorphic Demonstration:**
     1. Instantiate objects for both SavingsAccount and CheckingAccount.
     2. Illustrate polymorphism by invoking the withdraw() method on both objects, highlighting distinct behaviors based on the account type.

SOURCE CODE-

package ac;

public class Account {

protected double balance;

public Account(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: $" + amount);

displayBalance();

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: $" + amount);

} else {

System.out.println("Insufficient funds");

}

displayBalance();

}

public void displayBalance() {

System.out.println("Current Balance: $" + balance);

}

public static void main(String[] args) {

SavingsAccount savingsAccount = new SavingsAccount(1000);

savingsAccount.withdraw(200, "Education");

CheckingAccount checkingAccount = new CheckingAccount(1500, 500);

checkingAccount.withdraw(200);

Account account1 = savingsAccount;

Account account2 = checkingAccount;

account1.withdraw(100);

account2.withdraw(300);

}

}

class SavingsAccount extends Account {

public SavingsAccount(double initialBalance) {

super(initialBalance);

}

public void withdraw(double amount, String reason) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: $" + amount + " for " + reason);

} else {

System.out.println("Insufficient funds");

}

displayBalance();

}

}

class CheckingAccount extends Account {

private double overdraftLimit;

public CheckingAccount(double initialBalance, double overdraftLimit) {

super(initialBalance);

this.overdraftLimit = overdraftLimit;

}

@Override

public void withdraw(double amount) {

if (amount <= balance + overdraftLimit) {

balance -= amount;

System.out.println("Withdrawn: $" + amount);

} else {

System.out.println("Exceeds overdraft limit. Transaction declined.");

}

displayBalance();

}

}

package m3;

class Animal {

public void makeSound() {

System.***out***.println("Some generic sound");

}

}

class Dog extends Animal {

*@Override*

public void makeSound() {

System.***out***.println("Bark! Bark!");

}

public void makeSound(int times) { // Overloaded method

for (int i = 0; i < times; i++) {

System.***out***.println("Bark!"); }

}

}

class Cat extends Animal {

*@Override*

public void makeSound() {

System.***out***.println("Meow! Meow!");

}

public void makeSound(String emotion) {// Overloaded method

System.***out***.println("Purr... Feeling " + emotion);

}

}

public class Polymorphism {

public static void main(String[] args) {

Animal myDog = new Dog();

Animal myCat = new Cat();

myDog.makeSound();

myCat.makeSound();

((Dog) myDog).makeSound(3);

((Cat) myCat).makeSound("happy");

}

}

//Polymorphism,override,overloading

**------------------------------------------------------------------------------------------------**

1. Write a Java program in which you will declare an **abstract class** Vehicle inherits this class from two classes car and truck using the method engine in both displays “car has good engine” and “truck has bad engine”.

SOURCE CODE-

package ad;

public abstract class Vehicle {

public abstract void engine();

public static void main(String[] args) {

Car myCar = new Car();

Truck myTruck = new Truck();

myCar.engine();

myTruck.engine();

}

}

class Car extends Vehicle {

@Override

public void engine() {

System.out.println("Car has a good engine.");

}

}

class Truck extends Vehicle {

@Override

public void engine() {

System.out.println("Truck has a bad engine.");

}

}

package m4;

abstract class gfg {

abstract void printInfo();

}

class employee extends gfg {

void printInfo()

{

String name = "Yashwant";

int age = 19;

float salary = 55552.2F;

System.***out***.println(name);

System.***out***.println(age);

System.***out***.println(salary);

}

}

class Abstraction {

public static void main(String args[])

{

gfg s = new employee();

s.printInfo();

}

}

//Abstraction

**------------------------------------------------------------------------------------------------**

1. Develop an Employee Management System and demonstrate the concept of **encapsulation.** Create a class named Employee with private attributes for employeeId, firstName, lastName, dateOfBirth, and salary. Implement accessor methods for retrieving attribute values and mutator methods for modifying attributes, ensuring proper validation. Additionally, include a method calculateYearlyBonus() within the Employee class that utilizes encapsulated data to calculate a yearly bonus based on a predetermined formula.

SOURCE CODE-

package ae;

public class Employee {

private int employeeId;

private String firstName;

private String lastName;

private double salary;

public Employee(int employeeId, String firstName, String lastName, double salary) {

this.employeeId = employeeId;

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

}

public int getEmployeeId() {

return employeeId;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public double getSalary() {

return salary;

}

public void setFirstName(String firstName) {

if (firstName != null && !firstName.trim().isEmpty()) {

this.firstName = firstName;

} else {

System.out.println("Invalid first name. Please provide a non-empty value.");

}

}

public void setLastName(String lastName) {

if (lastName != null && !lastName.trim().isEmpty()) {

this.lastName = lastName;

} else {

System.out.println("Invalid last name. Please provide a non-empty value.");

}

}

public void setSalary(double salary) {

if (salary >= 0) {

this.salary = salary;

} else {

System.out.println("Invalid salary. Please provide a non-negative value.");

}

}

public double calculateYearlyBonus() {

return 0.1 \* salary;

}

public void displayEmployeeInfo() {

System.out.println("Employee ID: " + employeeId);

System.out.println("Name: " + firstName + " " + lastName);

System.out.println("Salary: $" + salary);

System.out.println("Yearly Bonus: $" + calculateYearlyBonus());

}

public static void main(String[] args) {

Employee employee = new Employee(101, "John", "Doe", 50000.0);

// Display employee information

System.out.println("Before modifications:");

employee.displayEmployeeInfo();

employee.setFirstName("Jane");

employee.setLastName("Smith");

employee.setSalary(60000.0);

System.out.println("\nAfter modifications:");

employee.displayEmployeeInfo();

}

}

package m5;

class Encapsulation {

private String name;

private int age;

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 static void main(String[] args)

{

Encapsulation person = new Encapsulation();

person.setName("Yashwant");

person.setAge(19);

System.***out***.println("Name: " + person.getName());

System.***out***.println("Age: " + person.getAge());

}

}

//Encapsulation

**------------------------------------------------------------------------------------------------**

**ASSIGNMENT-3**

**Create a Java class named "Reservation" to manage hotel room reservations. Develop methods to the reservation details from the user and print it.  
a) Implement a try-and-catch block to handle potential errors that may arise during reservation initialization and display, such as negative guest numbers or null reservation dates. Print relevant error messages.**

**b) Use multiple catch statements to handle distinct types of exceptions, like negative room numbers or null guest names. Handle these exceptions separately during both initialization and display operations.**

**c) Introduce a finally block to ensure cleanup operations, guaranteeing proper resource management regardless of whether an exception occurred. Verify the class by testing instances created using each constructor to confirm correct exception handling and resource cleanup in a hotel room reservation system.**

package ass3;

import java.util.Date;

public class Reservation {

private int roomNumber;

private String guestName;

private Date reservationDate;

// Constructor for initializing the reservation

public Reservation(int roomNumber, String guestName, Date reservationDate) {

try {

if (roomNumber <= 0) {

throw new IllegalArgumentException("Room number must be a positive integer.");

}

if (guestName == null || guestName.trim().isEmpty()) {

throw new IllegalArgumentException("Guest name cannot be null or empty.");

}

if (reservationDate == null) {

throw new IllegalArgumentException("Reservation date cannot be null.");

}

this.roomNumber = roomNumber;

this.guestName = guestName;

this.reservationDate = reservationDate;

} catch (IllegalArgumentException e) {

System.err.println("Error during reservation initialization: " + e.getMessage());

} finally {

// Cleanup operations (if needed) can be placed here

}

}

// Method to display reservation details

public void displayReservationDetails() {

try {

if (roomNumber <= 0) {

throw new IllegalArgumentException("Room number must be a positive integer.");

}

if (guestName == null || guestName.trim().isEmpty()) {

throw new IllegalArgumentException("Guest name cannot be null or empty.");

}

if (reservationDate == null) {

throw new IllegalArgumentException("Reservation date cannot be null.");

}

System.out.println("Room Number: " + roomNumber);

System.out.println("Guest Name: " + guestName);

System.out.println("Reservation Date: " + reservationDate);

} catch (IllegalArgumentException e) {

System.err.println("Error during displaying reservation details: " + e.getMessage());

} finally {

// Cleanup operations (if needed) can be placed here

}

}

public static void main(String[] args) {

// Testing the class with valid and invalid inputs

Reservation validReservation = new Reservation(101, "John Doe", new Date());

validReservation.displayReservationDetails();

System.out.println("\nTesting with invalid inputs:");

Reservation invalidReservation1 = new Reservation(0, "Jane Doe", new Date());

Reservation invalidReservation2 = new Reservation(102, "", new Date());

Reservation invalidReservation3 = new Reservation(103, "Bob Smith", null);

}

}

package m7;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class Tcmc {

private int roomNumber;

private String guestName;

private Date reservationDate;

public Tcmc() {

try {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter room number: ");

int roomNumberInput = scanner.nextInt();

if (roomNumberInput <= 0) {

throw new Exception("Room number must be positive.");

}

this.roomNumber = roomNumberInput;

scanner.nextLine();

System.***out***.print("Enter guest name: ");

String guestNameInput = scanner.nextLine();

if (guestNameInput == null || guestNameInput.isEmpty()) {

throw new Exception("Guest name cannot be null or empty.");

}

this.guestName = guestNameInput;

System.***out***.print("Enter reservation date (yyyy-mm-dd): ");

String dateInput = scanner.nextLine();

if (dateInput == null || dateInput.isEmpty()) {

throw new Exception("Reservation date cannot be null or empty.");

}

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

this.reservationDate = dateFormat.parse(dateInput);

} catch (Exception e) {

System.***err***.println("Error initializing reservation: " + e.getMessage());

}

}

public void displayReservationDetails() {

try {

System.***out***.println("Room Number: " + roomNumber);

System.***out***.println("Guest Name: " + guestName);

System.***out***.println("Reservation Date: " + reservationDate);

} catch (NullPointerException e) {

System.***err***.println("Error displaying reservation details: " + e.getMessage());

}

}

public static void main(String[] args) {

try {

Tcmc reservation = new Tcmc();

reservation.displayReservationDetails();

} catch (Exception e) {

System.***err***.println("Exception occurred: " + e.getMessage());

}

}

}

//Tcmc

**------------------------------------------------------------------------------------------------**

**ASSIGNMENT-4**

**Develop a simple attendance management system using multithreading in Java. There should be a system that maintains attendance records for students in different classes. Create two classes - ThreadAttendance and RunnableAttendance - to implement the attendance system.**

**a. ThreadAttendance Class**

**- Create a ThreadAttendance class that extends the Thread class.**

**- Each thread represents the attendance of a specific class.**

**- The attendance should be marked by printing the class name and the names of students present for each day.**

**b. RunnableAttendance Class:**

**- Create a RunnableAttendance class that implements the Runnable interface.**

**- Each RunnableAttendance instance represents a student.**

**- The run method should simulate a student arriving late by printing a message indicating the student's name and lateness.**

**c. Main Class**

**- In the main class, create instances of ThreadAttendance for different classes and start them.**

**- Create instances of RunnableAttendance for different students and execute them using a single thread.**

**SOURCE CODE-**

package ass4;

class ThreadAttendance extends Thread { //ThreadAttendance class

private String className;

public ThreadAttendance(String className) {

this.className = className;

}

@Override

public void run() {

for (int day = 1; day <= 5; day++) {

System.out.println("Class: " + className + " - Day " + day);

for (int i = 0; i < 5; i++) {

System.out.println(" Student " + (i + 1) + " is present");

}

System.out.println();

try {

Thread.sleep(1000);

} catch (Exception e) {

e.printStackTrace();

}

}

}

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

ThreadAttendance classAttendance = new ThreadAttendance("Class C");

classAttendance.start();

RunnableAttendance student1 = new RunnableAttendance("aa");

RunnableAttendance student2 = new RunnableAttendance("bb");

RunnableAttendance student3 = new RunnableAttendance("cc");

RunnableAttendance student4 = new RunnableAttendance("dd");

RunnableAttendance student5 = new RunnableAttendance("ee");

new Thread(student1).start();

new Thread(student2).start();

new Thread(student3).start();

new Thread(student4).start();

new Thread(student5).start();

}

}

class RunnableAttendance implements Runnable { //RunnableAttendance class

private String studentName;

public RunnableAttendance(String studentName) {

this.studentName = studentName;

}

@Override

public void run() {

System.out.println("Student " + studentName + " is late");

}

}

**------------------------------------------------------------------------------------------------**

**ASSIGNMENT-5**

**PROBLEM STATEMENT-**

Design a Java program to manage course enrollment details using an ArrayList. Implement the following functionalities using built-in methods(such as add(), set(), size(), remove(), clear(),contains(),isEmpty()) and user input:  
a. Include course\_id, course name, students enrolled, student\_name, and student\_id.  
b.Use built-in methods to add this entry to the ArrayList.Display a success message after adding the entry.

**SOURCE CODE-**

package ass5;

import java.util.ArrayList;

import java.util.Scanner;

class Course {

private String courseId;

private String courseName;

private ArrayList<String[]> students;

public Course(String courseId, String courseName) {

this.courseId = courseId;

this.courseName = courseName;

this.students = new ArrayList<>();

}

public void enrollStudent(String studentName, String studentId) {

String[] student = {studentName, studentId};

students.add(student);

}

public boolean removeStudent(String studentId) {

for (int i = 0; i < students.size(); i++) {

if (students.get(i)[1].equals(studentId)) {

students.remove(i);

return true;

}

}

return false;

}

public boolean containsStudent(String studentId) {

for (String[] student : students) {

if (student[1].equals(studentId)) {

return true;

}

}

return false;

}

public void clearStudents() {

students.clear();

}

public int getStudentCount() {

return students.size();

}

public boolean isEmpty() {

return students.isEmpty();

}

public String getCourseId() {

return courseId;

}

public String getCourseName() {

return courseName;

}

public ArrayList<String[]> getStudents() {

return students;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String courseId = scanner.nextLine();

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

String courseName = scanner.nextLine();

Course course = new Course(courseId, courseName);

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

String studentName = scanner.nextLine();

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

String studentId = scanner.nextLine();

course.enrollStudent(studentName, studentId);

System.out.println("Student enrolled successfully in the course: " + course.getCourseName());

System.out.println("Course ID: " + course.getCourseId());

System.out.println("Enrolled Students: ");

ArrayList<String[]> students = course.getStudents();

for (String[] student : students) {

System.out.println("Student Name: " + student[0] + ", Student ID: " + student[1]);

}

System.out.println("Size of enrolled students: " + course.getStudentCount());

System.out.println("Is the course empty? " + course.isEmpty());

System.out.print("Enter student ID to remove: ");

String studentIdToRemove = scanner.nextLine();

if (course.containsStudent(studentIdToRemove)) {

course.removeStudent(studentIdToRemove);

System.out.println("Student with ID " + studentIdToRemove + " removed successfully.");

} else {

System.out.println("Student with ID " + studentIdToRemove + " not found.");

}

System.out.println("Size of enrolled students after removal: " + course.getStudentCount());

course.clearStudents();

System.out.println("Enrolled students cleared.");

System.out.println("Is the course empty now? " + course.isEmpty());

scanner.close();

}

}

2nd code for collections:

package e1;

import java.util.ArrayList;

import java.util.Scanner;

class Collection {

public static void main(String args[]) {

ArrayList<String> al = new ArrayList<String>();

Scanner scanner = new Scanner(System.in);

System.out.println("Initial size of ArrayList: " + al.size());

// Add elements to the ArrayList

al.add("1");

al.add("2");

al.add("3");

al.add("4");

al.add("5");

al.add("6");

System.out.println("Size of ArrayList before additions: " + al.size());

System.out.println("Contents of ArrayList: " + al);

// Add an element at a specific index

al.add(1, "12");

System.out.println("Size of ArrayList after additions: " + al.size());

System.out.println("Contents of ArrayList: " + al);

// Remove elements from the ArrayList

al.remove("3");

al.remove(2);

System.out.println("Size of ArrayList after deletions: " + al.size());

System.out.println("Contents of ArrayList: " + al);

// Set an element at a specific index

System.out.print("Enter an element to set at index 2: ");

String elementToSet = scanner.nextLine();

al.set(2, elementToSet);

System.out.println("Contents of ArrayList after set operation: " + al);

// Check if an element exists in the ArrayList

System.out.print("Enter an element to check if it exists: ");

String elementToCheck = scanner.nextLine();

boolean contains = al.contains(elementToCheck);

System.out.println("ArrayList contains '" + elementToCheck + "': " + contains);

// Clear the ArrayList

al.clear();

System.out.println("Size of ArrayList after clear operation: " + al.size());

// Check if the ArrayList is empty

boolean isEmpty = al.isEmpty();

System.out.println("ArrayList is empty: " + isEmpty);

scanner.close();

}

}

//Collections

6.JDBC

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class JDBCExample {

// JDBC URL, username and password of MySQL server

private static final String URL = "jdbc:mysql://localhost:3306/testdb";

private static final String USER = "root";

private static final String PASSWORD = "09Sukhada1975";

public static void main(String[] args) {

Connection connection = null;

Statement statement = null;

try {

// Load the MySQL JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Establish the connection

connection = DriverManager.getConnection(URL, USER, PASSWORD);

// Create a statement object

statement = connection.createStatement();

// Create a new table

String createTableSQL = "CREATE TABLE IF NOT EXISTS users ("

+ "id INT(11) NOT NULL AUTO\_INCREMENT, "

+ "name VARCHAR(50) NOT NULL, "

+ "email VARCHAR(50) NOT NULL, "

+ "PRIMARY KEY (id))";

statement.execute(createTableSQL);

// Insert some data into the table

String insertDataSQL = "INSERT INTO users (name, email) VALUES "

+ "('John Doe', 'john.doe@example.com'), "

+ "('Jane Smith', 'jane.smith@example.com')";

statement.executeUpdate(insertDataSQL);

// Retrieve data from the table

String selectSQL = "SELECT \* FROM users";

ResultSet resultSet = statement.executeQuery(selectSQL);

// Display the retrieved data

while (resultSet.next()) {

int id = resultSet.getInt("id");

String name = resultSet.getString("name");

String email = resultSet.getString("email");

System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email);

}

// Clean up

resultSet.close();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

if (statement != null) statement.close();

if (connection != null) connection.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

Q.EXAMPLE OF A JDBC (EMPLOYEE):

Main.java-

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        String name;

        int id;

        EmployeeDaoIntrf dao = new EmployeeDaoImpl();

        System.out.println("Welcome to Employee management application");

        Scanner sc = new Scanner(System.in);

        try {

            do {

                System.out.println("1. Add Employee\n" +

                        "2. Show All Employee\n" +

                        "3. Show Employee based on id \n" +

                        "4. Update the employee\n" +

                        "5. Delete the employee\n");

                System.out.println("Enter Choice: ");

                int ch = sc.nextInt();

                switch (ch) {

                    case 1:

                        Employee emp = new Employee();

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

                        id = sc.nextInt();

                        System.out.println("Enter name ");

                        name = sc.next();

                        System.out.println("Enter Salary ");

                        double salary = sc.nextDouble();

                        System.out.println("Enter age");

                        int age = sc.nextInt();

                        emp.setId(id);

                        emp.setName(name);

                        emp.setSalary(salary);

                        emp.setAge(age);

                        dao.createEmployee(emp);

                        break;

                    case 2:

                        dao.showAllEmployee();

                        break;

                    case 3:

                        System.out.println("Enter id to show the details ");

                        int empid = sc.nextInt();

                        dao.showEmployeeBasedOnID(empid);

                        break;

                    case 4:

                        System.out.println("Enter id to update the details");

                        int empid1 = sc.nextInt();

                        System.out.println("Enter the new name");

                        name = sc.next();

                        dao.updateEmployee(empid1, name);

                        break;

                    case 5:

                        System.out.println("Enter the id to delete");

                        id = sc.nextInt();

                        dao.deleteEmployee(id);

                        break;

                    case 6:

                        System.out.println("Thank you for using our Application !!!");

                        return;

                    default:

                        System.out.println("Enter valid choice !");

                        break;

                }

            } while (true);

        } finally {

            sc.close(); // Close the Scanner

        }

    }

}

Employee.java-

public class Employee {

    private int id;

    private String name;

    private double salary;

    private int age;

    public Employee(){

    }

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

      this.id = id;

      this.name = name;

      this.salary = salary;

      this.age = age;

  }

    public int getId() {

      return id;

  }

  public void setId(int id) {

      this.id = id;

  }

  public String getName() {

      return name;

  }

  public void setName(String name) {

      this.name = name;

  }

  public double getSalary() {

      return salary;

  }

  public void setSalary(double salary) {

      this.salary = salary;

  }

  public int getAge() {

      return age;

  }

  public void setAge(int age) {

      this.age = age;

  }

  @Override

  public String toString() {

      return "Employee{" +

              "id=" + id +

              ", name='" + name + '\'' +

              ", salary=" + salary +

              ", age=" + age +

              '}';

  }

  }

EmployeeDaoIntrf.java-

public interface EmployeeDaoIntrf {

    //create employee

    public void createEmployee(Employee emp);

    //show all employee

    public void showAllEmployee();

    //show employee based on id

    public void showEmployeeBasedOnID(int id);

    //update employee

    public void updateEmployee(int id,String name);

    //delete employee

    public void deleteEmployee(int id);

}

EmployeeDaoImpl.java-

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

public class EmployeeDaoImpl implements EmployeeDaoIntrf {

    Connection con;

    @Override

    public void createEmployee(Employee emp) {

       con =DBConnection.createDBConnetion();

       String query="insert into employee values(?,?,?,?)";

       try{

           PreparedStatement pstm=con.prepareStatement(query);

           pstm.setInt(1,emp.getId());

           pstm.setString(2,emp.getName());

           pstm.setDouble(3,emp.getSalary());

           pstm.setInt(4,emp.getAge());

          int cnt= pstm.executeUpdate();

          if(cnt!=0)

              System.out.println("Employee Inserted Successfully !!!");

       }catch (Exception ex){

           ex.printStackTrace();

       }

    }

    @Override

    public void showAllEmployee() {

        con=DBConnection.createDBConnetion();

        String query="select \* from employee";

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

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

        System.out.format("%s\t%s\t%s\t%s\n","ID","Name","Salary","age");

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

        try{

            Statement stmt=con.createStatement();

            ResultSet result= stmt.executeQuery(query);

            while (result.next()){

                System.out.format("%d\t%s\t%f\t%d\n",

                        result.getInt(1),

                        result.getString(2),

                        result.getDouble(3),

                        result.getInt(4));

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

            }

        }catch (Exception ex){

            ex.printStackTrace();

        }

    }

    @Override

    public void showEmployeeBasedOnID(int id) {

        con=DBConnection.createDBConnetion();

        String query="select \* from employee where id="+id;

        try{

            Statement stmt=con.createStatement();

           ResultSet result= stmt.executeQuery(query);

            while (result.next()){

                System.out.format("%d\t%s\t%f\t%d\n",

                        result.getInt(1),

                        result.getString(2),

                        result.getDouble(3),

                        result.getInt(4));

            }

        }

        catch (Exception ex){

            ex.printStackTrace();

        }

    }

    @Override

    public void updateEmployee(int id, String name) {

        con=DBConnection.createDBConnetion();

        String query="update employee set name=? where id=?";

        try{

            PreparedStatement pstm=con.prepareStatement(query);

            pstm.setString(1,name);

            pstm.setInt(2,id);

            int cnt=pstm.executeUpdate();

            if(cnt!=0)

                System.out.println("Employee Details updated successfully !!");

        }catch (Exception ex){

            ex.printStackTrace();

        }

    }

    @Override

    public void deleteEmployee(int id) {

        con=DBConnection.createDBConnetion();

        String query="delete from employee where id=?";

        try{

            PreparedStatement pstm=con.prepareStatement(query);

            pstm.setInt(1,id);

           int cnt= pstm.executeUpdate();

           if(cnt!=0)

               System.out.println("Employee Deleted Successfully!!! "+id);

        }catch (Exception ex){

            ex.printStackTrace();

        }

    }

}

DBConnection.java-

import java.sql.Connection;

import java.sql.DriverManager;

public class DBConnection {

    static Connection con;

    public static Connection createDBConnetion(){

        try{

            //load driver

            Class.forName("com.mysql.jdbc.Driver");

            //get connection

            String url="jdbc:mysql://localhost:3306/proDB?useSSL=false";

            String username="root";

            String password="root";

           con= DriverManager.getConnection(url,username,password);

        }catch (Exception ex){

            ex.printStackTrace();

        }

     return con;

    }

}