Constructor,

        package Assignment1;

        class product{

            private String name;

            private double price;

            private int quantity;

            public product() {

                this.name="Atharva";

                this.price=90.0;

                this.quantity=30;

            }

            public product(String name, double price, int quantity) {

                this.name=name;

                this.price=price;

                this.quantity=quantity;

            }

            public void display() {

                System.out.println("name= " + name);

                System.out.println("price= "+ price);

                System.out.println("quantity= "+ quantity);

            }

        }

        public class constructor {

            public static void main(String[] args) {

                // TODO Auto-generated method stub

                product defaultproduct = new product();

                System.out.println("Default");

                defaultproduct.display();

                product define = new product("hii", 12.1, 13);

                System.out.println("Default");

                define.display();

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Inheritance

        package Assignment1;

        class Person{

            String name;

            int age;

            public Person(String name, int age) {

                this.name=name;

                this.age=age;

            }

            public void displayinfo() {

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

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

            }

        }

        class Student extends Person{

            int student\_id;

            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(String course)  {

                System.out.println("course : "+ course);

            }

        }

        class GraduateStudent extends Student{

            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("research : "+ research\_topic);

            }

        }

        class Professor extends Person {

            int employee\_id;

            String department;

            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("conduct\_lecture : "+ teaching\_subject);

            }

        }

        public class inheritance {

            public static void main(String[] args) {

                // TODO Auto-generated method stub

                Student student = new Student("atharva", 20 ,73, "cs");

                student.displayinfo();

                student.enroll\_course("Java");

                GraduateStudent graduateStudent = new GraduateStudent("gaikwad", 30 ,23, "aa", "python");

                graduateStudent.displayinfo();

                graduateStudent.conduct\_research();

                Professor professor = new Professor("hii", 380 ,283, "uuu", "matha");

                professor.displayinfo();

                professor.conduct\_lecture();

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Interface

            package Assignment1;

            interface CardioWorkout {

                String getWorkoutName();

                int getDurationMinutes();

                void displayWorkoutDetails();

            }

            interface StrengthTraining {

                int getEquipmentUsed();

                String getInstructorName();

                void displayStrengthTrainingDetails();

            }

            class FullBodyWorkout implements CardioWorkout, StrengthTraining{

                private String workoutName;

                private int workoutDuration;

                private int equipmentUsed;

                private String instructorName;

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

                    this.workoutName=workoutName;

                    this.workoutDuration=workoutDuration;

                    this.equipmentUsed=equipmentUsed;

                    this.instructorName=instructorName;

                }

                @Override

                public String getWorkoutName() {

                    return workoutName;

                }

                @Override

                public int getDurationMinutes() {

                    return workoutDuration;

                }

                @Override

                public void displayWorkoutDetails() {

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

                    System.out.println("workoutName : "+workoutName);

                    System.out.println("workoutDuration : "+workoutDuration);

                }

                @Override

                public int getEquipmentUsed() {

                    return equipmentUsed;

                }

                @Override

                public String getInstructorName() {

                    return instructorName;

                }

                @Override

                public void displayStrengthTrainingDetails() {

                    System.out.println("Strength Details");

                    System.out.println("equipmentUsed : "+equipmentUsed);

                    System.out.println("instructorName : "+instructorName);

                }

                public void DisplayFullbodyDetails() {

                    displayWorkoutDetails();

                    displayStrengthTrainingDetails();

                }

            }

            public class interfac {

                public static void main(String[] args) {

                    // TODO Auto-generated method stub

                    FullBodyWorkout fullBodyWorkout = new FullBodyWorkout("mussle" , 60, 3, "atharva");

                    fullBodyWorkout.DisplayFullbodyDetails();

                }

            }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Polymorphism,

        package Assignment2;

        abstract class Account {

            protected double balance;

            public Account(double balance) {

                this.balance = balance;

            }

            public void deposit (double amount) {

                if (amount > 0) {

                    balance += amount;

                    System.out.println("Deposite : " + amount);

                } else {

                    System.out.println("enter valid amount");

                }

            }

            public abstract void withdraw(double amount);

        }

        class SavingsAccount extends Account {

            public SavingsAccount(double balance) {

                super(balance);

            }

            @Override

            public void withdraw(double amount) {

                if (amount >0 && amount <= balance ) {

                    balance -= amount;

                    System.out.println("Withdraw : " + amount);

                } else {

                    System.out.println("enter valid amount");

                }

            }

            //overloadding

            public void withdraw(double amount, String reason) {

                if (amount >0 && amount <= balance ) {

                    balance -= amount;

                    System.out.println("Withdraw : " + amount + reason);

                } else {

                    System.out.println("enter valid amount or " + reason);

                }

            }

        }

        class CheckingAccount extends Account {

            private double overdraft;

            public CheckingAccount(double balance, double overdraft) {

                super(balance);

                this.overdraft= overdraft;

            }

            @Override

            public void withdraw(double amount) {

                if (amount >0 && (balance + overdraft) >= amount ) {

                    balance -= amount;

                    System.out.println("Withdraw : " + amount);

                    if (balance < 0) {

                        System.out.println("overdraft applied " + balance);

                    }

                }else {

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

                }

            }

        }

        public class Polymorphism {

            public static void main(String[] args) {

                // TODO Auto-generated method stub

                Account Saving = new SavingsAccount(500);

                Account Checking = new CheckingAccount (500 ,200);

                System.out.println("Saving account");

                Saving.withdraw(100);

                ((SavingsAccount)Saving).withdraw(200, "hii");

                System.out.println("Checking account");

                Checking.withdraw(600);

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Abstraction

        package Assignment2;

        abstract class vehicle{

            public abstract void engine();

        }

        class car extends vehicle{

            @Override

            public void engine() {

                System.out.println("car has good engine");

            }

        }

        class truck  extends vehicle{

            @Override

            public void engine() {

                System.out.println("truck has good engine");

            }

        }

        public class main {

            public static void main(String[] args) {

                // TODO Auto-generated method stub

                vehicle Car = new car();

                vehicle Truck = new truck();

                Car.engine();

                Truck.engine();

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Encapsulation

        package Assignment2;

        class Employee {

            private int employeeId;

            private String firstName;

            private String lastName;

            private String dateOfBirth;

            private int salary;

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

                this.employeeId =employeeId;

                this.firstName = firstName;

                this.lastName = lastName;

                this.dateOfBirth = dateOfBirth;

                this.salary = salary;

            }

            public int GetID() {

                return employeeId;

            }

            public String GetFN(){

                return firstName;

            }

            public String GetLn(){

                return lastName;

            }

            public String GetDOB(){

                return dateOfBirth;

            }

            public int GetSal() {

                return salary;

            }

            public void SETID(int employeeId) {

                if(employeeId > 0) {

                    this.employeeId = employeeId;

                }else {

                    System.out.println("invalid ID");

                }

            }

            public void SETFN(String firstName){

                if(firstName != null) {

                    this.firstName = firstName;

                }else {

                    System.out.println("invalid Fname");

                }

            }

            public void SETLn(String lastName){

                if(lastName != null) {

                    this.lastName = lastName;

                }else {

                    System.out.println("invalid Lname");

                }

            }

            public void SETDOB(String dateOfBirth){

                if(dateOfBirth != null) {

                    this.dateOfBirth = dateOfBirth;

                }else {

                    System.out.println("invalid DOB");

                }

            }

            public void SETSal(int salary) {

                if(salary > 0) {

                    this.salary = salary;

                }else {

                    System.out.println("invalid salary");

                }

            }

            public double calcbon() {

                return salary \* 0.10;

            }

        }

        public class EmpMan {

            public static void main(String[] args) {

                Employee employee = new Employee(12, "Atharva", "gaikwad", "2002", 1000);

                System.out.println("employeeId : " + employee.GetID());

                System.out.println("firstName : " + employee.GetFN());

                System.out.println("lastName : " + employee.GetLn());

                System.out.println("dateOfBirth : " + employee.GetDOB());

                System.out.println("salary : " + employee.GetSal());

                employee.SETID(10);

                employee.SETLn("hhhh");

                System.out.println("employeeId : " + employee.GetID());

                System.out.println("firstName : " + employee.GetFN());

                System.out.println("lastName : " + employee.GetLn());

                System.out.println("dateOfBirth : " + employee.GetDOB());

                System.out.println("salary : " + employee.GetSal());

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

try-catch, finally

        package assignment3;

        import java.util.Date;

        public class finall {

            private String guestName;

            private int numberOfGuests;

            private Date reservationDate;

            // Constructor

            public finall(String guestName, int numberOfGuests, Date reservationDate) {

                try {

                    if (numberOfGuests < 0) {

                        throw new IllegalArgumentException("Number of guests cannot be negative.");

                    }

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

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

                    }

                    if (reservationDate == null) {

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

                    }

                    this.guestName = guestName;

                    this.numberOfGuests = numberOfGuests;

                    this.reservationDate = reservationDate;

                } catch (IllegalArgumentException e) {

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

                } catch (NullPointerException e) {

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

                } finally {

                    // Any cleanup operations can be done here

                    // In this example, there are no specific cleanup operations to perform

                }

            }

            // Method to print reservation details

            public void printReservationDetails() {

                try {

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

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

                    System.out.println("Number of Guests: " + numberOfGuests);

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

                } catch (NullPointerException e) {

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

                } finally {

                    cleanup();

                }

            }

            private void cleanup() {

                System.out.println("Cleanup operations completed.");

            }

            public static void main(String[] args) {

                // Example usage

                Date today = new Date(); // Current date

                // Example 1: Negative number of guests

                finall reservation1 = null;

                try {

                    reservation1 = new finall("John", -2, today);

                } finally {

                    if (reservation1 != null) {

                        reservation1.printReservationDetails();

                    }

                }

                // Example 2: Null guest name

                finall reservation2 = null;

                try {

                    reservation2 = new finall(null, 3, today);

                } finally {

                    if (reservation2 != null) {

                        reservation2.printReservationDetails();

                    }

                }

                // Example 3: Null reservation date

                finall reservation3 = null;

                try {

                    reservation3 = new finall("hello", 2, null);

                } finally {

                    if (reservation3 != null) {

                        reservation3.printReservationDetails();

                    }

                }

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Multithreading

        package assignment4;

        class AttendanceThread extends Thread {

            private String divisionName;

            private String[] students;

            public AttendanceThread(String divisionName, String... students) {

                this.divisionName = divisionName;

                this.students = students;

            }

            @Override

            public void run() {

                System.out.println("Attendance for " + divisionName + ":");

                for (String student : students) {

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

                }

            }

        }

        class LateStudent implements Runnable {

            private String studentName;

            public LateStudent(String studentName) {

                this.studentName = studentName;

            }

            @Override

            public void run() {

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

            }

        }

        public class Multithreading {

            public static void main(String[] args) {

                AttendanceThread class1Thread = new AttendanceThread("Class 1", "Student 1", "Student 2");

                class1Thread.start();

                AttendanceThread class2Thread = new AttendanceThread("Class 2", "Student 3", "Student 4");

                class2Thread.start();

                // Separate instances of LateStudent for each class

                for (String student : new String[]{"Student 3", "Student 2"}) {

                    new Thread(new LateStudent(student)).start();

                }

            }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

ArrayList.

        package Assignment3;

        import java.util.ArrayList;

        public class ARR {

            public static void main(String[] args) {

                // Create an ArrayList

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

                // Adding elements to the ArrayList

                fruits.add("Apple");

                fruits.add("Banana");

                fruits.add("Cherry");

                System.out.println("Initial ArrayList: " + fruits);

                // Accessing elements in the ArrayList

                String firstFruit = fruits.get(0);

                System.out.println("First fruit: " + firstFruit);

                // Updating elements in the ArrayList

                fruits.set(1, "Blueberry");

                System.out.println("ArrayList after update: " + fruits);

                // Removing elements from the ArrayList by index

                fruits.remove(2);

                System.out.println("ArrayList after removing element at index 2: " + fruits);

                // Removing elements from the ArrayList by value

                fruits.remove("Apple");

                System.out.println("ArrayList after removing 'Apple': " + fruits);

                // Iterating over the ArrayList using a for-each loop

                System.out.println("Iterating over the ArrayList:");

                for (String fruit : fruits) {

                    System.out.println(fruit);

                }

                // Iterating over the ArrayList using a for loop with index

                System.out.println("Iterating over the ArrayList using index:");

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

                    System.out.println(fruits.get(i));

                }

                // Checking if the ArrayList contains a specific element

                boolean containsBanana = fruits.contains("Banana");

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

                // Getting the size of the ArrayList

                int size = fruits.size();

                System.out.println("Size of the ArrayList: " + size);

                // Clearing all elements from the ArrayList

                fruits.clear();

                System.out.println("ArrayList after clearing all elements: " + fruits);

            }

        }

            package assignment5;

            import java.util.ArrayList;

            import java.util.Scanner;

            class CourseEnrollment {

                private String courseId;

                private String courseName;

                private ArrayList<String> studentsEnrolled;

                public CourseEnrollment(String courseId, String courseName) {

                    this.courseId = courseId;

                    this.courseName = courseName;

                    this.studentsEnrolled = new ArrayList<>();

                }

                public String getCourseId() {

                    return courseId;

                }

                public String getCourseName() {

                    return courseName;

                }

                public ArrayList<String> getStudentsEnrolled() {

                    return studentsEnrolled;

                }

                public void addStudent(String studentId, String studentName) {

                    String entry = "Student ID: " + studentId + ", Student Name: " + studentName;

                    studentsEnrolled.add(entry);

                }

            }

            public class CourseManagement {

                public static void main(String[] args) {

                    Scanner scanner = new Scanner(System.in);

                    // Create an ArrayList to store course details

                    ArrayList<CourseEnrollment> courses = new ArrayList<>();

                    // Take input for course details

                    System.out.println("Enter Course Details:");

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

                    String courseId = scanner.nextLine();

                    System.out.print("Course Name: ");

                    String courseName = scanner.nextLine();

                    // Create a new course object

                    CourseEnrollment course = new CourseEnrollment(courseId, courseName);

                    // Take input for student details

                    System.out.println("Enter Student Details:");

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

                    String studentId = scanner.nextLine();

                    System.out.print("Student Name: ");

                    String studentName = scanner.nextLine();

                    // Add the student to the course

                    course.addStudent(studentId, studentName);

                    // Add the course to the list of courses

                    courses.add(course);

                    // Display success message

                    System.out.println("Entry added successfully!");

                    // Optional: Display the enrolled students for each course

                    for (CourseEnrollment enrolledCourse : courses) {

                        System.out.println("\nCourse ID: " + enrolledCourse.getCourseId());

                        System.out.println("Course Name: " + enrolledCourse.getCourseName());

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

                        for (String student : enrolledCourse.getStudentsEnrolled()) {

                            System.out.println(student);

                        }

                    }

                    // Close the scanner

                    scanner.close();

                }

            }

            package assignment5;

            import java.util.ArrayList;

            public class App {

                public static void main(String[] args) {

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

                    fruits.add("Apple");

                    fruits.add("Pear");

                    fruits.add("Grapes");

                    System.out.println("Fruits: " + fruits);

                    System.out.println("Fruits in the list");

                    for(String fruit: fruits)

                    {

                        System.out.println(fruit);

                    }

                    System.out.println("First Fruit: " + fruits.get(0));

                    System.out.println("Second Fruit: " +  fruits.get(1));

                    fruits.set(1, "Mango");

                    System.out.println(fruits);

                    String removed\_fruit = "Grapes";

                    fruits.remove(removed\_fruit);

                    System.out.println("List of fruits after removing "+ removed\_fruit + fruits);

                    java.util.Collections.sort(fruits);

                    System.out.println("Sorted list of fruits: " + fruits);

                    if(fruits.contains(removed\_fruit))

                    {

                        System.out.println("Fruit Exist!");

                    }else {

                        System.out.println("Fruit Not present");

                    }

                    if(fruits.isEmpty())

                    {

                        System.out.print(true);

                    }else {

                        System.out.println("Empty");

                    }

                    fruits.clear();

                    System.out.println("List of fruits after clearing: " + fruits);

                }

            }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

JDBC

        import java.sql.\*;

        import java.util.Scanner;

        public class jdbc1 {

            // JDBC URL, username, and password

            static final String JDBC\_URL = "jdbc:mysql://localhost:3306/java1";

            static final String USERNAME = "root";

            static final String PASSWORD = "tentacool";

            public static void main(String[] args) {

                try {

                    // Connect to the database

                    Connection connection = DriverManager.getConnection(JDBC\_URL, USERNAME, PASSWORD);

                    // Create tables if not exists

                    createStudentTable(connection);

                    // Take user input for functionality choice

                    Scanner scanner = new Scanner(System.in);

                    System.out.println("Choose an option:");

                    System.out.println("a) Student registration");

                    System.out.println("b) Recording attendance");

                    System.out.println("c) Generating attendance reports");

                    char option = scanner.nextLine().charAt(0);

                    // Perform action based on user's choice

                    switch (option) {

                        case 'a':

                            performStudentRegistration(connection, scanner);

                            break;

                        case 'c':

                            generateAttendanceReport(connection);

                            break;

                        default:

                            System.out.println("Invalid option!");

                    }

                    // Close the connection

                    connection.close();

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

            // Create student table if not exists

            public static void createStudentTable(Connection connection) throws SQLException {

                String sql = "CREATE TABLE IF NOT EXISTS students (" +

                             "id INT AUTO\_INCREMENT PRIMARY KEY," +

                             "name VARCHAR(100) NOT NULL," +

                             "student\_id VARCHAR(20) UNIQUE NOT NULL," +

                             "department VARCHAR(100) NOT NULL)";

                Statement statement = connection.createStatement();

                statement.executeUpdate(sql);

                statement.close();

            }

            // Perform student registration

            public static void performStudentRegistration(Connection connection, Scanner scanner) {

                try {

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

                    String name = scanner.nextLine();

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

                    String studentId = scanner.nextLine();

                    System.out.println("Enter student department:");

                    String department = scanner.nextLine();

                    registerStudent(connection, name, studentId, department);

                    System.out.println("Student registered successfully!");

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

            // Register a new student

            public static void registerStudent(Connection connection, String name, String studentId, String department) throws SQLException {

                String sql = "INSERT INTO students (name, student\_id, department) VALUES (?, ?, ?)";

                PreparedStatement preparedStatement = connection.prepareStatement(sql);

                preparedStatement.setString(1, name);

                preparedStatement.setString(2, studentId);

                preparedStatement.setString(3, department);

                preparedStatement.executeUpdate();

                preparedStatement.close();

            }

            // Generate attendance report

            public static void generateAttendanceReport(Connection connection) {

                try {

                    String sql = "SELECT \* FROM attendance";

                    Statement statement = connection.createStatement();

                    ResultSet resultSet = statement.executeQuery(sql);

                    while (resultSet.next()) {

                        String studentId = resultSet.getString("student\_id");

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

                        boolean isPresent = resultSet.getBoolean("is\_present");

                        System.out.println("Student ID: " + studentId + ", Date: " + date + ", Present: " + isPresent);

                    }

                    resultSet.close();

                    statement.close();

                } catch (SQLException e) {

                    e.printStackTrace();

                }

            }

        }