```
import java.util.Scanner;
public class TheaterSeatingArrangement {
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);
        // Input: Number of rows and columns
in the theater
        System.out.print("Enter the number of
rows in the theater: ");
        int numRows = scanner.nextInt();
        System.out.print("Enter the number of
columns in the theater: ");
        int numCols = scanner.nextInt();
        // Create a 2D array to represent the
theater
        int[][] theater = new
int[numRows][numCols];
        // Main loop to process commands
        while (true) {
            // Print the current seating
arrangement
            System.out.println("CurrentSeating
Arrangement:");
            for (int i = 0; i < numRows; i++) {
                for (int j = 0; j < numCols; j++) {
                    System.out.print(theater[i][j] + "
");
                }
                System.out.println();
            }
            // Prompt for a command (row and
column)
            System.out.print("Enter a command
(row and column) or 'exit' to quit: ");
            String input = scanner.next();
            if (input.equals("exit")) {
                break; // Exit the program
            }
            // Parse row and column from the
input
            int row = Integer.parseInt(input);
            int col = scanner.nextInt();
            // Check if the input is valid
            if (row >= 0 && row < numRows
&& col >= 0 && col < numCols) {
                // Mark the seat as occupied

                theater[row][col] = 1;
                System.out.println("Seat at row " +
row + ", column " + col + " is now
occupied.");
            } else {
                System.out.println("Invalid input.
Please enter valid row and column
numbers.");
            }
        }
        System.out.println("Thank you for using
the Theater Seating Arrangement
program!");
    }
}
```

```
import java.util.Scanner;
public class UserProfileManager {
  public static void main(String[] args) {
    Scanner scanner = new
Scanner(System.in);
    // Input: User's initial profile
    System.out.print("Enter user's name: ");
    String name = scanner.nextLine();
    System.out.print("Enter user's email
address: ");
    String email = scanner.nextLine();
    System.out.print("Enter user's bio: ");
    String bio = scanner.nextLine();
    // Main loop to process user actions
    while (true) {
      // Display menu of options
      System.out.println("\nUser Profile
Options:");
      System.out.println("1. Display
name");
      System.out.println("2. Display email
address");
      System.out.println("3. Display bio");
      System.out.println("4. Update email
address");
      System.out.println("5. Update bio");
      System.out.println("6. Exit");

      // Read user's choice
      System.out.print("Enter your choice
(1-6): ");
      int choice = scanner.nextInt();
      scanner.nextLine(); // Consume the
newline character
      switch (choice) {
        case 1:
          // Display the user's name
          System.out.println("Name: " +
name);
          break;
        case 2:
          // Display the user's email
address
          System.out.println("Email
Address: " + email);
          break;
        case 3:
          // Display the user's bio
          System.out.println("Bio: " + bio);
          break;
        case 4:
          // Update the user's email
address
          System.out.print("Enter new
email address: ");
          email = scanner.nextLine();
          System.out.println("Email
address updated successfully.");
          break;
        case 5:
          // Update the user's bio
          System.out.print("Enter new bio:
");
          bio = scanner.nextLine();
          System.out.println("Bio updated
successfully.");
          break;
        case 6:
          // Exit the program
          System.out.println("Goodbye!");
          System.exit(0);
        default:
                System.out.println("Invalid choice. Please select a valid option (1-6).");
            }
        }
    }
}
```

```
// Superclass: Vehicle
class Vehicle {
    private String brand;
    private double speed;
    public Vehicle(String brand) {
        this.brand = brand;
        this.speed = 0.0;
    }
    public void accelerate(double amount) {
        speed += amount;
        System.out.println(brand + " is accelerating. Current speed: " + speed + " km/h");
    }
    public void brake(double amount) {
        speed -= amount;
        System.out.println(brand + " is braking. Current speed: " + speed + " km/h");
    }
}
// Subclass: Car
class Car extends Vehicle {
    private int numDoors;
    private String fuelType;
    public Car(String brand, int numDoors, String fuelType) {
        super(brand);
        this.numDoors = numDoors;
        this.fuelType = fuelType;
    }
    public void honk() {
        System.out.println(getBrand() + " is honking!");
    }
    public String getBrand() {
        return super.brand;
    }
}
// Subclass: Bicycle
class Bicycle extends Vehicle {
    private int numGears;
    private String bikeType;
    public Bicycle(String brand, int numGears, String bikeType) {
        super(brand);
        this.numGears = numGears;
        this.bikeType = bikeType;
    }
    public void ringBell() {
        System.out.println(getBrand() + " is ringing the bell!");
    }
    public String getBrand() {
        return super.brand;
    }
}
public class VehicleInheritanceDemo {
    public static void main(String[] args) {
        // Create a car and a bicycle
        Car myCar = new Car("Toyota", 4, "Gasoline");
        Bicycle myBicycle = new Bicycle("Schwinn", 21, "Mountain Bike");
        // Demonstrate car's and bicycle's attributes and behaviors
        System.out.println("Car Brand: " + myCar.getBrand());
        myCar.accelerate(40);
        myCar.brake(10);
        myCar.honk();
        System.out.println("\nBicycle Brand: " +
myBicycle.getBrand());
        myBicycle.accelerate(20);
        myBicycle.brake(5);
        myBicycle.ringBell();
    }
}
```

```
// Create a package "university"
package university;
// Create a sub-package "students" inside
"university"
package university.students;
public class StudentManager {
    public void enrollStudent(String
studentName, String courseName) {
        System.out.println("Enrolled student "
+ studentName + " in course " +
courseName);
    }
    public void graduateStudent(String
studentName) {
        System.out.println("Graduated student
" + studentName);
    }
}
// Create a sub-package "courses" inside
"university"
package university.courses;
public class CourseManager {
    public void createCourse(String
courseName) {
        System.out.println("Created course " +
courseName);
    }
    public void deleteCourse(String
courseName) {
        System.out.println("Deleted course " +
courseName);
    }
}
// Create a class "UniversityDemo" to
demonstrate the use of packages and sub-
packages
import university.students.StudentManager;
import university.courses.CourseManager;
public class UniversityDemo {
    public static void main(String[] args) {
        // Create instances of StudentManager
and CourseManager from the respective
sub-packages
        StudentManager studentManager =
new StudentManager();
        CourseManager courseManager = new
CourseManager();
        // Demonstrate student management
operations
        studentManager.enrollStudent("Alice",
"Math");

courseManager.createCourse("Physics");
    }
}
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class ExceptionHandlingDemo {
   public static void main(String[] args) {
      performFileOperations();
   }
   public static void performFileOperations()
{
      try {
         // Attempt to read from a non-
existent file
         FileReader fileReader = new
FileReader("non_existent_file.txt");
         BufferedReader reader = new
BufferedReader(fileReader);
         String line = reader.readLine();
         reader.close();
      } catch (IOException e) {
         System.out.println("IOException: " +
e.getMessage());
         System.out.println("File not found or
unable to read.");
      }
      try {
         // Attempt to write to a read-only
file
         FileWriter fileWriter = new
FileWriter("read_only_file.txt");
         fileWriter.write("This is a write
operation.");
         fileWriter.close();
      } catch (IOException e) {
         System.out.println("IOException: " +
e.getMessage());
         System.out.println("Permission
denied for writing.");
      }
      try {
         // Attempt to read from a file with
invalid content
         FileReader fileReader = new
FileReader("invalid_content.txt");
         BufferedReader reader = new
BufferedReader(fileReader);
         String line = reader.readLine();
         int number = Integer.parseInt(line);
// This line will throw a
NumberFormatException
         reader.close();
      } catch (IOException e) {
         System.out.println("IOException: " +
e.getMessage());
```

```
            System.out.println("Error reading file.");
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException: " + e.getMessage());
            System.out.println("Invalid content in the file.");
        }
    }
}
```

```
class Worker extends Thread {
    private String name;
    public Worker(String name) {
        this.name = name;
    }
    @Override
    public void run() {
        System.out.println(name + " is starting.");
        for (int i = 1; i <= 10; i++) {
            System.out.println(name + " - Count: " + i);
            try {
                // Sleep for a random amount of time (simulating work)
                Thread.sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println(name + " is done.");
    }
}
public class ThreadDemo {
    public static void main(String[] args) {
        System.out.println("Main thread is starting.");
        // Create three worker threads
        Worker worker1 = new Worker("Worker 1");
        Worker worker2 = new Worker("Worker 2");
        Worker worker3 = new Worker("Worker 3");
        // Start the worker threads
        worker1.start();
        worker2.start();
        worker3.start();
        try {
            // Wait for all worker threads to finish
            worker1.join();
            worker2.join();
            worker3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Main thread is
done.");
    }
}
```

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class FileHandlingDemo {
    public static void main(String[] args) {
        String inputFile = "input.txt";
        try {
            // Step 1: Open the input file for
reading
            FileReader fileReader = new
FileReader(inputFile);
            BufferedReader reader = new
BufferedReader(fileReader);
            // Step 2: Create a StringBuilder to
store the modified data
            StringBuilder modifiedData = new
StringBuilder();
            // Step 3: Read each line from the
input file
            String line;
            while ((line = reader.readLine()) !=
null) {
                // Step 4: Convert each line to
uppercase and append it to the
StringBuilder

modifiedData.append(line.toUpperCase()).ap
pend("\n");
            }
            // Step 5: Close the input file
            reader.close();
            // Step 6: Open the same file for
writing (this will overwrite the existing
content)
            FileWriter fileWriter = new
FileWriter(inputFile);
            BufferedWriter writer = new
BufferedWriter(fileWriter);
```

```
            // Step 7: Write the modified data from the StringBuilder to the file
            writer.write(modifiedData.toString());
            // Step 8: Close the output file
            writer.close();
            System.out.println("File processing complete.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
class Student {
    private int id;
    private String name;
    private int age;
    private double gpa;
    public Student(int id, String name, int age, double gpa) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gpa = gpa;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public double getGpa() {
        return gpa;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", age=" + age +
            ", gpa=" + gpa +
            '}';
    }
}
public class CollectionFrameworkDemo {
    public static void main(String[] args) {
        List<Student> students = new
ArrayList<>();

            Map<Integer, Student> studentMap
= new HashMap<>();
        students.add(new Student(101, "Alice",
20, 3.8));
      students.add(newStudent(103,"Charlie",
21, 3.9));
        for (Student student : students) {
            studentMap.put(student.getId(),
student);
        }
        // Search for a student by ID and
display their details
        int searchId = 103;
        Student searchedStudent =
studentMap.get(searchId);
        System.out.println("Searched Student: "
+ searchedStudent);
        // Sort the students by GPA and display
the sorted list
        students.sort((s1, s2) ->
Double.compare(s2.getGpa(), s1.getGpa()));
        System.out.println("Sorted Students by
GPA:");
        for (Student student : students) {
            System.out.println(student);
        }
        // Filter students who are older than a
certain age and display the filtered list
        int filterAge = 20;
        List<Student> filteredStudents = new
ArrayList<>();
        for (Student student : students) {
            if (student.getAge() > filterAge) {
                filteredStudents.add(student);
            }
        }
        System.out.println("Students older than
" + filterAge + " years:");
        for (Student student : filteredStudents)
{
            System.out.println(student);
        }
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;
public class StudentRegistration {
    // Database connection parameters
    private static final String DB_URL =
"jdbc:mysql://localhost:3306/studentdb";
    private static final String DB_USER =
"root";
    private static final String DB_PASSWORD
= "your_password";
    public static void main(String[] args) {
        try {
            // Load the MySQL JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");
            // Collect student details from the
user
            Scanner scanner = new
Scanner(System.in);
            System.out.print("Enter student
name: ");
            String name = scanner.nextLine();
            System.out.print("Enter student age:
");
            int age = scanner.nextInt();
            scanner.nextLine(); // Consume the
newline character
            System.out.print("Enter student
course: ");
            String course = scanner.nextLine();
            // Validate the collected data
            if (name.isEmpty() || course.isEmpty()
|| age <= 0) {
                System.out.println("Invalid input.
Please provide valid student details.");
                return;
            }
            // Establish a database connection
            Connection connection =
DriverManager.getConnection(DB_URL,
DB_USER, DB_PASSWORD);
            // Create an SQL INSERT statement
            String insertSQL = "INSERT INTO
students (name, age, course) VALUES (?, ?,
?)";
            PreparedStatement
preparedStatement =
connection.prepareStatement(insertSQL);
            preparedStatement.setString(1,
name);
            preparedStatement.setInt(2, age);
            preparedStatement.setString(3,
course);
            // Execute the SQL INSERT statement
            int rowsAffected =
preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Student
registration successful.");
            } else {
                System.out.println("Student
registration failed.");
            }
            // Close the database connection
            connection.close();
        } catch (ClassNotFoundException |
SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.util.ArrayList;
import java.util.List;
class GenericStack<T> {
    private List<T> stack;
    public GenericStack() {
        stack = new ArrayList<>();
    }
    public void push(T element) {
        stack.add(element);
        System.out.println("Pushed: " +
element);
    }
    public T pop() {
        if (!isEmpty()) {
            T element =
stack.remove(stack.size() - 1);
            System.out.println("Popped: " +
element);
            return element;
        } else {
            System.out.println("Stack is empty.");
            return null;
        }
    }
    public boolean isEmpty() {
        return stack.isEmpty();
    }
    public void displayStack() {
        System.out.println("Stack Contents: " +
stack);
    }
}
public class GenericStackDemo {
    public static void main(String[] args) {
        // Create a generic stack for integers
        GenericStack<Integer> intStack = new
GenericStack<>();
        intStack.push(10);
        intStack.push(20);
        intStack.displayStack();
        intStack.pop();
        intStack.displayStack();
        // Create a generic stack for strings
        GenericStack<String>    stringStack    =
new GenericStack<>();
        stringStack.push("Hello");
        stringStack.push("World");
        stringStack.displayStack();
        stringStack.pop();
        stringStack.displayStack();
        // Create a generic stack for doubles
        GenericStack<Double>  doubleStack  =
new GenericStack<>();
        doubleStack.push(3.14);
        doubleStack.push(2.71);
        doubleStack.displayStack();
        doubleStack.pop();
        doubleStack.displayStack();
    }
}
```