```
import java.util.ArrayList;
import java.util.Scanner;

public class ArrayListExample {
   public static void main(String[] args) {
      ArrayList<Integer> list = new ArrayList<Integer>();
      Scanner sc = new Scanner(System.in);
      int choice;

      do {
         System.out.println("Choose an operation:");
         System.out.println("1. Display the list");
         System.out.println("2. Search for a number");
         System.out.println("3. Remove an element from a position");
         System.out.println("4. Add an element to the list");
         System.out.println("5. Check if the list is empty");
         System.out.println("6. Exit");
         choice = sc.nextInt();

         switch (choice) {
            case 1:
               if (list.isEmpty()) {
                  System.out.println("The list is empty.");
               } else {
                  System.out.println("The list is:");
                  for (int i : list) {
                     System.out.print(i + " ");
                  }
                  System.out.println();
               }
               break;

            case 2:
               if (list.isEmpty()) {
                  System.out.println("The list is empty.");
               } else {
                  System.out.println("Enter a number to search for:");
                  int searchNum = sc.nextInt();
                  if (list.contains(searchNum)) {
                     System.out.println(searchNum + " is present in the
list.");
                  } else {
```

```
                    System.out.println(searchNum + " is not present in
the list.");
                }
            }
            break;

        case 3:
            if (list.isEmpty()) {
                System.out.println("The list is empty.");
            } else {
                System.out.println("Enter the position of the element
to remove (starting from 0):");
                int pos = sc.nextInt();
                if (pos < 0 || pos >= list.size()) {
                    System.out.println("Invalid position.");
                } else {
                    list.remove(pos);
                    System.out.println("Element removed.");
                }
            }
            break;

        case 4:
            System.out.println("Enter a number to add to the
list:");
            int addNum = sc.nextInt();
            list.add(addNum);
            System.out.println("Element added.");
            break;

        case 5:
            if (list.isEmpty()) {
                System.out.println("The list is empty.");
            } else {
                System.out.println("The list is not empty.");
            }
            break;

        case 6:
            System.out.println("Exiting...");
            break;

        default:
```

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

        } while (choice != 6);

        sc.close();
    }
}
```

```
import java.util.LinkedList;
import java.util.Scanner;

class Student {
    private String name;
    private int age;
    private double mark;

    public Student(String name, int age, double mark) {
        this.name = name;
        this.age = age;
        this.mark = mark;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public double getMark() {
        return mark;
    }

    public String toString() {
        return "Name: " + name + ", Age: " + age + ", Mark: " + mark;
    }

    public boolean equals(Object obj) {
        if (obj instanceof Student) {
```

```
         Student other = (Student) obj;
         return this.name.equals(other.name) && this.age == other.age
&& this.mark == other.mark;
      }
      return false;
   }
}

public class LinkedListExample {
   public static void main(String[] args) {
      LinkedList<Student> list = new LinkedList<Student>();
      Scanner sc = new Scanner(System.in);
      int choice;

      do {
         System.out.println("Choose an operation:");
         System.out.println("1. Display the list");
         System.out.println("2. Search for a student");
         System.out.println("3. Remove a student");
         System.out.println("4. Count the number of students");
         System.out.println("5. Exit");
         choice = sc.nextInt();

         switch (choice) {
            case 1:
               if (list.isEmpty()) {
                  System.out.println("The list is empty.");
               } else {
                  System.out.println("The list is:");
                  for (Student s : list) {
                     System.out.println(s);
                  }
               }
               break;

            case 2:
               if (list.isEmpty()) {
                  System.out.println("The list is empty.");
               } else {
                  System.out.println("Enter the details of the student
to search for:");
                  System.out.print("Name: ");
                  String name = sc.next();
```

```
                System.out.print("Age: ");
                int age = sc.nextInt();
                System.out.print("Mark: ");
                double mark = sc.nextDouble();
                Student searchStudent = new Student(name, age, mark);
                if (list.contains(searchStudent)) {
                   System.out.println("The student exists in the
list.");
                } else {
                   System.out.println("The student does not exist in
the list.");
                }
             }
             break;

          case 3:
             if (list.isEmpty()) {
                System.out.println("The list is empty.");
             } else {
                System.out.println("Enter the details of the student
to remove:");
                System.out.print("Name: ");
                String name = sc.next();
                System.out.print("Age: ");
                int age = sc.nextInt();
                System.out.print("Mark: ");
                double mark = sc.nextDouble();
                Student removeStudent = new Student(name, age, mark);
                if (list.remove(removeStudent)) {
                   System.out.println("The student has been removed
from the list.");
                } else {
                   System.out.println("The student does not exist in
the list.");
                }
             }
             break;

          case 4:
             System.out.println("The number of students in the list
is " + list.size() + ".");
             break;
```

```
            case 5:
               System.out.println("Exiting...");
               break;

            default:
               System.out.println("Invalid choice.");
         }

      } while (choice != 5);

      sc.close();
   }
}
```

```
import java.util.Stack;

public class DecimalToBinaryUsingStack {
   public static void main(String[] args) {
      int decimal = 27;
      Stack<Integer> stack = new Stack<Integer>();
      while (decimal > 0) {
         int remainder = decimal % 2;
         stack.push(remainder);
         decimal /= 2;
      }
      System.out.print("The binary equivalent of 27 is: ");
      while (!stack.isEmpty()) {
         System.out.print(stack.pop());
      }
   }
}
```

```
import java.util.Stack;

public class PostfixEvaluationUsingStack {
   public static int evaluatePostfix(String postfix) {
      Stack<Integer> stack = new Stack<Integer>();
      for (int i = 0; i < postfix.length(); i++) {
         char ch = postfix.charAt(i);
         if (Character.isDigit(ch)) {
```

```
            stack.push(ch - '0');
         } else {
            int operand2 = stack.pop();
            int operand1 = stack.pop();
            switch (ch) {
               case '+':
                  stack.push(operand1 + operand2);
                  break;
               case '-':
                  stack.push(operand1 - operand2);
                  break;
               case '*':
                  stack.push(operand1 * operand2);
                  break;
               case '/':
                  stack.push(operand1 / operand2);
                  break;
            }
         }
      }
      return stack.pop();
   }

   public static void main(String[] args) {
      String postfix = "53+82-*";
      int result = evaluatePostfix(postfix);
      System.out.println("The result of " + postfix + " is " + result +
".");
   }
}
```

```
import java.util.*;

public class BreadthFirstSearchUsingArrayDeque {
   private int V; // number of vertices
   private List<List<Integer>> adj; // adjacency list

   public BreadthFirstSearchUsingArrayDeque(int v) {
      V = v;
      adj = new ArrayList<>(v);
```

```
      for (int i = 0; i < v; i++) {
         adj.add(new ArrayList<Integer>());
      }
   }

   public void addEdge(int v, int w) {
      adj.get(v).add(w);
   }

   public void BFS(int start) {
      boolean[] visited = new boolean[V];
      ArrayDeque<Integer> queue = new ArrayDeque<Integer>();
      visited[start] = true;
      queue.add(start);
      while (!queue.isEmpty()) {
         int vertex = queue.poll();
         System.out.print(vertex + " ");
         for (int neighbor : adj.get(vertex)) {
            if (!visited[neighbor]) {
               visited[neighbor] = true;
               queue.add(neighbor);
            }
         }
      }
   }

   public static void main(String[] args) {
      BreadthFirstSearchUsingArrayDeque g = new
BreadthFirstSearchUsingArrayDeque(6);
      g.addEdge(0, 1);
      g.addEdge(0, 2);
      g.addEdge(1, 2);
      g.addEdge(2, 0);
      g.addEdge(2, 3);
      g.addEdge(3, 3);
      g.addEdge(4, 5);
      System.out.print("Breadth-first traversal of the graph starting
from vertex 2: ");
      g.BFS(2);
   }
}
```

```java
import java.util.*;

public class DepthFirstSearchUsingStack {
   private int V; // number of vertices
   private List<List<Integer>> adj; // adjacency list

   public DepthFirstSearchUsingStack(int v) {
      V = v;
      adj = new ArrayList<>(v);
      for (int i = 0; i < v; i++) {
         adj.add(new ArrayList<Integer>());
      }
   }

   public void addEdge(int v, int w) {
      adj.get(v).add(w);
   }

   public void DFS(int start) {
      boolean[] visited = new boolean[V];
      Stack<Integer> stack = new Stack<Integer>();
      visited[start] = true;
      stack.push(start);
      while (!stack.isEmpty()) {
         int vertex = stack.pop();
         System.out.print(vertex + " ");
         for (int neighbor : adj.get(vertex)) {
            if (!visited[neighbor]) {
               visited[neighbor] = true;
               stack.push(neighbor);
            }
         }
      }
   }

   public static void main(String[] args) {
      DepthFirstSearchUsingStack g = new DepthFirstSearchUsingStack(6);
      g.addEdge(0, 1);
      g.addEdge(0, 2);
      g.addEdge(1, 2);
      g.addEdge(2, 0);
      g.addEdge(2, 3);
      g.addEdge(3, 3);
```

```
        g.addEdge(4, 5);
        System.out.print("Depth-first traversal of the graph starting
from vertex 2: ");
        g.DFS(2);
    }
}
```