1. Singly Linked List implementation using built-in class

Program:

```
import java.util.LinkedList;
class Main1 {
  public static void main(String[] args) {
    LinkedList<String> languages = new LinkedList<>();
    // add elements in LinkedList
    languages.add("Java");
    languages.add("Python");
    languages.add("JavaScript");
    languages.add("Kotlin");
    languages.add("1");
    System.out.println("LinkedList: " + languages);
    // remove elements from index 1
    String str = languages.remove(1);
    System.out.println("Removed Element: " + str);
    System.out.println("Updated LinkedList: " + languages);
  }
}
```

2. Implement Stack using Stack class

Program:

```
import java.io.*;
import java.util.Stack;

class Stack1
{
    // Pushing element on the top of the stack
    static void stack_push(Stack<Integer> stack)
    {
        for(int i = 0; i < 5; i++)
        {
            stack.push(i);
        }
    }
// popping element from the top of the stack
    static void stack_pop(Stack<Integer> stack)
    {
        System.out.println("Pop Operation:");
```

```
            for(int i = 0; i < 5; i++)
            {
                Integer y = (Integer) stack.pop();
                System.out.println(y);
            }
        }

        // Displaying element on the top of the stack
        static void stack_peek(Stack<Integer> stack)
        {
            Integer element = (Integer) stack.peek();
            System.out.println("Element on stack top: " + element);
        }

        // Searching element in the stack
        static void stack_search(Stack<Integer> stack, int element)
        {
            Integer pos = (Integer) stack.search(element);
              if(pos == -1)
                System.out.println("Element not found");
            else
                System.out.println("Element is found at position: " + pos);
        }
    public static void main (String[] args)
        {
            Stack<Integer> stack = new Stack<Integer>();
            stack_push (stack);
            stack_pop(stack);
            stack_push(stack);
            stack_peek(stack);
            stack_search(stack, 2);
            stack_search(stack, 6);
        }
}
```

3. Queue Implementation Queue using Queue class

Program:

```
import java.util.LinkedList;
import java.util.Queue;

public class QueueExample {

    public static void main(String[] args)
```

```
    {
        Queue<Integer> q  = new LinkedList<>();
         // Adds elements {0, 1, 2, 3, 4} to  the queue
        for (int i = 0; i < 5; i++)
            q.add(i);
         // Display contents of the queue.
        System.out.println("Elements of queue "  + q);
         // To remove the head of queue.
        int removedele = q.remove();
        System.out.println("removed element-"  + removedele);
         System.out.println(q);
         // To view the head of queue
        int head = q.peek();
        System.out.println("head of queue-"   + head);
        int size = q.size();
        System.out.println("Size of queue-"  + size);
    }
}
```

4. Implement Set using Set class

Program

```
import java.util.*;
public class SetExample {
    public static void main(String[] args)  {
        // Set demo with HashSet
        Set<String> Colors_Set = new HashSet<String>();
        Colors_Set.add("Red");
        Colors_Set.add("Green");
        Colors_Set.add("Blue");
        Colors_Set.add("Cyan");
        Colors_Set.add("Magenta");
        //print set contents
        System.out.print("Set contents:");
        System.out.println(Colors_Set);

        // Set demo with TreeSet
        System.out.print("\nSorted Set after converting to TreeSet:");
        Set<String> tree_Set = new TreeSet<String>(Colors_Set);
        System.out.println(tree_Set);
    }
}
```

5. Implementation of Map using LinkedHashMap class

Program

```
import java.util.LinkedHashMap;
public class MapExample {
    public static void main(String[] args)
    {
        // Creating an empty Linked Hash Map
        LinkedHashMap<Integer, String> students = new LinkedHashMap<>();
        // Adding data to Linked Hash Map in key-value pair
        students.put(101, "Aaliyah");
        students.put(102, "Taylor");
        students.put(103, "Zayn");
        students.put(104, "Sabrina");
        students.put(105, "Paul");
        // Showing size and data of the Linked Hash Map
        System.out.println("The size of the Linked Hash Map is:- "+ students.size());
        System.out.println(students);
        // Checking whether a certaint key is available or not
        if (students.containsKey(105)) {
            String name = students.get(105);
            System.out.println("The name of the student having Id 105 is:- " + name);
        }
    }
}
```