**StackDs.java**

```java
/* Stack java */
class Stack {
    private int maxSize;
    private int[] stackArray;
    private int top;
    public Stack(int size) {
        this.maxSize = size;
        this.stackArray = new int[maxSize];
        this.top = -1;
    }
public void push(int value) {
    if (top < maxSize - 1) {
        stackArray[++top] = value;
        System.out.println("Pushed: " + value);
    } else {
        System.out.println("Stack is full. Cannot push " + value);
    }
 }
public int pop() {
    if (top >= 0) {
        int value = stackArray[top--];
        System.out.println("Popped: " + value);
        return value;
    } else {
        System.out.println("Stack is empty. Cannot pop.");
        return -1;
    }
 }

public int peek() {
    if (top >= 0) {
        return stackArray[top];
    } else {
        System.out.println("Stack is empty. Cannot peek.");
        return -1;
    }
    }
}
public class StackDriver {
    public static void main(String[] args) {
        Stack stack = new Stack(5);
        stack.push(10);
        stack.push(20);
        stack.push(30);
        System.out.println("Top element: " + stack.peek());
        stack.pop();
        System.out.println("Top element after pop: " + stack.peek());
        stack.pop();
        stack.pop();
```

```
50          stack.pop();
51      }
52  }
```

**Stack.cpp**

```cpp
/* Stack In CPP */
#include <iostream>
using namespace std;
class Stack
{
private:
    int maxSize;
    int *stackArray;
    int top;

public:
    Stack(int size)
    {
        maxSize = size;
        stackArray = new int[maxSize];
        top = -1;
    }
    void push(int value)
    {
        if (top < maxSize - 1)
        {
            stackArray[++top] = value;
            cout << "Pushed: " << value << endl;
        }
        else
        {
            cout << "Stack is full. Cannot push " << value << endl;
        }
    }
    int pop()
    {
        if (top >= 0)
        {
            int value = stackArray[top--];
            cout << "Popped: " << value << endl;
            return value;
        }
        else
        {
            cout << "Stack is empty. Cannot pop." << endl;
            return -1;
        }
    }
    int peek()
    {
        if (top >= 0)
        {
            return stackArray[top];
        }
```

```
        else
        {
            cout << "Stack is empty. Cannot peek." << endl;
            return -1;
        }
    }
    ~Stack()
    {
        delete[] stackArray;
    }
};
int main()
{
    Stack stack(5);
    stack.push(10);
    stack.push(20);
    stack.push(30);
    cout << "Top element: " << stack.peek() << endl;
    stack.pop();
    cout << "Top element after pop: " << stack.peek() << endl;
    stack.pop();
    stack.pop();
    stack.pop();
    return 0;
}
```

**LinkedList.java**

```java
/* Linked List in Java */

class SingleLinkedList {
    public static void main(String args[]) {
        Node head = new Node(6);
        head.next = new Node(9);
        head.next.next = new Node(7);
        head.printLinkedList(head);
    }
}
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        this.next = null;
    }
    void printLinkedList(Node head) {
        while (head != null) {
            System.out.println(head.data);
            head = head.next;
        }
    }
}
```

**LinkedList.cpp**

```cpp
/* LINKED LIST IN CPP */
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    Node *next;
    Node(int data)
    {
        this->data = data;
        this->next = nullptr;
    }
    void printLinkedList(Node *head)
    {
        while (head != nullptr)
        {
            cout << head->data << endl;
            head = head->next;
        }
    }
};
int main()
{
    Node *head = new Node(6);
    head->next = new Node(9);
    head->next->next = new Node(7);
    head->printLinkedList(head);
    return 0;
}
```

**CircularLinkedList.java**

```java
/* Circular Linked List in Java */
class CircularLinkedList {
    public static void main(String args[]) {
        Node head = new Node(6);
        Node second = new Node(9);
        Node third = new Node(7);
        head.next = second;
        second.next = third;
        third.next = head;
        head.printLinkedList(head);
    }
}
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        this.next = null;
    }
    void printLinkedList(Node head) {
        if (head == null) return;
            Node current = head;
        do {
            System.out.println(current.data);
            current = current.next;
        } while (current != head);
    }
}
```

**CircularLinkedList.cpp**

```cpp
/* Circular Linked List CPP */
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    Node *next;
    Node(int data)
    {
        this->data = data;
        this->next = nullptr;
    }
    void printLinkedList(Node *head)
    {
        if (head == nullptr)
            return;
        Node *current = head;
        do
        {
            cout << current->data << endl;
            current = current->next;
        } while (current != head);
    }
};
int main()
{
    Node *head = new Node(6);
    Node *second = new Node(9);
    Node *third = new Node(7);
    head->next = second;
    second->next = third;
    third->next = head;
    head->printLinkedList(head);
    return 0;
}
```