1.Implement the program by using map.

CODE:

import java.util.HashMap;

import java.util.Map;

public class MapExample {

public static void main(String[] args) {

Map<String, Integer> map = new HashMap<>();

map.put("apple", 3);

map.put("banana", 5);

map.put("orange", 2);

System.out.println("Fruit Inventory:");

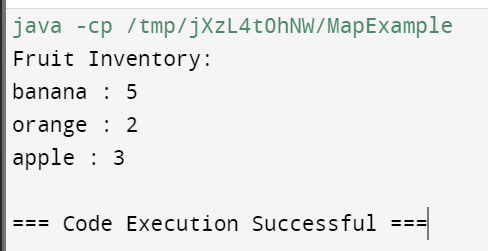
for (Map.Entry<String, Integer> entry : map.entrySet()) {

System.out.println(entry.getKey() + " : " + entry.getValue());

}

}

}OUTPUT:



2) Create program using linkedlist in queue.

CODE:

import java.util.LinkedList;

import java.util.Queue;

public class QueueExample {

public static void main(String[] args) {

Queue<Integer> queue = new LinkedList<>();

queue.offer(1);

queue.offer(2);

queue.offer(3);

System.out.println("Elements in the queue: " + queue);

System.out.println("Removed element: " + queue.poll());

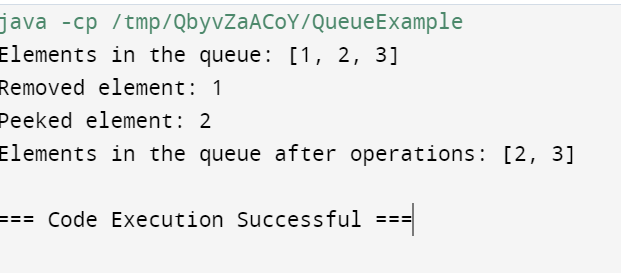
System.out.println("Peeked element: " + queue.peek());

System.out.println("Elements in the queue after operations: " + queue);

}

}

OUTPUT:



3) Create program using linkedlist in stack.

CODE:

import java.util.LinkedList;

public class StackUsingLinkedList {

private LinkedList<Integer> stack;

public StackUsingLinkedList() {

stack = new LinkedList<>();

}

public void push(int value) {

stack.addLast(value);

}

public int pop() {

if (stack.isEmpty()) {

throw new IllegalStateException("Stack is empty");

}

return stack.removeLast();

}

public int peek() {

if (stack.isEmpty()) {

throw new IllegalStateException("Stack is empty");

}

return stack.getLast();

}

public boolean isEmpty() {

return stack.isEmpty();

}

public int size() {

return stack.size();

}

public static void main(String[] args) {

StackUsingLinkedList stack = new StackUsingLinkedList();

stack.push(5);

stack.push(10);

stack.push(15);

System.out.println("Stack size: " + stack.size());

System.out.println("Top element: " + stack.peek());

System.out.println("Popping elements:");

while (!stack.isEmpty()) {

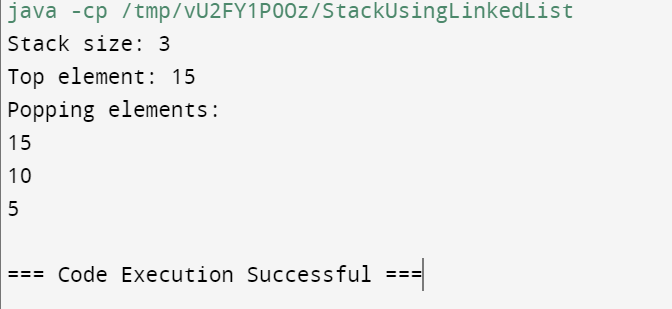
System.out.println(stack.pop());

}

}

}

OUTPUT:



4) Create mobile class with field price brand and so on take any one field and sort the class.

CODE:

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Mobile {

private String brand;

private double price;

public Mobile(String brand, double price) {

this.brand = brand;

this.price = price;

}

public double getPrice() {

return price;

}

public String getBrand() {

return brand;

}

public static void main(String[] args) {

List<Mobile> mobiles = new ArrayList<>();

mobiles.add(new Mobile("Samsung", 799.99));

mobiles.add(new Mobile("Apple", 999.99));

mobiles.add(new Mobile("OnePlus", 699.99));

Collections.sort(mobiles, Comparator.comparing(Mobile::getPrice));

for (Mobile mobile : mobiles) {

System.out.println("Brand: " + mobile.getBrand() + ", Price: $" + mobile.getPrice());

}

}

}

OUTPUT:

