(ArrayList)

2. **Search an Element**

Write a program to:

* Create an ArrayList of integers.
* Ask the user to enter a number.
* Check if the number exists in the list.

package day8\_Assign;

import java.util.\*;

public class Q1 {

public static void main(String[] args) {

ArrayList<Integer> nums = new ArrayList<>(Arrays.*asList*(10, 25, 30, 45, 50));

Scanner sc = new Scanner(System.*in*);

System.*out*.print("Enter a number:");

int num = sc.nextInt();

System.*out*.println(nums.contains(num) ? num + " is present." : num + " is not present.");

sc.close();

}

}

3. **. Remove Specific Element**

**Write a program to:**

* **Create an ArrayList of Strings.**
* **Add 5 fruits.**
* **Remove a specific fruit by name.**

**Display the updated list.**

package day8\_Assign;

import java.util.\*;

public class Q2 {

public static void main(String[] args) {

ArrayList<String> fruits = new ArrayList<>(Arrays.*asList*("Apple", "Banana", "Mango", "Orange", "Grapes"));

Scanner sc = new Scanner(System.*in*);

System.*out*.print("Enter fruit to remove: ");

String fruit = sc.nextLine();

fruits.remove(fruit);

System.*out*.println("updated list:"+fruits);

sc.close();

}

}

4. **Sort Elements**

Write a program to:

* Create an ArrayList of integers.
* Add at least 7 random numbers.
* Sort the list in ascending order.
* Display the sorted list.

import java.util.\*;

public class Q3 {

public static void main(String[] args) {

ArrayList<Integer> nums = new ArrayList<>(Arrays.asList(45, 12, 78, 34, 23, 89, 56));

Collections.sort(nums);

System.out.println("sorted list:"+nums);

}

}

5. **Reverse the ArrayList**

Write a program to:

* Create an ArrayList of characters.
* Add 5 characters.
* Reverse the list using Collections.reverse() and display it.

package day8\_Assign;

import java.util.\*;

public class Q4 {

public static void main(String[] args) {

ArrayList<Character> chars = new ArrayList<>(Arrays.asList('A', 'B', 'C', 'D', 'E'));

Collections.reverse(chars);

System.out.println("reversed list:"+chars);

}

}

6. **Update an Element**

Write a program to:

* Create an ArrayList of subjects.
* Replace one of the subjects (e.g., “Math” to “Statistics”).
* Print the list before and after the update.

package day8\_Assign;

import java.util.\*;

public class Q5 {

public static void main(String[] args) {

ArrayList<String> subjects = new ArrayList<>(Arrays.asList("Math", "Science", "English", "History"));

System.out.println("Before update:"+subjects);

subjects.set(subjects.indexOf("Math"),"Statistics");

System.out.println("After update:" +subjects);

}

}

7. **. Remove All Elements**

Write a program to:

* Create an ArrayList of integers.
* Add multiple elements.
* Remove all elements using clear() method.
* Display the size of the list.

package day8\_Assign;

import java.util.\*;

public class Q6 {

public static void main(String[] args) {

ArrayList<Integer> nums = new ArrayList<>(Arrays.asList(10, 20, 30, 40, 50));

nums.clear();

System.out.println("size after clearing: " + nums.size());

}

}

8. **. Iterate using Iterator**

Write a program to:

* Create an ArrayList of cities.
* Use Iterator to display each city.

package day8\_Assign;

import java.util.\*;

public class Q7 {

public static void main(String[] args) {

ArrayList<String> cities = new ArrayList<>(Arrays.asList("Delhi", "Mumbai", "Chennai", "Kolkata"));

Iterator<String> i = cities.iterator();

while (i.hasNext()) {

System.out.println(i.next());

}

}

}

9. **Store Custom Objects**

Write a program to:

* Create a class Student with fields: id, name, and marks.
* Create an ArrayList of Student objects.
* Add at least 3 students.
* Display the details using a loop.

package day8\_Assign;

import java.util.\*;

class Student{

int id;

String name;

int marks;

Student(int id,String name,int marks){

this.id=id;

this.name=name;

this.marks=marks;

}

}

public class Q8{

public static void main(String[] args){

ArrayList<Student> students=new ArrayList<>();

students.add(new Student(1,"Riya",98));

students.add(new Student(2,"Varma",95));

students.add(new Student(3,"RiyaVarma",89));

for(Student s:students){

System.out.println(s.id+" "+s.name+" "+s.marks);

}

}

}

10. **Copy One ArrayList to Another**

Write a program to:

* Create an ArrayList with some elements.
* Create a second ArrayList.
* Copy all elements from the first to the second using addAll() method.

package day8\_Assign;

import java.util.\*;

public class Q9{

public static void main(String[] args){

ArrayList<String> l1=new ArrayList<>(Arrays.asList("A","B","C","D"));

ArrayList<String> l2=new ArrayList<>();

l2.addAll(l1);

System.out.println("List1:"+l1);

System.out.println("List2:"+l2);

}

}

List(LinkedList)

1. **Create and Display a LinkedList**

Write a program to:

* Create a LinkedList of Strings.
* Add five colors to it.
* Display the list using a for-each loop.

package day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

LinkedList<String> colors=new LinkedList<>();

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

colors.add("Black");

for(String c:colors){

System.out.println(c);

}

}

}

2. **Add Elements at First and Last Position**

Write a program to:

* Create a LinkedList of integers.
* Add elements at the beginning and at the end.
* Display the updated list.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

LinkedList<Integer> l1=new LinkedList<>();

l1.addFirst(10);

l1.add(20);

l1.add(30);

l1.addLast(40);

System.out.println(l1);

}

}

3. **. Insert Element at Specific Position**

Write a program to:

* Create a LinkedList of names.
* Insert a name at index 2.
* Display the list before and after insertion.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

LinkedList<String> names=new LinkedList<>(Arrays.asList("A","B","C","D"));

System.out.println("Before:"+names);

names.add(2,"Emma");

System.out.println("After:"+names);

}

}

4. **Remove Elements**

Write a program to:

* Create a LinkedList of animal names.
* Remove the first and last elements.
* Remove a specific element by value.
* Display the list after each removal.

package day8\_Assign;

import java.util.\*;

public class Q4{

public static void main(String[] args){

LinkedList<String> animals=new LinkedList<>(Arrays.asList("Dog","Cat","Lion","Tiger","Elephant"));

animals.removeFirst();

System.out.println(animals);

animals.removeLast();

System.out.println(animals);

animals.remove("Lion");

System.out.println(animals);

}

}

5. **. Search for an Element**

Write a program to:

* Create a LinkedList of Strings.
* Ask the user for a string to search.
* Display if the string is found or not.

package day8\_Assign;

import java.util.\*;

public class Q5{

public static void main(String[] args){

LinkedList<String> list=new LinkedList<>(Arrays.asList("Red","Blue","Green","Yellow","Black"));

Scanner sc=new Scanner(System.in);

System.out.print("Enter string to search:");

String s=sc.nextLine();

if(list.contains(s)) System.out.println(s+" found");

else System.out.println(s+" not found");

}

}

6. **Iterate using ListIterator**

Write a program to:

* Create a LinkedList of cities.
* Use ListIterator to display the list in both forward and reverse directions.

package day8\_Assign;

import java.util.\*;

public class Q6{

public static void main(String[] args){

LinkedList<String> cities=new LinkedList<>(Arrays.asList("Delhi","Mumbai","Chennai","Kolkata"));

ListIterator<String> i=cities.listIterator();

while(i.hasNext()){

System.out.println(i.next());

}

while(i.hasPrevious()){

System.out.println(i.previous());

}

}

}

7. **Sort a LinkedList**

Write a program to:

* Create a LinkedList of integers.
* Add unsorted numbers.
* Sort the list using Collections.sort().
* Display the sorted list.

package day8\_Assign;

import java.util.\*;

public class Q7{

public static void main(String[] args){

LinkedList<Integer> l1=new LinkedList<>(Arrays.asList(5,2,9,1,6));

Collections.sort(l1);

System.out.println(l1);

}

}

8. **Convert LinkedList to ArrayList**

Write a program to:

* Create a LinkedList of Strings.
* Convert it into an ArrayList.
* Display both the LinkedList and ArrayList.

package day8\_Assign;

import java.util.\*;

public class Q8{

public static void main(String[] args){

LinkedList<String> l1=new LinkedList<>(Arrays.asList("Red","Green","Blue"));

ArrayList<String> l2=new ArrayList<>(l1);

System.out.println("LinkedList: "+l1);

System.out.println("ArrayList:"+l2);

}

}

9. **Store Custom Objects in LinkedList**

Write a program to:

* Create a class Book with fields: id, title, and author.
* Create a LinkedList of Book objects.
* Add 3 books and display their details using a loop.

package day8\_Assign;

import java.util.\*;

class Book{

int id;

String title;

String author;

Book(int id,String title,String author){

this.id=id;

this.title=title;

this.author=author;

}

}

public class Q9{

public static void main(String[] args){

LinkedList<Book> l1=new LinkedList<>();

l1.add(new Book(1,"Java","James Gosling"));

l1.add(new Book(2,"Python","Guido van Rossum"));

l1.add(new Book(3,"C++","Bjarne Stroustrup"));

for(Book b:l1){

System.out.println(b.id+" "+b.title+" "+b.author);

}

}

}

10. **Clone a LinkedList**

Write a program to:

* Create a LinkedList of numbers.
* Clone it using the clone() method.

Display both original and cloned lists

package day8\_Assign;

import java.util.\*;

public class Q10{

public static void main(String[] args){

LinkedList<Integer> l1=new LinkedList<>(Arrays.asList(10,20,30,40,50));

LinkedList<Integer> l2=(LinkedList<Integer>)l1.clone();

System.out.println("Original list:"+l1);

System.out.println("Cloned list:"+l2);

}

}

Vector

* 1. **Create a Vector of integers** and perform the following operations:
* Add 5 integers to the Vector.
* Insert an element at the 3rd position.
* Remove the 2nd element.
* Display the elements using Enumeration.

package day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

Vector<Integer> v=new Vector<>();

v.add(10);

v.add(20);

v.add(30);

v.add(40);

v.add(50);

v.add(2,25);

v.remove(1);

Enumeration<Integer> e=v.elements();

while(e.hasMoreElements()){

System.out.println(e.nextElement());

}

}

}

* 2. **Create a Vector of Strings** and:
* Add at least 4 names.
* Check if a specific name exists in the vector.
* Replace one name with another.
* Clear all elements from the vector.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

Vector<String> v=new Vector<>();

v.add("Alice");

v.add("Bob");

v.add("Charlie");

v.add("David");

if(v.contains("Charlie")){

System.out.println("Charlie exists in the vector");

}

v.set(1,"Brian");

v.clear();

System.out.println("Vector after clearing:"+v);

}

}

* 3. **Write a program** to:
* Copy all elements from one Vector to another Vector.
* Compare both vectors for equality.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

Vector<Integer> v1=new Vector<>(Arrays.asList(10,20,30,40));

Vector<Integer> v2=new Vector<>(v1);

System.out.println(v1);

System.out.println(v2);

System.out.println(v1.equals(v2));

}

}

* 4. **Write a method** that takes a Vector<Integer> and returns the **sum of all elements**.

package day8\_Assign;

import java.util.\*;

public class Q4{

public static void main(String[] args){

Vector<Integer> v=new Vector<>(Arrays.asList(5,10,15,20));

System.out.println("Sum of elements:"+getSum(v));

}

public static int getSum(Vector<Integer> v){

int sum=0;

for(int n:v){

sum+=n;

}

return sum;

}

}

Stack

* 1.
* **Create a Stack of integers** and:
* Push 5 elements.
* Pop the top element.
* Peek the current top.

Check if the stack is emptypackage day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

Stack<Integer> s=new Stack<>();

s.push(10);

s.push(20);

s.push(30);

s.push(40);

s.push(50);

s.pop();

System.out.println(s.peek());

System.out.println(s.isEmpty());

}

}

* 2. **Reverse a string using Stack**:
* Input a string from the user.
* Use a stack to reverse and print the string.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.print("Enter a string: ");

String str=sc.nextLine();

Stack<Character> s=new Stack<>();

for(char c : str.toCharArray()){

s.push(c);

}

StringBuilder reversed=new StringBuilder();

while(!s.isEmpty()){

reversed.append(s.pop());

}

System.out.println("Reversed string: "+reversed);

sc.close();

}

}

* 3. **Use Stack to check for balanced parentheses** in an expression.
* Input: (a+b) \* (c-d)
* Output: Valid or Invalid expression

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.print("Enter expression: ");

String expr=sc.nextLine();

Stack<Character> s=new Stack<>();

boolean isValid=true;

for(char c:expr.toCharArray()){

if(c=='(') s.push(c);

else if(c==')'){

if(s.isEmpty()){

isValid=false;

break;

}

s.pop();

}

}

if(!s.isEmpty()) isValid=false;

System.out.println(isValid ? "Valid expression" : "Invalid expression");

sc.close();

}

}

* 4. **Convert a decimal number to binary using Stack**.

package day8\_Assign;

import java.util.\*;

public class Q4{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.print("Enter decimal number:");

int num=sc.nextInt();

Stack<Integer> s=new Stack<>();

if(num == 0) s.push(0);

while(num > 0){

s.push(num % 2);

num /= 2;

}

System.out.print("Binary:");

while(!s.isEmpty()){

System.out.print(s.pop());

}

sc.close();

}

}

HashSet

1. 1. **Create a HashSet of Strings**:
   * Add 5 different city names.
   * Try adding a duplicate city and observe the output.
   * Iterate using an Iterator and print each city.

package day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

HashSet<String> cities=new HashSet<>();

cities.add("Delhi");

cities.add("Mumbai");

cities.add("Chennai");

cities.add("Kolkata");

cities.add("Hyderabad");

System.out.println(cities.add("Delhi"));

Iterator<String> i=cities.iterator();

while(i.hasNext()){

System.out.println(i.next());

}

}

}

1. 2. **Perform operations**:
   * Remove an element.
   * Check if a city exists.

Clear the entire HashSet

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

HashSet<String> cities=new HashSet<>(Arrays.asList("Delhi","Mumbai","Chennai","Kolkata"));

cities.remove("Mumbai");

System.out.println(cities);

System.out.println(cities.contains("Chennai"));

cities.clear();

System.out.println(cities);

}

}

3. **Write a method** that takes a HashSet<Integer> and returns the maximum element

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

HashSet<Integer> nums=new HashSet<>(Arrays.asList(10,20,5,35,15));

System.out.println(getMax(nums));

}

public static int getMax(HashSet<Integer> set){

int max=Integer.MIN\_VALUE;

for(int n:set){

if(n>max) max=n;

}

return max;

}

}

LinkedHashSet

1. **Create a LinkedHashSet of Integers**:

* + Add numbers: 10, 5, 20, 15, 5.
  + Print the elements and observe the order.

package day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

LinkedHashSet<Integer> s=new LinkedHashSet<>();

s.add(10);

s.add(5);

s.add(20);

s.add(15);

s.add(5);

System.out.println(s);

}

}

1. 2. **Create a LinkedHashSet of custom objects (e.g., Student with id and name)**:
   * Override hashCode() and equals() properly.
   * Add at least 3 Student objects.
   * Try adding a duplicate student and check if it gets added.

package day8\_Assign;

import java.util.\*;

class Student{

int id;

String name;

Student(int id,String name){

this.id=id;

this.name=name;

}

public boolean equals(Object o){

if(this==o) return true;

if(o==null || getClass()!=o.getClass()) return false;

Student s=(Student)o;

return id==s.id && name.equals(s.name);

}

public int hashCode(){

return Objects.hash(id,name);

}

public String toString(){

return id+" "+name;

}

}

public class Q2{

public static void main(String[] args){

LinkedHashSet<Student> s=new LinkedHashSet<>();

s.add(new Student(1,"A"));

s.add(new Student(2,"B"));

s.add(new Student(3,"C"));

s.add(new Student(1,"D")); // duplicate

for(Student st:s){

System.out.println(st);

}

}

}

1. 3. **Write a program** to:
   * Merge two LinkedHashSets and print the result.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

LinkedHashSet<Integer> s1=new LinkedHashSet<>(Arrays.asList(10,20,30));

LinkedHashSet<Integer> s2=new LinkedHashSet<>(Arrays.asList(40,50,20));

s1.addAll(s2);

System.out.println(s1);

}

}

TreeSet

1. **TreeSet**

**1. Create a TreeSet of Strings**:

* + Add 5 country names in random order.

Print the sorted list of countries using TreeSetpackage day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

TreeSet<String> countries=new TreeSet<>();

countries.add("India");

countries.add("USA");

countries.add("China");

countries.add("Brazil");

countries.add("Australia");

System.out.println(countries);

}

}

.

**2.Create a TreeSet of Integers**:

* + Add some numbers and print the first and last elements.
  + Find the elements lower than and higher than a given number using lower() and higher() methods.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

TreeSet<Integer> nums=new TreeSet<>(Arrays.asList(10,5,20,15,25));

System.out.println(nums.first());

System.out.println(nums.last());

System.out.println(nums.lower(15));

System.out.println(nums.higher(15));

}

}

1. 3. **Create a TreeSet with a custom comparator**:

Sort strings in **reverse alphabetical order** using Comparatorpackage day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

TreeSet<String> s=new TreeSet<>(new Comparator<String>(){

public int compare(String a,String b){

return b.compareTo(a);

}

});

s.add("Apple");

s.add("Banana");

s.add("Cherry");

s.add("Date");

s.add("Elderberry");

System.out.println(s);

}

}

Queue

1. 1. **Bank Queue Simulation**:
   * Create a queue of customer names using Queue<String>.
   * Add 5 customers to the queue.

Serve (remove) customers one by one and print the queue after each removalpackage day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

Queue<String> q=new LinkedList<>();

q.add("Alice");

q.add("Bob");

q.add("Charlie");

q.add("David");

q.add("Eve");

while(!q.isEmpty()){

System.out.println(q);

q.poll();

}

}

}

1. 2. **Task Manager**:
   * Queue of tasks (String values).
   * Add tasks, peek at the next task, and poll completed tasks.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

Queue<String> tasks=new LinkedList<>();

tasks.add("Task1");

tasks.add("Task2");

tasks.add("Task3");

System.out.println(tasks.peek());

tasks.poll();

System.out.println(tasks);

}

}

1. 3. **Write a method**:
   * That takes a queue of integers and returns a list of even numbers.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

Queue<Integer> q=new LinkedList<>(Arrays.asList(1,2,3,4,5,6,7,8));

List<Integer> evens=getEvenNumbers(q);

System.out.println(evens);

}

public static List<Integer> getEvenNumbers(Queue<Integer> q){

List<Integer> res=new ArrayList<>();

for(int n:q){

if(n%2==0)

{

res.add(n);

}

}

return res;

}

}

PriorityQueue

1. **Hospital Emergency Queue**:
   * Create a class Patient with fields: name and severityLevel (int).
   * Use PriorityQueue<Patient> with a comparator to serve the most critical patients first (highest severityLevel).

1.

package day8\_Assign;

import java.util.\*;

class Patient{

String name;

int severityLevel;

Patient(String name,int severityLevel){

this.name=name;

this.severityLevel=severityLevel;

}

}

public class Q1{

public static void main(String[] args){

PriorityQueue<Patient> pq=new PriorityQueue<>(new Comparator<Patient>(){

public int compare(Patient p1,Patient p2){

return p2.severityLevel - p1.severityLevel;

}

});

pq.add(new Patient("Alice",5));

pq.add(new Patient("Bob",9));

pq.add(new Patient("Charlie",3));

while(!pq.isEmpty()){

Patient p=pq.poll();

System.out.println(p.name+" "+p.severityLevel);

}

}

}

1. 2. **Print Jobs Priority**:
   * Add different print jobs (String) with priority levels.

Use PriorityQueue to simulate serving high-priority jobs before otherspackage day8\_Assign;

import java.util.\*;

class PrintJob{

String jobName;

int priority;

PrintJob(String jobName,int priority){

this.jobName=jobName;

this.priority=priority;

}

}

public class Q2{

public static void main(String[] args){

PriorityQueue<PrintJob> pq=new PriorityQueue<>(new Comparator<PrintJob>(){

public int compare(PrintJob j1,PrintJob j2){

return j2.priority-j1.priority;

}

});

pq.add(new PrintJob("Doc1",3));

pq.add(new PrintJob("Doc2",5));

pq.add(new PrintJob("Doc3",1));

while(!pq.isEmpty()){

PrintJob job=pq.poll();

System.out.println(job.jobName+" "+job.priority);

}

}

}

1. 3. **Write a method**:
   * To merge two PriorityQueue<Integer> and return a sorted merged queue.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

PriorityQueue<Integer> pq1=new PriorityQueue<>(Arrays.asList(10,30,20));

PriorityQueue<Integer> pq2=new PriorityQueue<>(Arrays.asList(25,15,5));

PriorityQueue<Integer> merged=mergeQueues(pq1,pq2);

while(!merged.isEmpty()){

System.out.print(merged.poll()+" ");

}

}

public static PriorityQueue<Integer> mergeQueues(PriorityQueue<Integer> q1,PriorityQueue<Integer> q2){

PriorityQueue<Integer> result=new PriorityQueue<>();

result.addAll(q1);

result.addAll(q2);

return result;

}

}

Deque

1. **Palindrome Checker**:
   * Input a string and check if it is a palindrome using a Deque<Character>.

package day8\_Assign;

import java.util.\*;

public class Q1{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

System.out.print("Enter a string: ");

String str=sc.nextLine();

Deque<Character> dq=new LinkedList<>();

for(char c:str.toCharArray()){

dq.addLast(c);

}

boolean isPalindrome=true;

while(dq.size()>1){

if(dq.removeFirst()!=dq.removeLast()){

isPalindrome=false;

break;

}

}

System.out.println(isPalindrome);

sc.close();

}

}

2. **Double-ended Order System**:

* + Add items from front and rear.
  + Remove items from both ends.
  + Display contents of the deque after each operation.

package day8\_Assign;

import java.util.\*;

public class Q2{

public static void main(String[] args){

Deque<String> dq=new LinkedList<>();

dq.addFirst("Item1");

System.out.println(dq);

dq.addLast("Item2");

System.out.println(dq);

dq.addFirst("Item3");

System.out.println(dq);

dq.removeFirst();

System.out.println(dq);

dq.removeLast();

System.out.println(dq);

}

}

3. **Browser History Simulation**:

* + Implement browser back and forward navigation using two deques.

package day8\_Assign;

import java.util.\*;

public class Q3{

public static void main(String[] args){

Deque<String> b=new LinkedList<>();

Deque<String> f=new LinkedList<>();

b.push("Page1");

b.push("Page2");

b.push("Page3");

System.out.println(b.peek());

f.push(b.pop());

System.out.println(b.peek());

System.out.println(f.peek());

b.push(f.pop());

System.out.println(b.peek());

}

}