TECH-RANCH

PRACTICE PROJECTS

(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

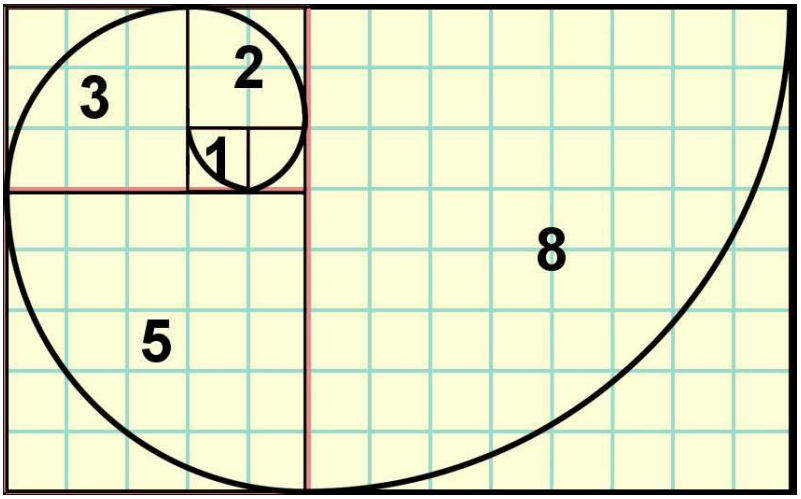
BASIC JAVA PRACTICE PROJECTS

1. Fibonacci
2. Palindrome
3. Prime Number
4. Factorial
5. Decimal to Binary Number Converter
6. Binary Searching
7. Bubble Sorting
8. Star/Number Pyramid Pattern
9. Permutation & Combination
10. Stack Implementation

ADVANCE JAVA PRACTICE PROJECTS

1. Simple Chat Application
2. Bar Code Generator
3. Bar Code Reader
4. Simple & Scientific Calculator
5. Tic – Tac – Toe Game
6. Body Mass Index Calculator
7. Equity Monthly Income Calculator
8. Online Random Password
9. Online Exam Tool
10. Online Scale Converter
11. File Converter
12. Audio / Video Player
13. Personal Budget Database Management
14. Graphical Report Generator
15. Export Oracle to MS Excel Database

JAVA PRACTICE PROJECT

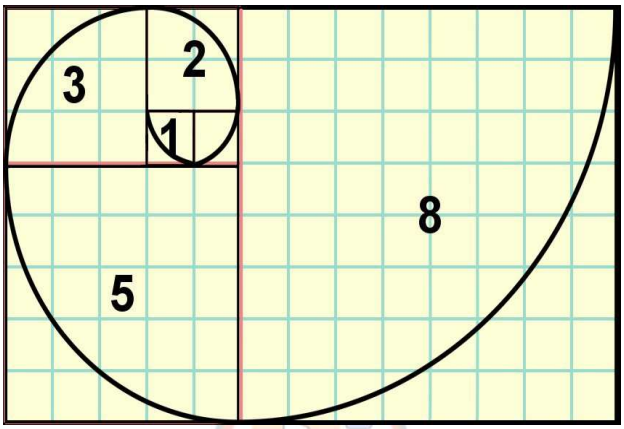


FIBONACCI

Let's begin.....

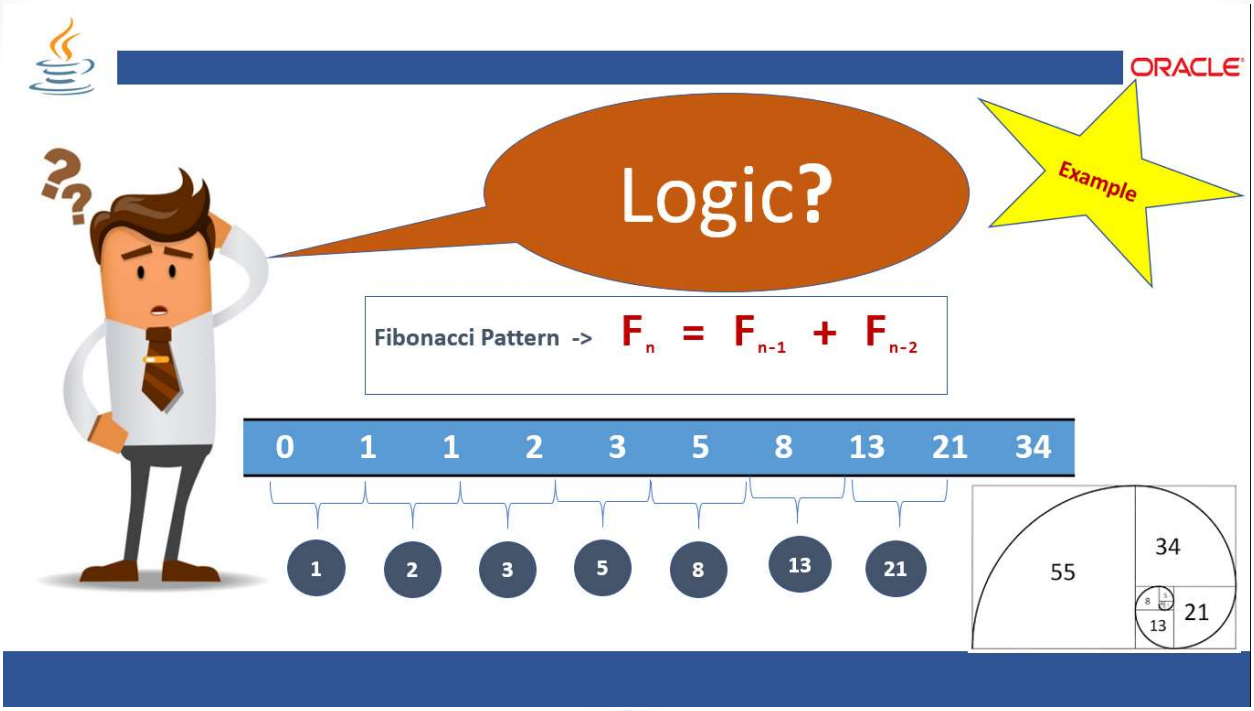
OBJECTIVE

Tech-Ranch presents how to implement **Fibonacci Series** in Java using control statements and arrays.



REQUIREMENT

Fibonacci sequence is a **sequence** of **numbers** in which each successive number in the **sequence** is obtained by adding the two previous **numbers** in the **series**.



Fibonacci sequence is one of the most famous formulas in mathematics. Each number in the **sequence** is the sum of the two **numbers** that precede it. So, the **sequence** goes: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on

IMPLEMENTATION

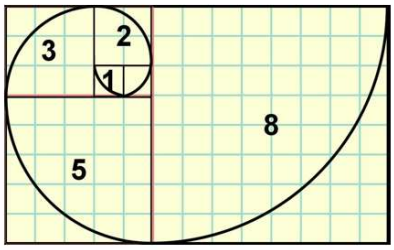
There are so many ways to implement Fibonacci Series. One of the simplest ways is using Recursion.

Solution – Fibonacci using Recursion

ORACLE

```
static int fib(int n)
    {
    if (n <= 1)
       return n;
    return fib(n-1) + fib(n-2);
    }
public static void main(String args[]){
int n = 9;
System.out.println(fib(n));
}
```

Recursion means auto iterated or self repeated loop



Time Complexity: T(n) = T(n-1) + T(n-2) , exponential behavior

For Practice, try to implement using do-While, for loop or While loop

CONCLUSION

Fibonacci Series is one of the most important algorithms to program machines and learn about their process and performance. There are number of ways to implement in all kind of programming languages. Here, we have shown to implement Fibonacci series in Java using control statement and array.

TECH-RANCH

PRACTICE PROJECTS

(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

TECH-RANCH

JAVA PRACTICE PROJECT

"MADAM" "MOM"
"NOON" "RACECAR"
"LEVEL" "KAYAK" "CIVIC"
"WOW" "REFER"
"MYGYM"

PALINDROME

Let's begin.....

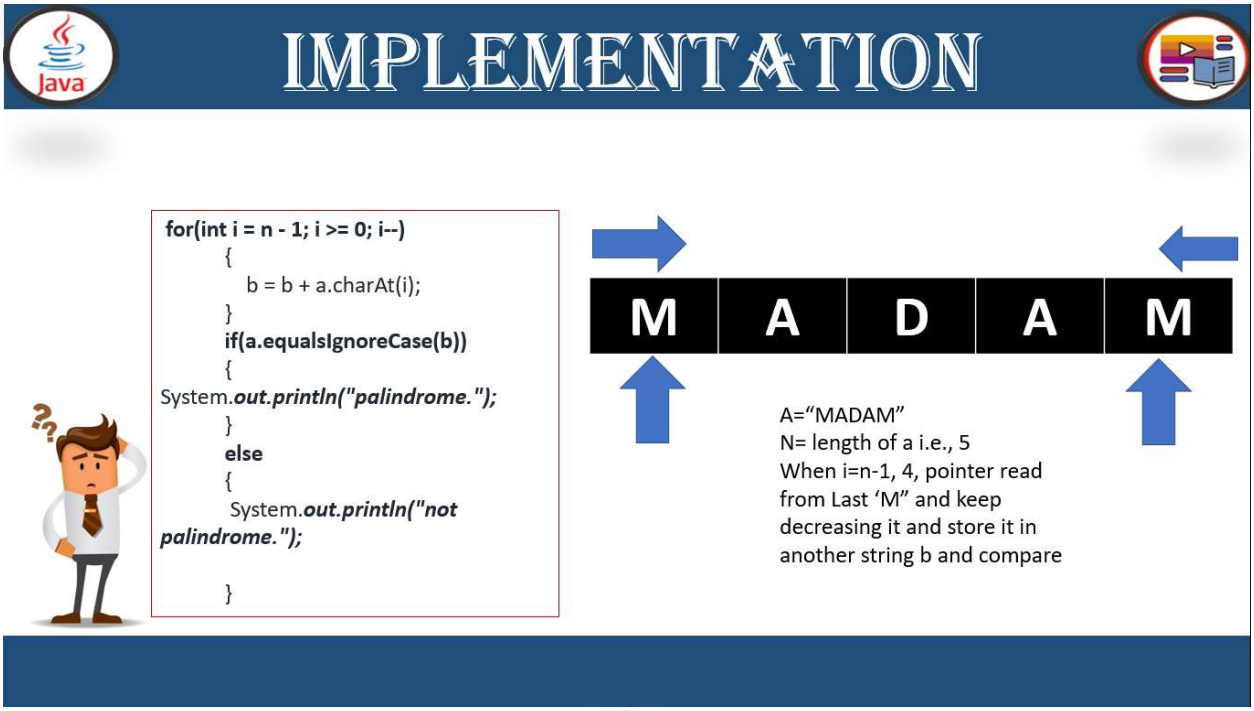
OBJECTIVE

Tech-Ranch presents how to implement **Palindrome** in Java using control statements and arrays.

"MADAM" "MOM"
"NOON" "RACECAR"
"LEVEL" "KAYAK" "CIVIC"
"WOW" "REFER"

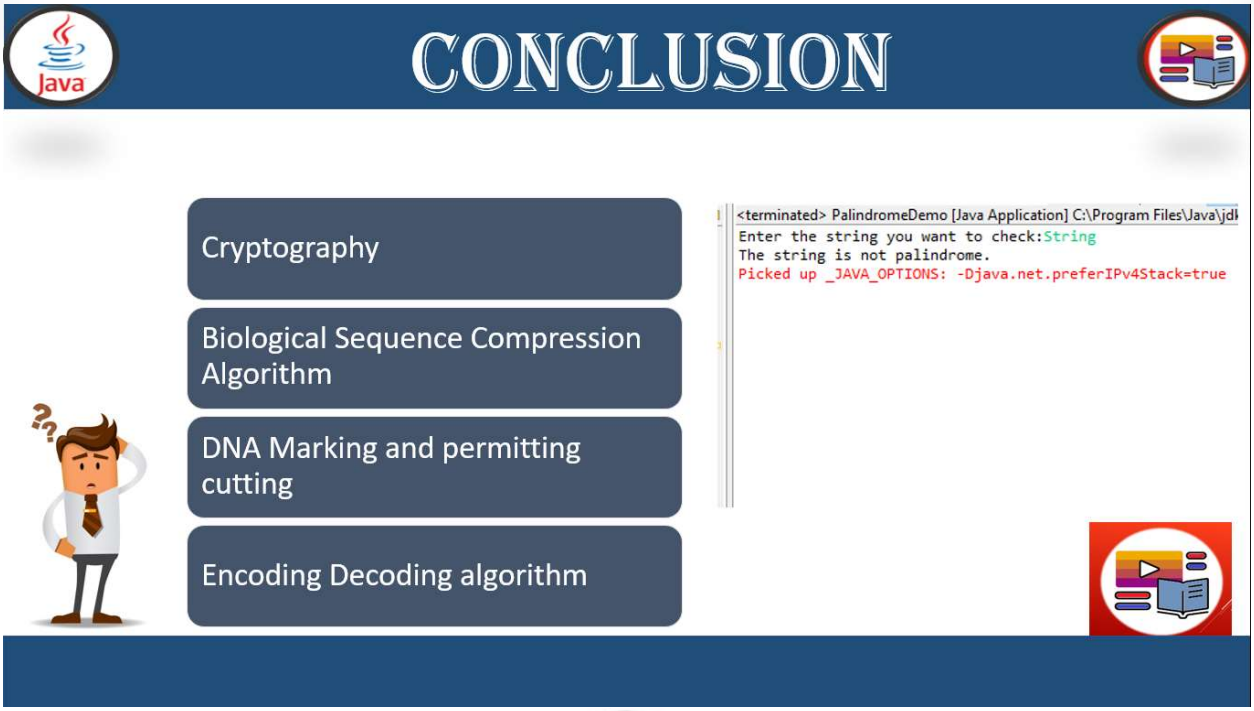
REQUIREMENT

Palindrome is string that reads the same backwards as forwards. For example:



IMPLEMENTATION

There are so many ways to implement Palindrome. One of the simplest ways is using String and array of characters with control statements.



CONCLUSION

Palindrome is one of the most important algorithms to program machines and learn about their process and performance. There are number of ways to implement in all kind of programming languages. Here, we have shown to implement **Palindrome** in Java using control statement and array.

TECH-RANCH

PRACTICE PROJECTS

(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

TECH-RANCH

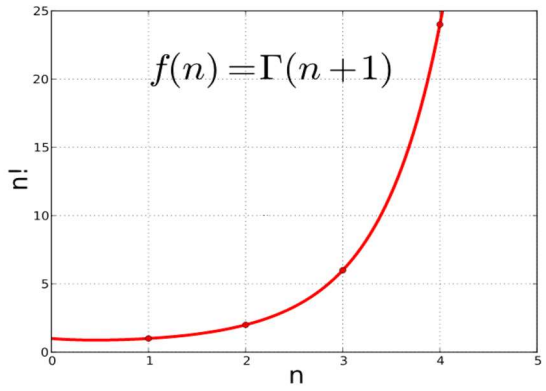
JAVA PRACTICE PROJECT

FACTORIAL

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to implement **Factorial** in Java using recursion, control statements and arrays.



REQUIREMENT

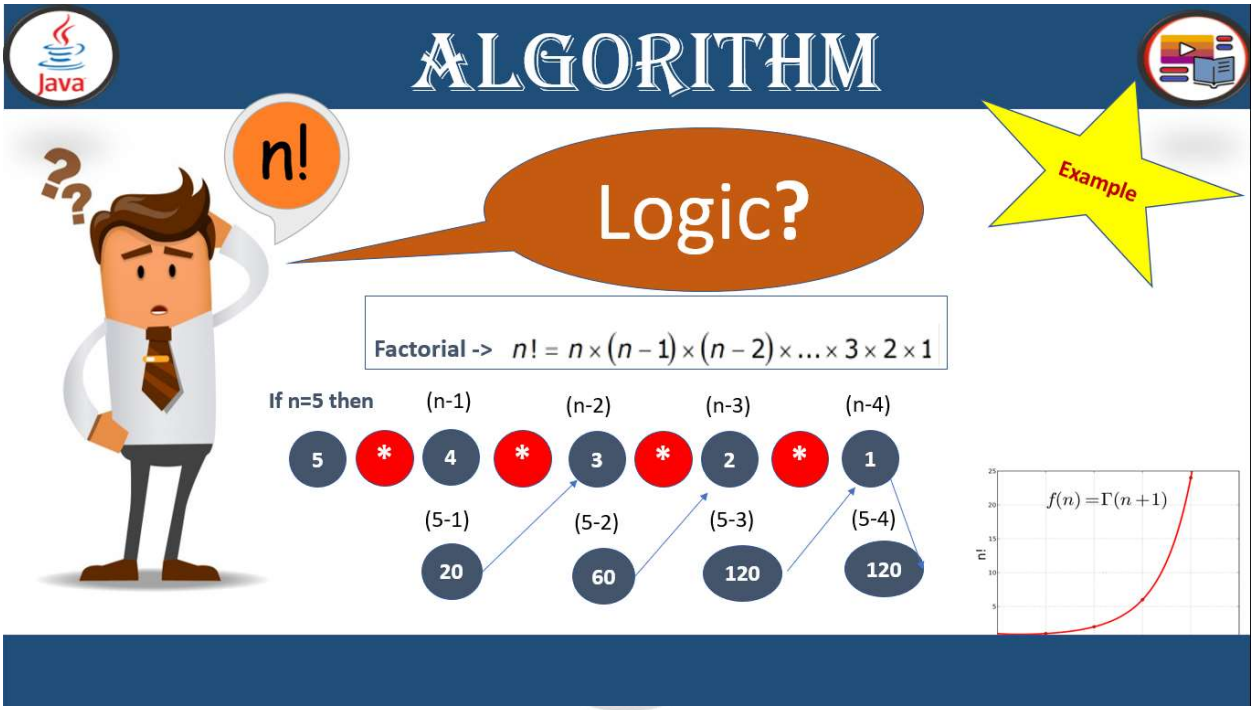
In mathematics, the factorial of a positive integer n, denoted by n!, is the product of all positive integers less than or equal to n: For example, The value of 0! is 1, according to the convention for an empty product..

Factorial of n is the *product of all positive descending integers*.
Factorial of *n* is denoted by n!

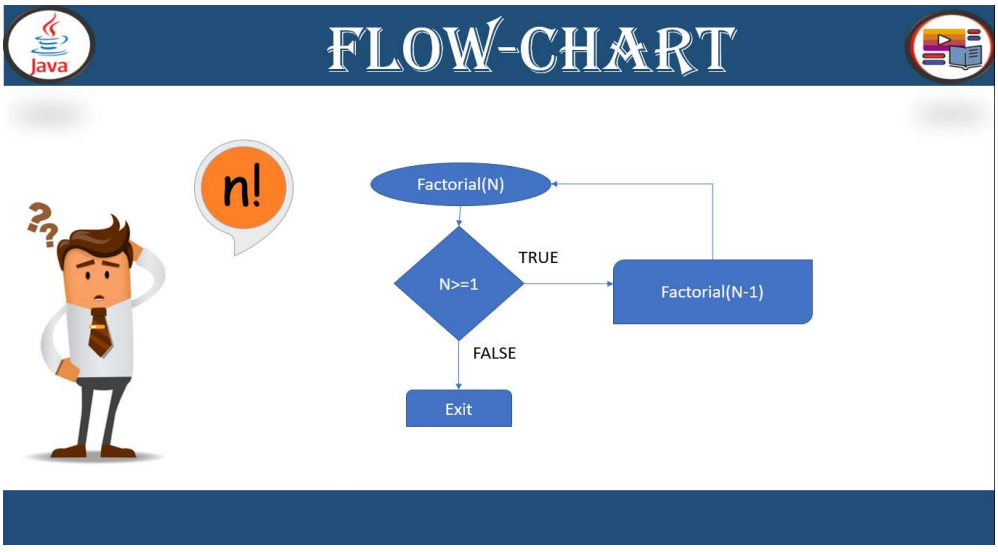
In the simple words,

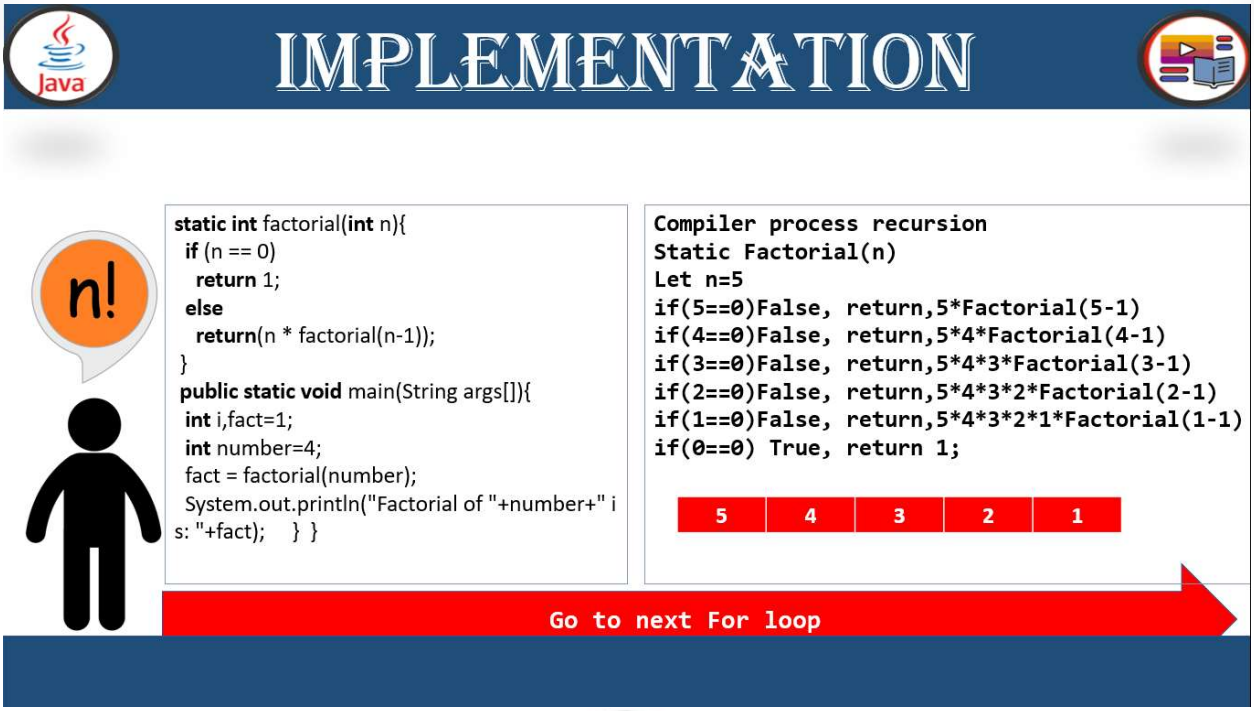
$$n! = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1$$

IMPLEMENTATION



There are so many ways to implement Factorial. One of the simplest ways is using array and control statements.





CONCLUSION

The product of an integer and all the integers below it; e.g. factorial four (*4!*) is equal to 24.. **In this session, we have seen how to implement Factorial using Control statements and recursion in Java.**

PRACTICE PROJECTS

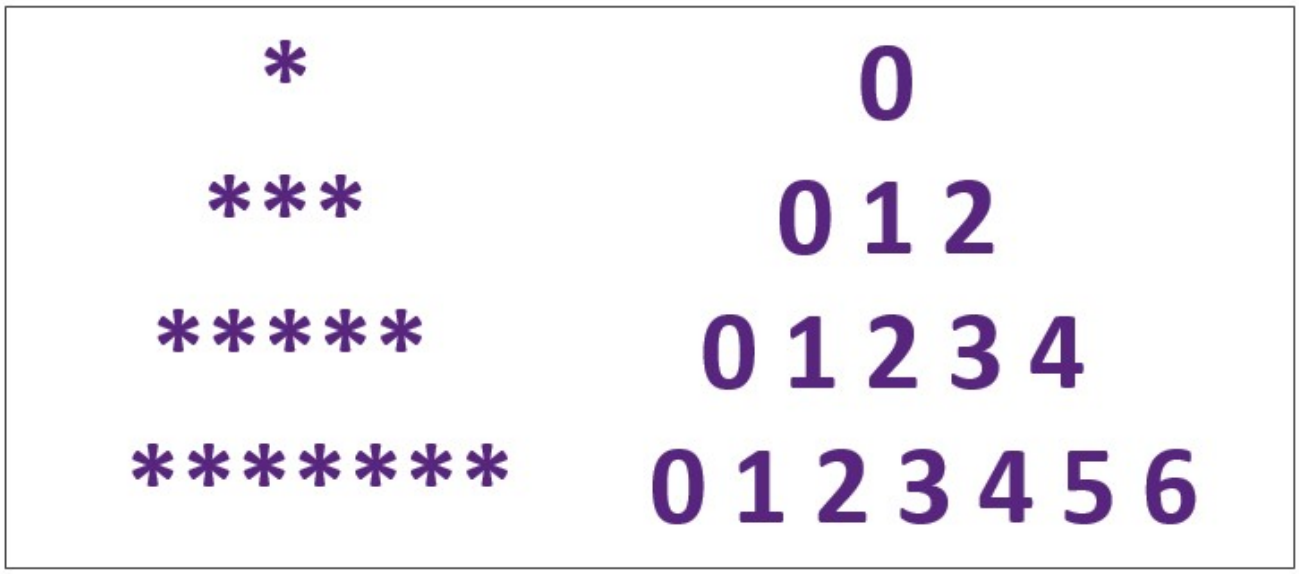
(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT

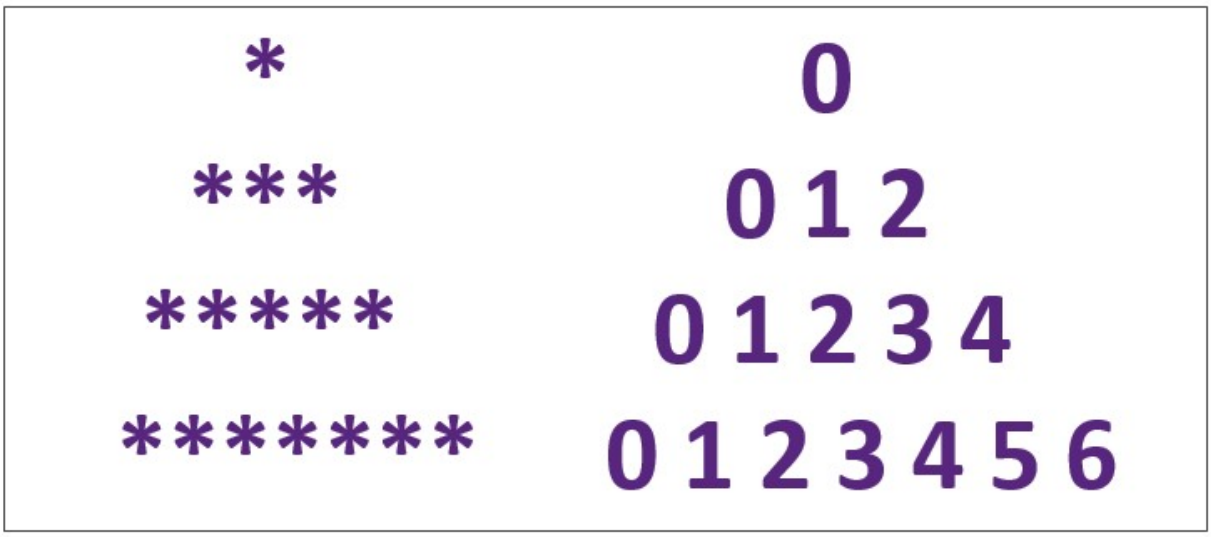


STAR/NUMBER PATTERN

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to Star /Number Pattern in Java using recursion, control statements and arrays.

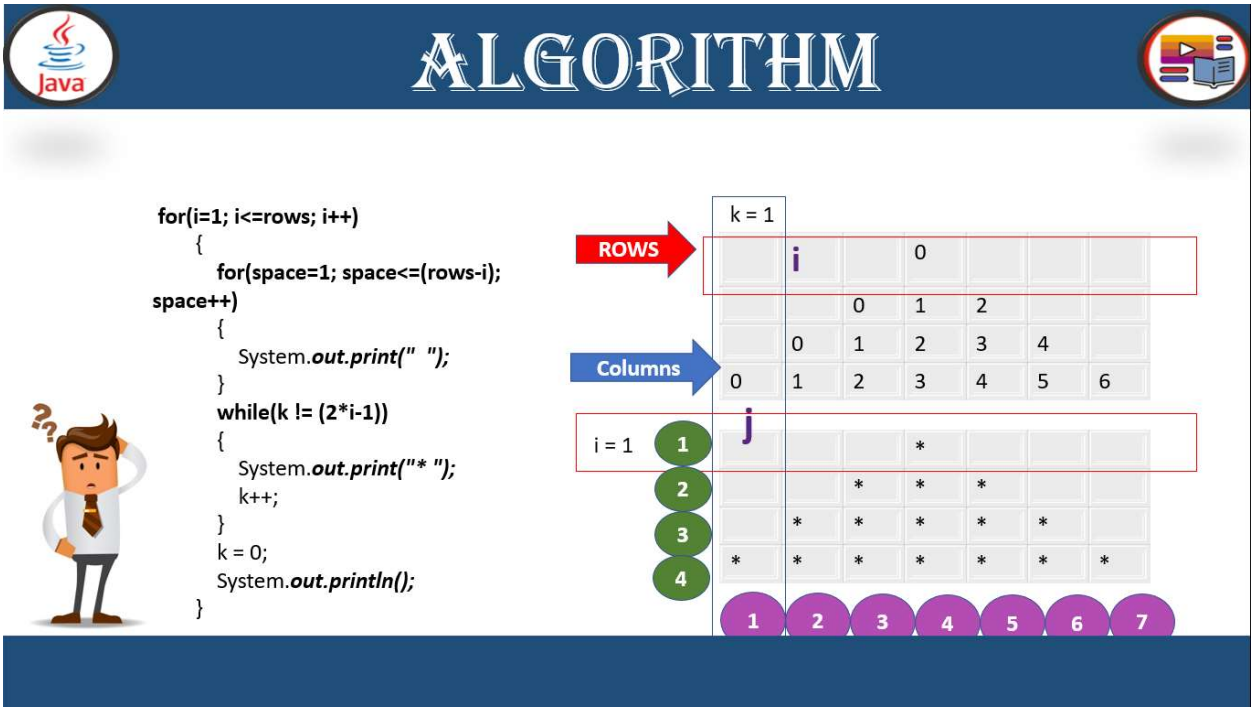


REQUIREMENT

For pattern printing in any programming language, can be done by practicing control statements and make logic for implementation. There are so many ways to implement such requirement.

IMPLEMENTATION

There are so many ways to implement Decimal to Binary and Binary to Decimal Conversion. One of the simplest ways is using array and control statements.



CONCLUSION

Star/Number Pattern can be implemented in Java for practice making logics and understand basics programming concepts. **In this session, we have seen how to implement Star/Number Pattern using Control statements and recursion in Java.**

PRACTICE PROJECTS

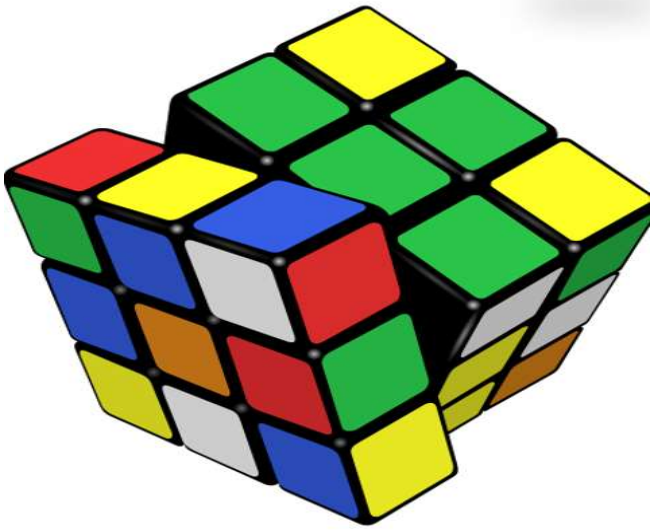
(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT



PERMUTATION & COMBINATION

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to Permutation and Combination using recursion, control statements and arrays.

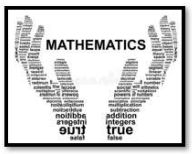
REQUIREMENT

In mathematics, **Permutation** is the act of arranging the members of a set into a sequence or order, or, if the set is already ordered, rearranging its elements—a process called permuting. Permutations differ from combinations, which are selections of some members of a set regardless of order.

Combination is a selection of items from a collection, such that (unlike permutations) the order of selection does not matter.

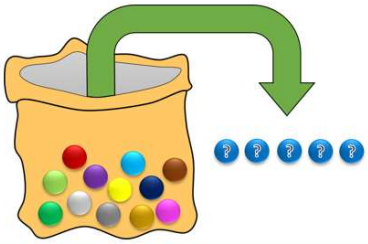
For Permutation & Combination in any programming language, can be done by practicing control statements and make logic for implementation. There are so many ways to implement such requirement.

PERMUTATION

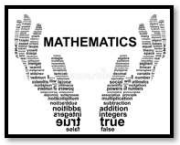


$$ {}_nP_r = \frac{n!}{(n-r)!} $$

Where
N = number of things
R = times

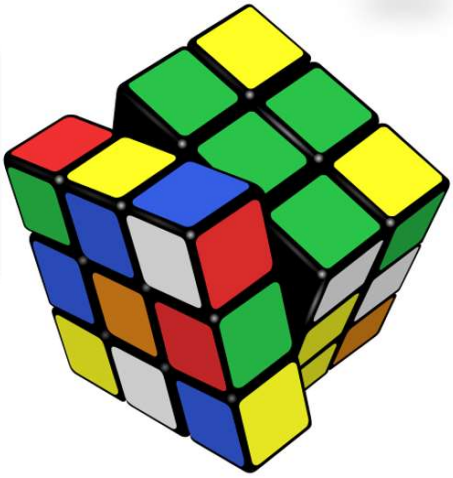


COMBINATION



$$ {}_nC_r = \frac{n!}{(n-r)!r!} $$

Where
N = number of things
R = times



IMPLEMENTATION

There are so many ways to implement Permutation & Combination. One of the simplest ways is using array and control statements.

```
public class PermutationCombinationDemo
{
static void printPermutation(String str , String ans){
        if(str.length() == 0){
            System.out.println(ans +" ");
            return;                }
        for(int i =0; i<str.length();i++){
            char ch=str.charAt(i);
            String r = str.substring(0,i)+str.substring(i+1);
            printPermutation(r,ans+ch);
        }       }
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter string:");
        String str = input.next();
        printPermutation(str, " ");
            }   }
```

CONCLUSION

. **In this session, we have seen how to implement Permutation & Combination using Control statements and recursion in Java.**

PRACTICE PROJECTS

(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT

Decimal	Binary	Hexadecimal
63	00111111	3F
250	11111010	FA
117	01110101	75
220	11011100	DC
171	10101011	AB
94	01011110	5E

DECIMAL BINARY CONVERTER

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to implement **Decimal – Binary Number Conversion** in Java using recursion, control statements and arrays.

Decimal	Binary	Hexadecimal
63	00111111	3F
250	11111010	FA
117	01110101	75
220	11011100	DC
171	10101011	AB
94	01011110	5E

REQUIREMENT

In mathematics, Number system helps to program machines and compute all numbers. The binary system is the internal language of electronic computers. If you are a serious computer programmer, you should understand how to convert from binary to decimal.

1.Write down the binary number.

2 Starting from the left, double your previous total and add the current digit.

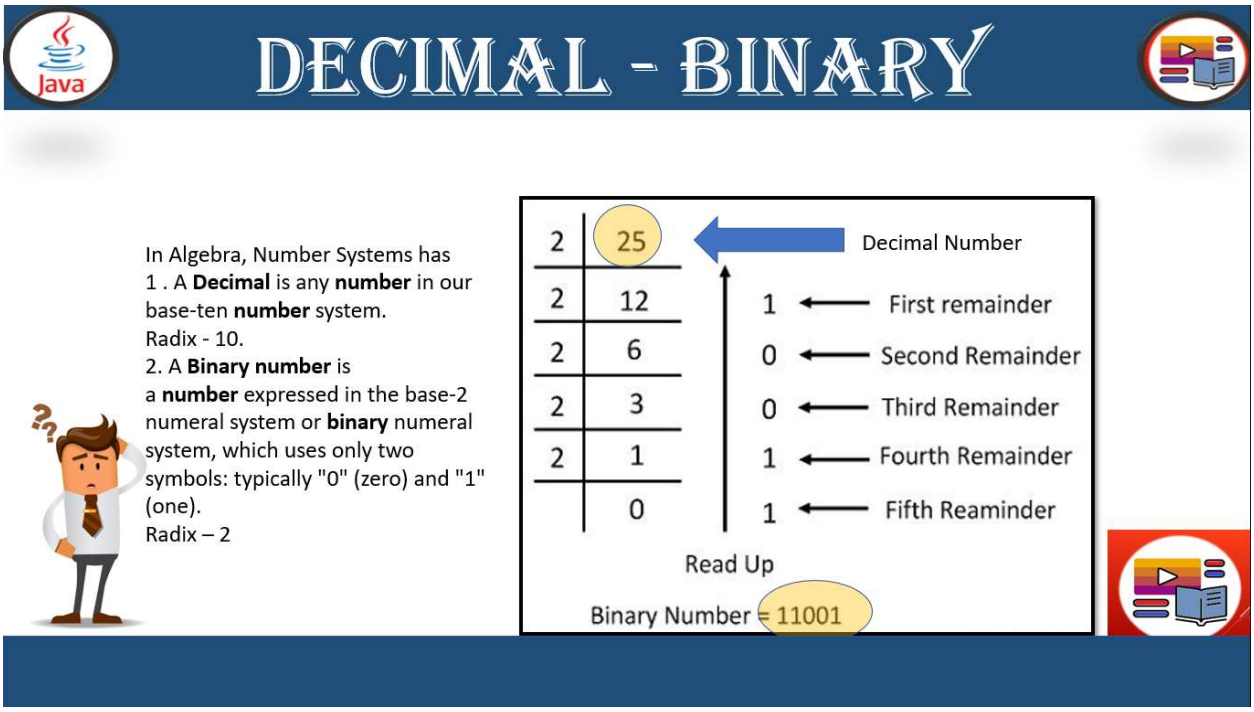
3. Double your current total and add the next leftmost digit.

4. Repeat the previous step

5. Continue doubling your current total and adding the next digit until you've run out of digits

6. Write the answer along with its base subscript.

7. Use this method to convert from *any* base to decimal.



BINARY - DECIMAL

Convert From Binary to Decimal

```
128 64 32 16  8  4  2  1 weight
 0  0  1  0  0  1  1  0
       32    +  4 + 2  =  38 decimal

128 64 32 16  8  4  2  1 weight
 1  1  0  0  1  1  0  1
128 + 64    +   8 + 4  +  1  =  205 decimal
```

Powers of 2

2^0	= 1
2^1	= 2
2^2	= 4
2^3	= 8
2^4	= 16
2^5	= 32
2^6	= 64
2^7	= 128
2^8	= 256

NUMBER SYSTEM

Practice Project
- Converted to other number system like HexaDecimal

Demonstration
- Implemented in Java

Uses
- Machines can be programmed with different types of number systems

Decimal	Binary	Hexadecimal
63	00111111	3F
250	11111010	FA
117	01110101	75
220	11011100	DC
171	10101011	AB
94	01011110	5E

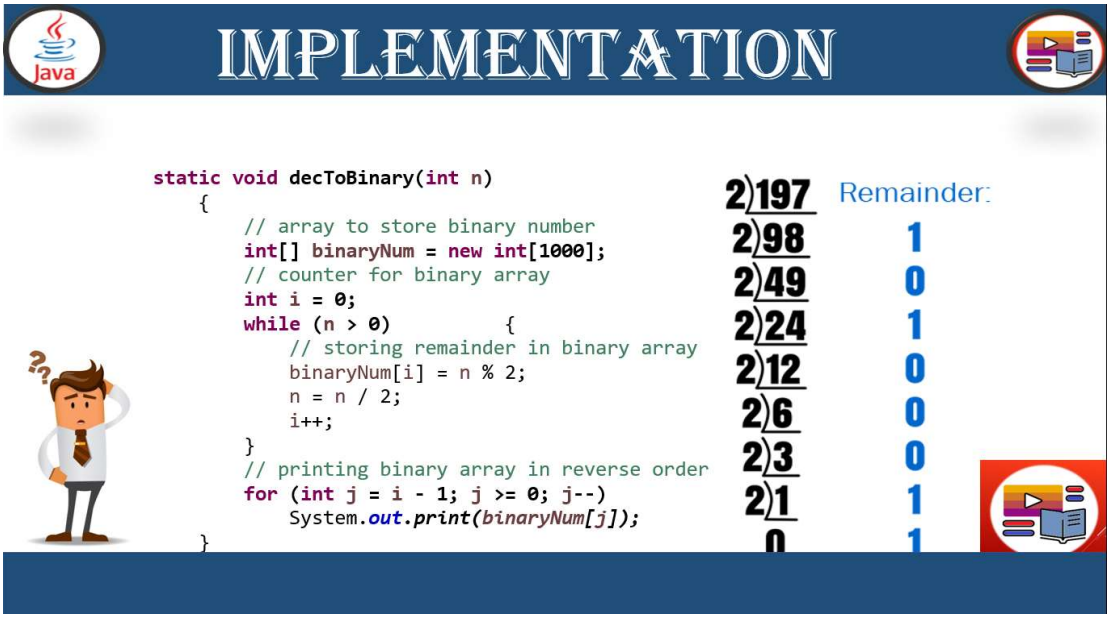
Please contact techranch2019@gmail.com

\

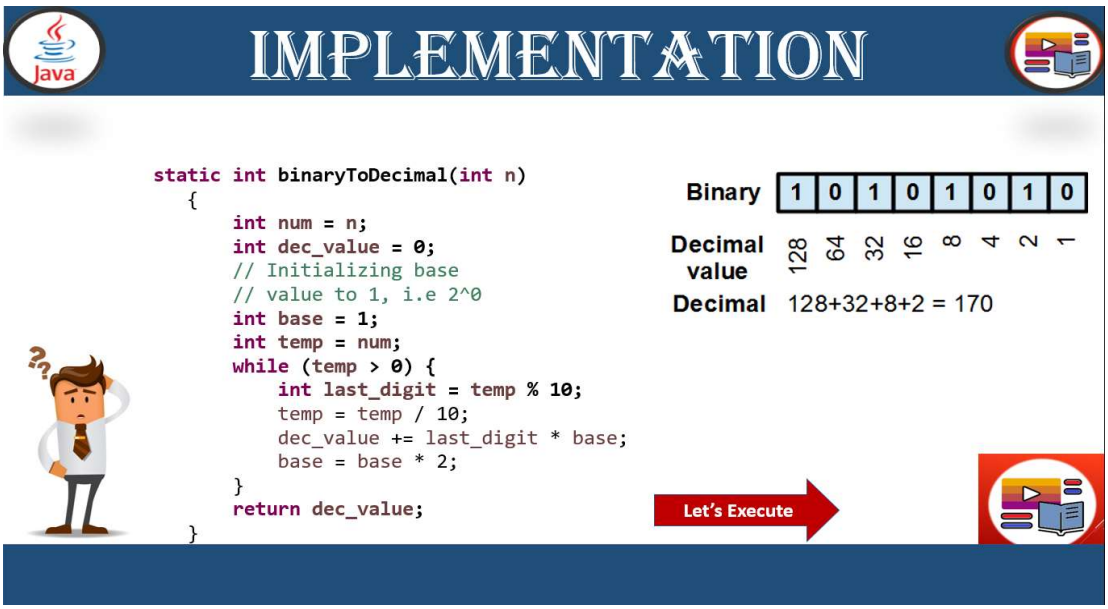
IMPLEMENTATION

There are so many ways to implement Decimal to Binary and Binary to Decimal Conversion. One of the simplest ways is using array and control statements.

DECIMAL to BINARY



BINARY to DECIMAL



CONCLUSION

The decimal (base ten) numeral system has ten possible values (0,1,2,3,4,5,6,7,8, or 9) for each place-value. In contrast, the binary (base two) numeral system has two possible values represented as 0 or 1 for each place-value. **In this session, we have seen how to implement Decimal – Binary Conversion using Control statements and recursion in Java.**

TECH-RANCH

PRACTICE PROJECTS

(Java Practice Projects)

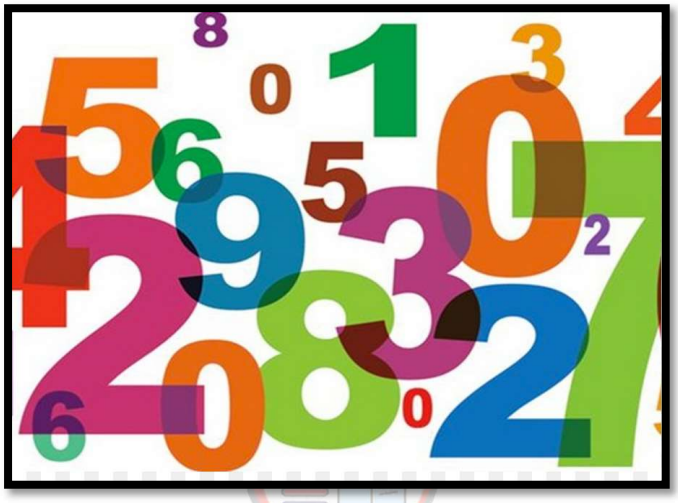
Presented by

Anjali Singh

Let's make coding fun!

TECH-RANCH

JAVA PRACTICE PROJECT

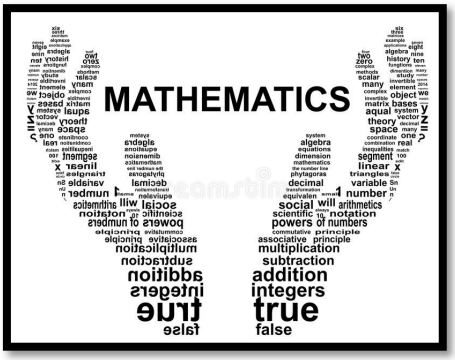


PRIME NUMBER

Let's begin.....

OBJECTIVE

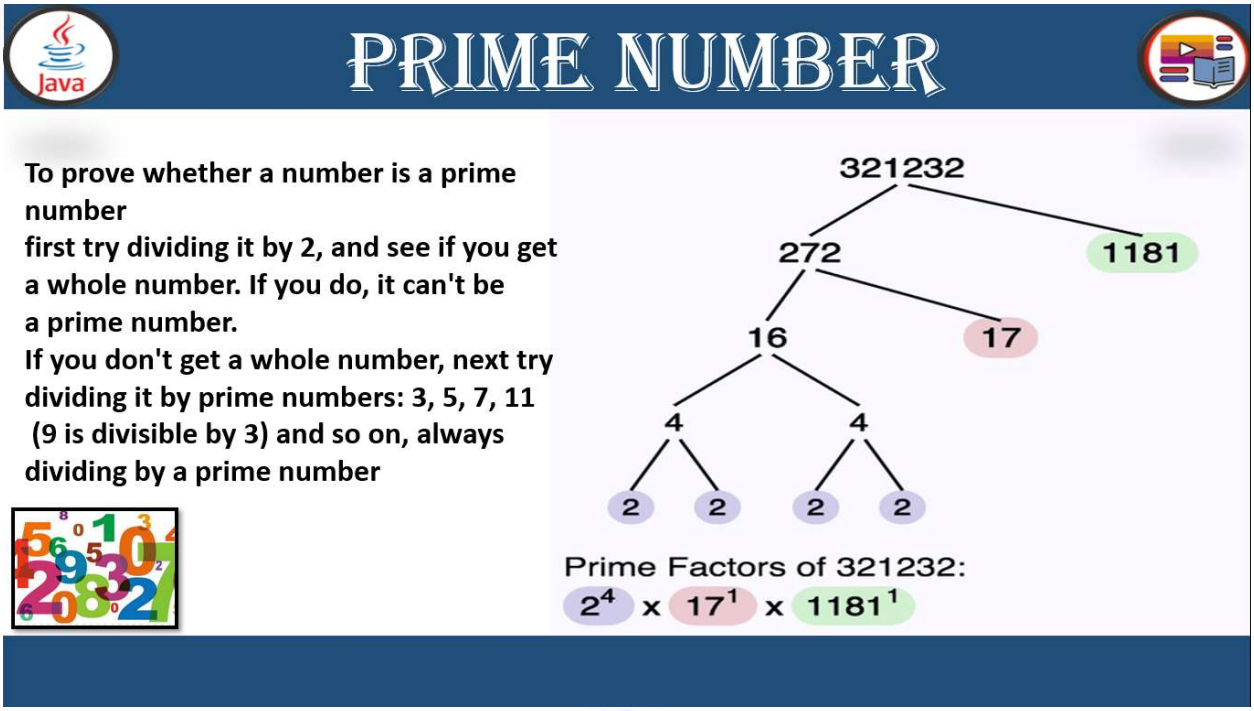
Tech-Ranch presents how to implement **Prime Number** in Java using control statements and arrays.



REQUIREMENT

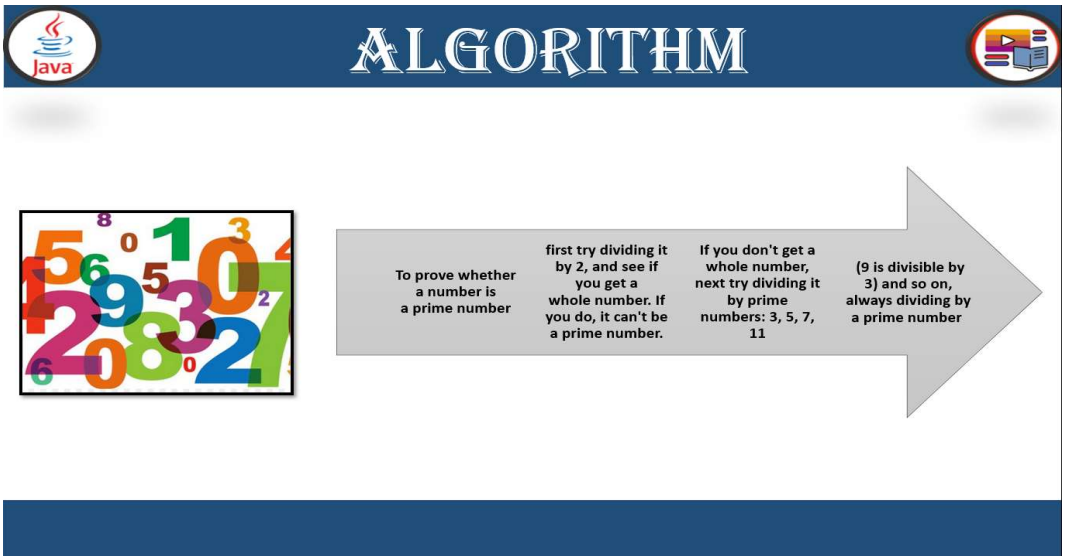
A prime number is a whole number greater than 1 whose only factors are 1 and itself. A factor is a whole number that can be divided evenly into another number. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29. Numbers that have more than two factors are called composite numbers. The number 1 is neither prime nor composite.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

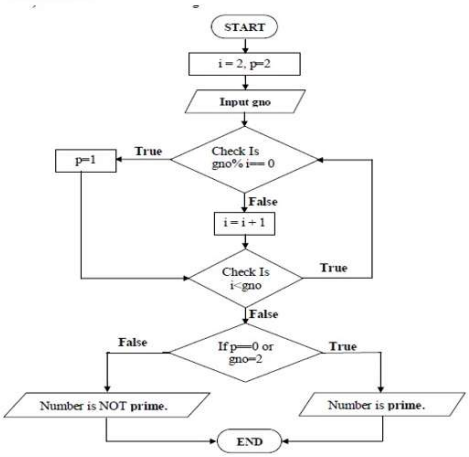


IMPLEMENTATION

There are so many ways to implement Prime Number. One of the simplest ways is using array and control statements.



IMPLEMENTATION




```
void primeNumber(){
Scanner in = new Scanner(System.in);
System.out.println("Enter number");
int n = in.nextInt();
int I,m=0,flag=0;
  m=n/2;
  if(n==0||n==1){
   System.out.println(n+" is not prime number");
  }else{
   for(i=2;i<=m;i++){
    if(n%i==0){
     System.out.println(n+" is not prime number");
     flag=1;
     break;    }
   }
   if(flag==0)  { System.out.println(n+" is prime number");
  }//end of else
}
```

CONCLUSION

Prime Number is one of the most important algorithms to program machines and learn about their process and performance. There are number of ways to implement in all kind of programming languages. Here, we have shown to implement **Prime Number** in Java using control statement and array.

PRACTICE PROJECTS

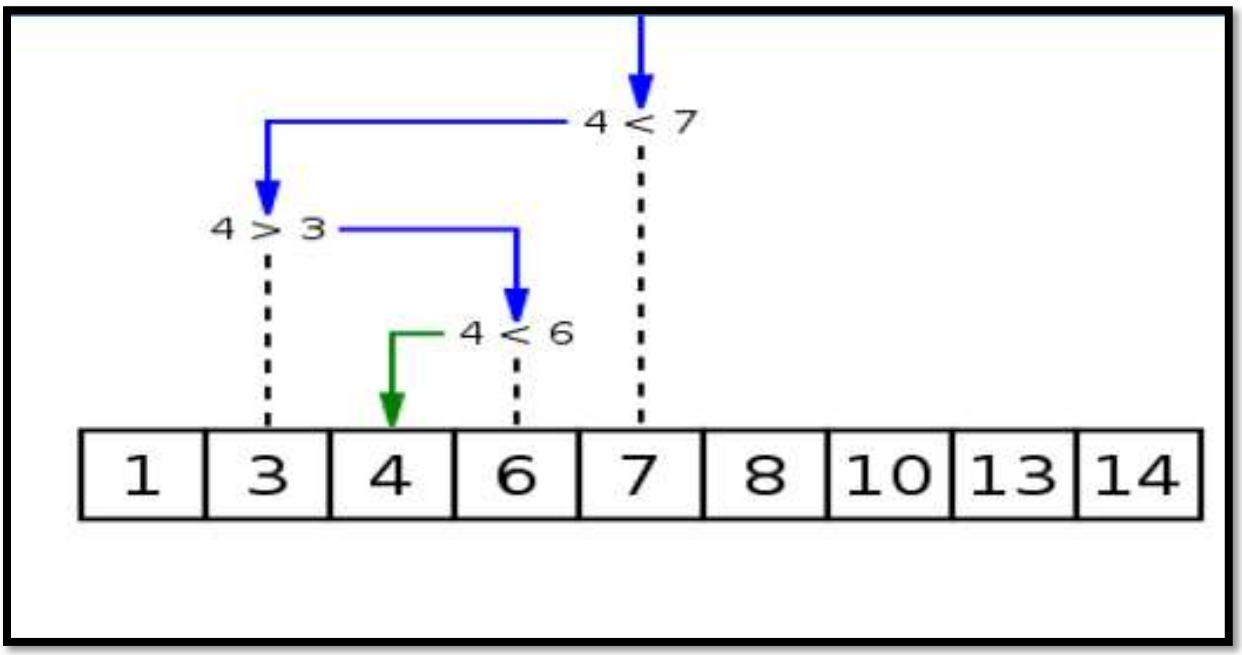
(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT

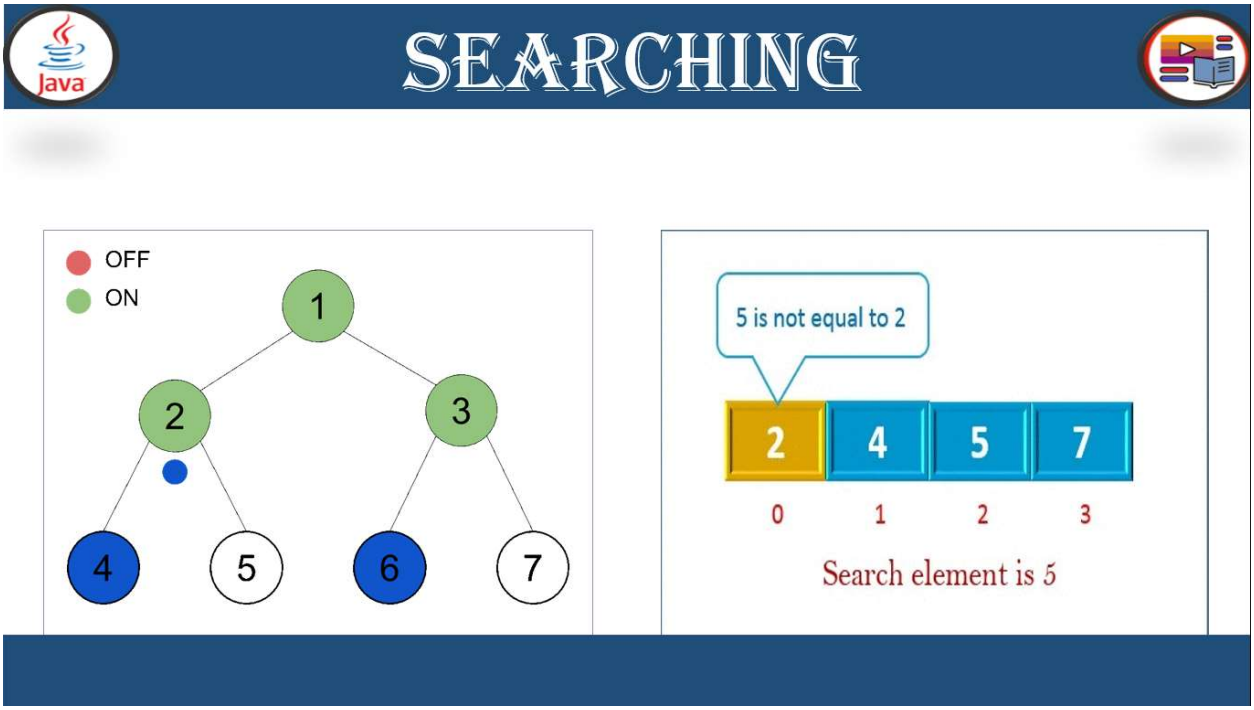


BINARY SEARCHING ALGORITHM

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to implement one of the most important data structures, Binary Searching algorithm using control statements and arrays.



REQUIREMENT

In computer science, binary search, also known as half-interval search, logarithmic search, or binary chop, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array.

Worst complexity: O(log n)

Average complexity: O(log n)

Best complexity: O(1)

Space complexity: O(1)

ALGORITHM

Binary search is the most popular Search algorithm.It is efficient and also one of the most commonly used techniques that is used to solve problems.

If all the names in the world are written down together in order and you want to search for the position of a specific name, binary search will accomplish this in a maximum of 35 iterations.

Binary search works only on a sorted set of elements. To use binary search on a collection, the collection must first be sorted.

IMPLEMENTATION

Get the user input in ascending sorted order

Search for a key

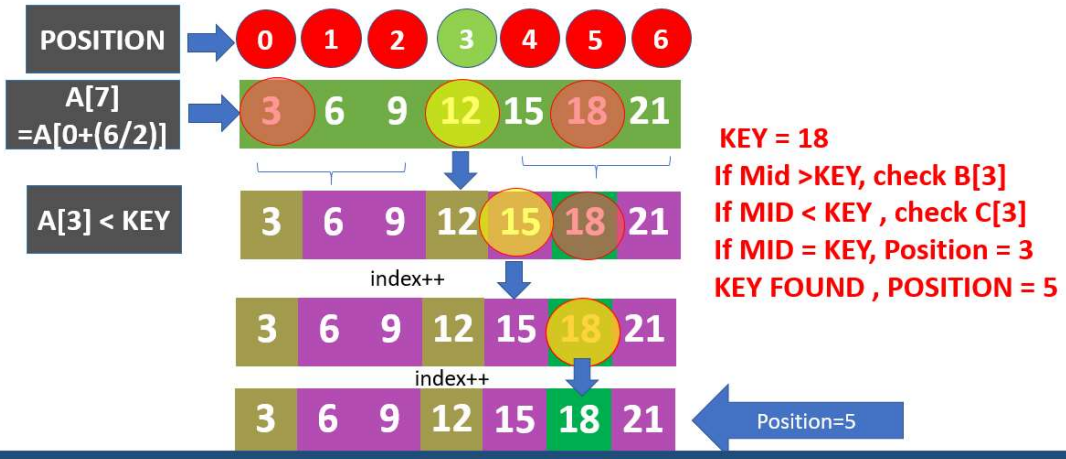
Get key position

Check for key if it is not found

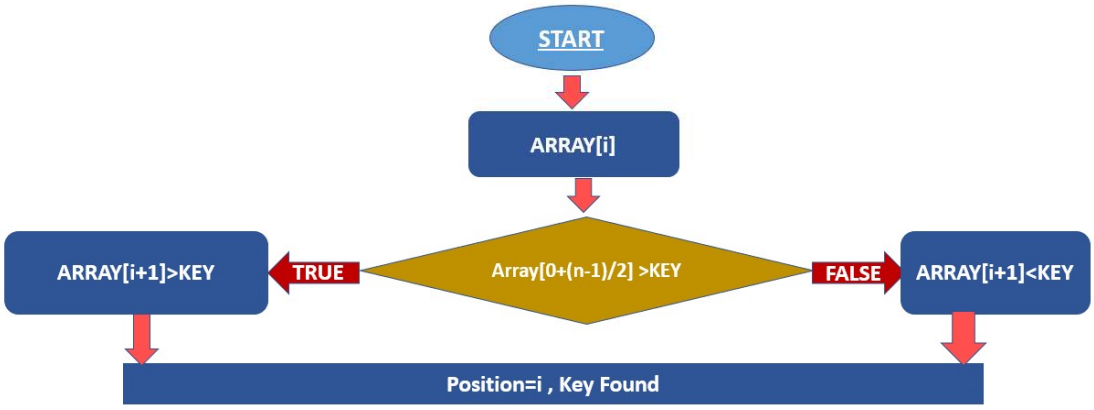
IMPLEMENTATION

When binary search is used to perform operations on a sorted set, the number of iterations can always be reduced on the basis of the value that is being searched

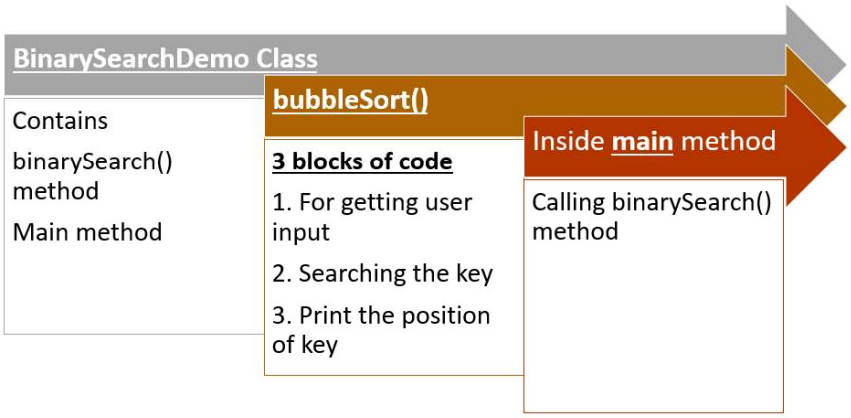
ALGORITHM



FLOW-CHART



DEMONSTRATION



SOURCE CODE

```
import java.util.Scanner;
public class BinarySearchDemoV1 {
       public void BinarySearch(){
        int c, first, last, middle, n, search, array[];
           @SuppressWarnings("resource")
              Scanner in = new Scanner(System.in);
           System.out.println("Enter number of elements");
           n = in.nextInt();
           array = new int[n];
           System.out.println("Enter " + n + " integers");
           for (c = 0; c < n; c++)
             array[c] = in.nextInt();
           System.out.println("Enter value to find");
           search = in.nextInt();
           first  = 0;
           last   = n - 1;
           middle = (first + last)/2;
           while( first <= last )
           {
             if ( array[middle] < search )
               first = middle + 1;
             else if ( array[middle] == search )
             {
               System.out.println(search + " found at location " + (middle + 1) +
".");
               break;
             }
             else
                last = middle - 1;
```

```
            middle = (first + last)/2;
         }
         if (first > last)
            System.out.println(search + " isn't present in the list.\n");
       }
      public static void main(String args[]){
      BinarySearchDemoV1 binary = new BinarySearchDemoV1();
      binary.BinarySearch();
      }
}
```

CONCLUSION

. In this session, we have seen how to implement Binary Search data structure algorithm using Control statements and recursion in Java.

PRACTICE PROJECTS

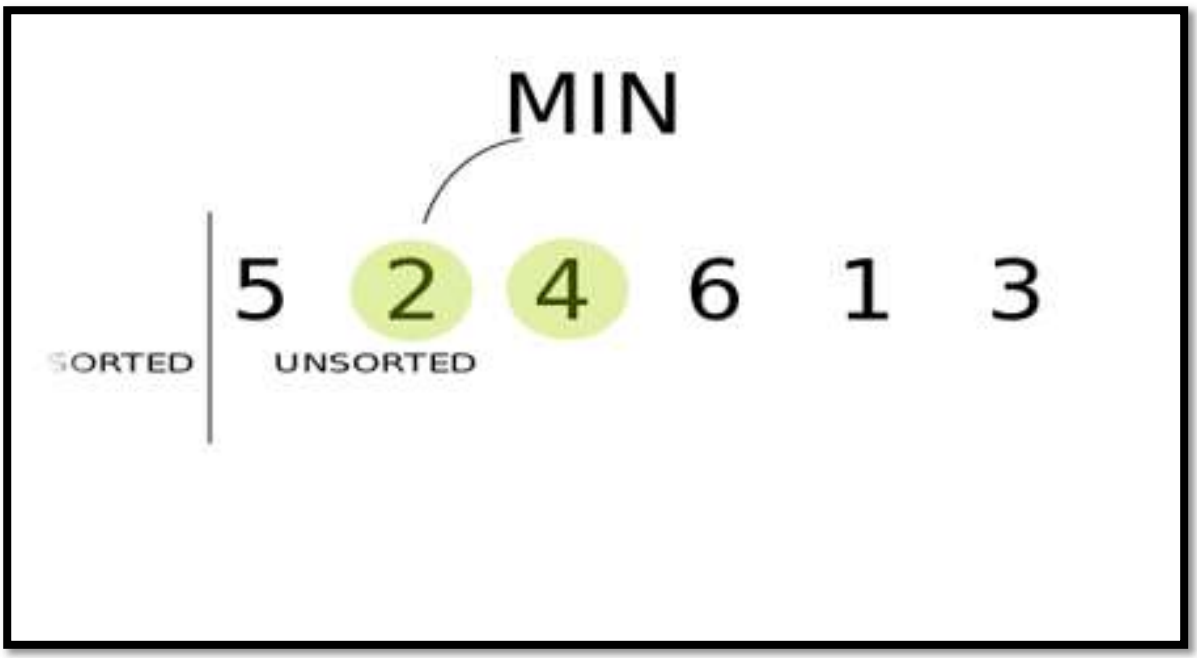
(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT

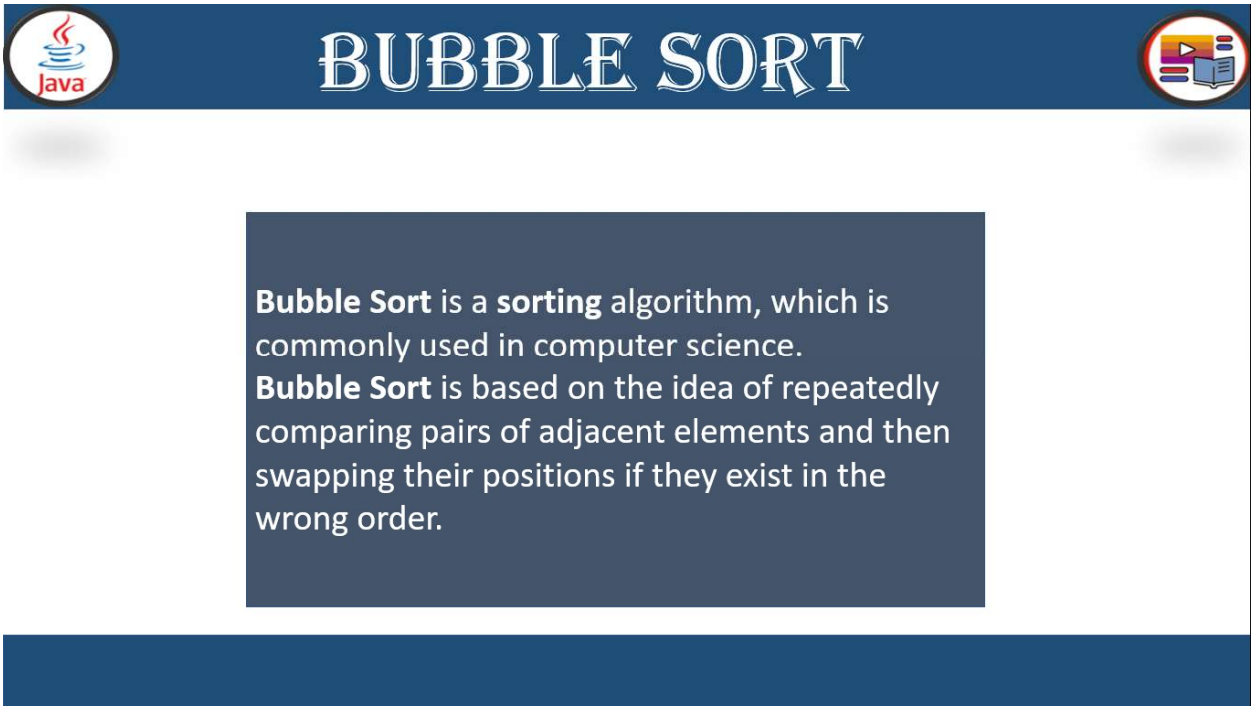


BUBBLE SORTING ALGORITHM

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to implement one of the most important data structures, Bubble Sorting algorithm using control statements and arrays.



REQUIREMENT

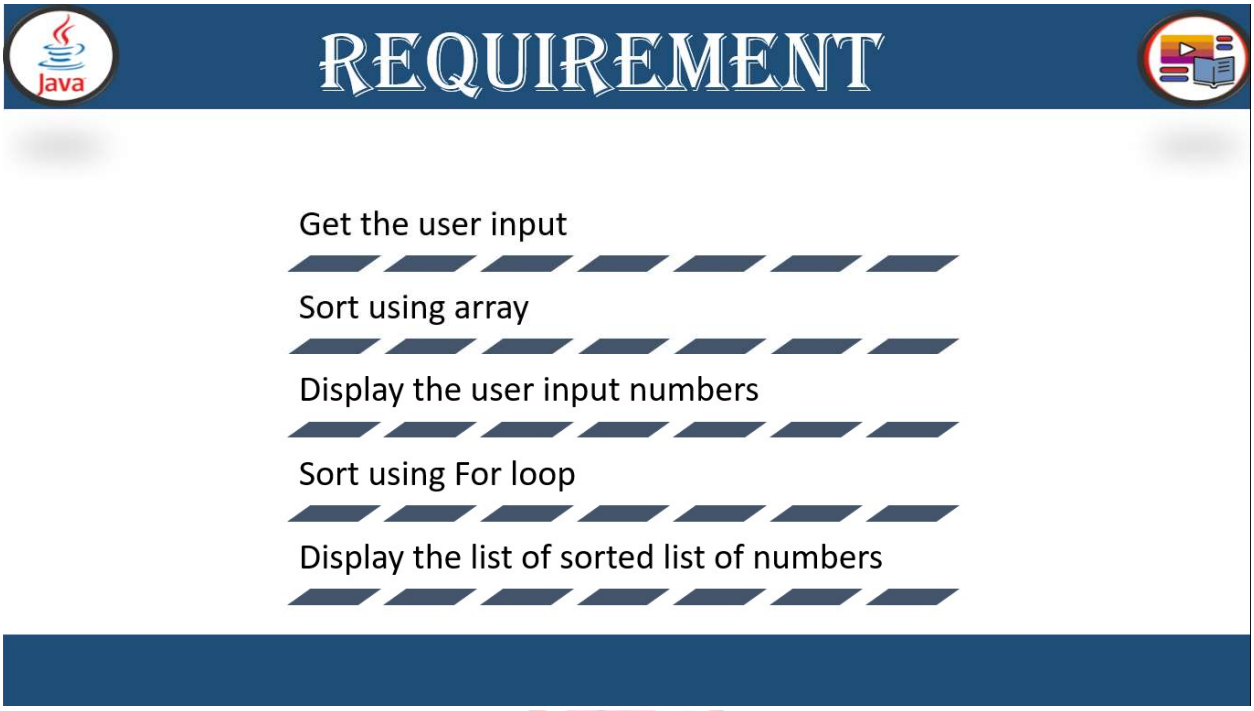
Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

Worst complexity: n^2

Average complexity: n^2

Best complexity: n

Space complexity: 1



ALGORITHM

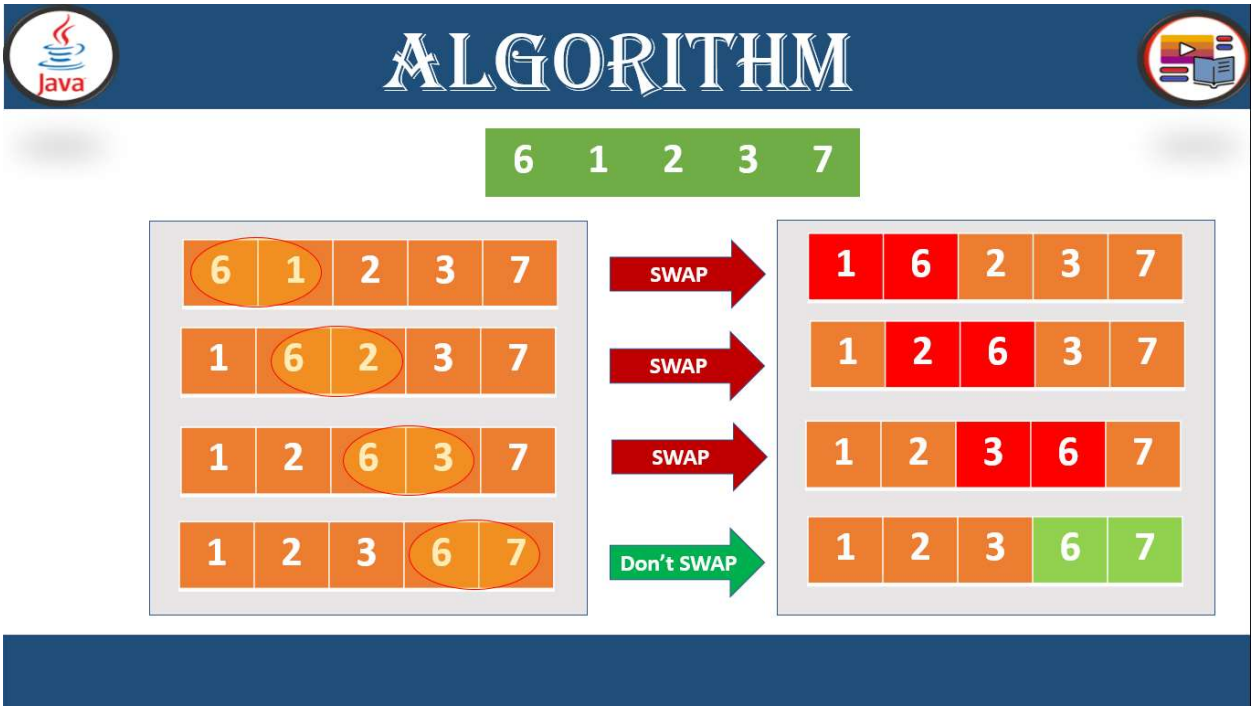
Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Worst and Average Case Time Complexity: O(n*n). Worst case occurs when array is reverse sorted.

Best Case Time Complexity: O(n). Best case occurs when array is already sorted.
Auxiliary Space: O(1)
Boundary Cases: Bubble sort takes minimum time (Order of n) when elements are already sorted.
Sorting In Place: Yes
Stable: Yes

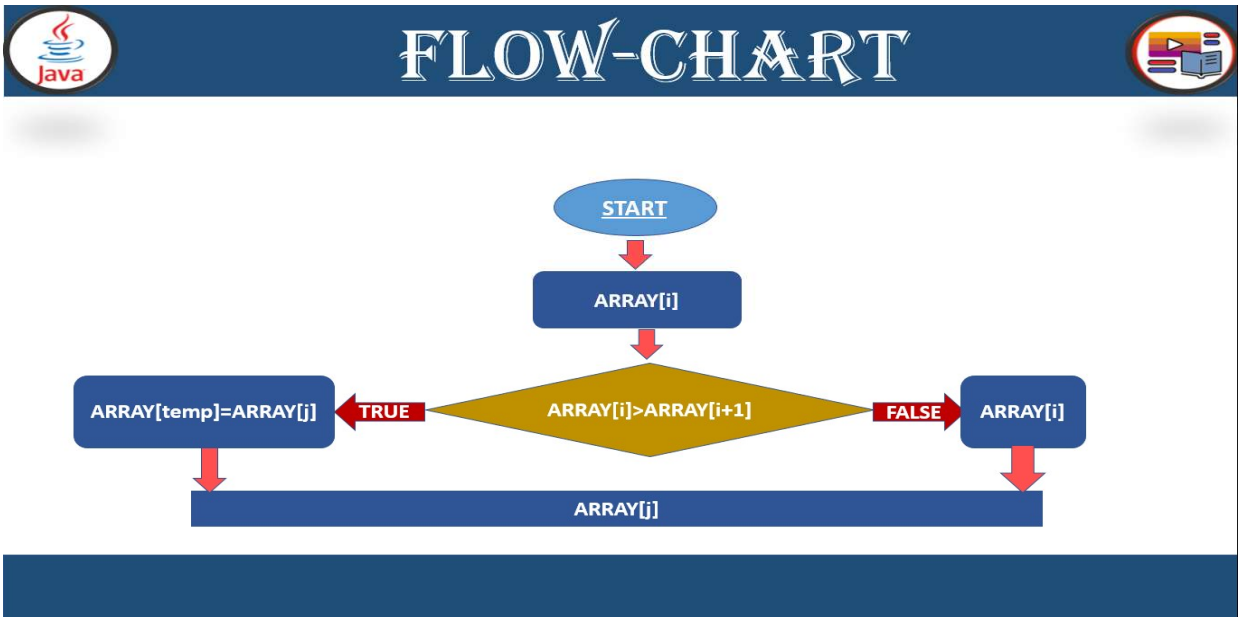
Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.

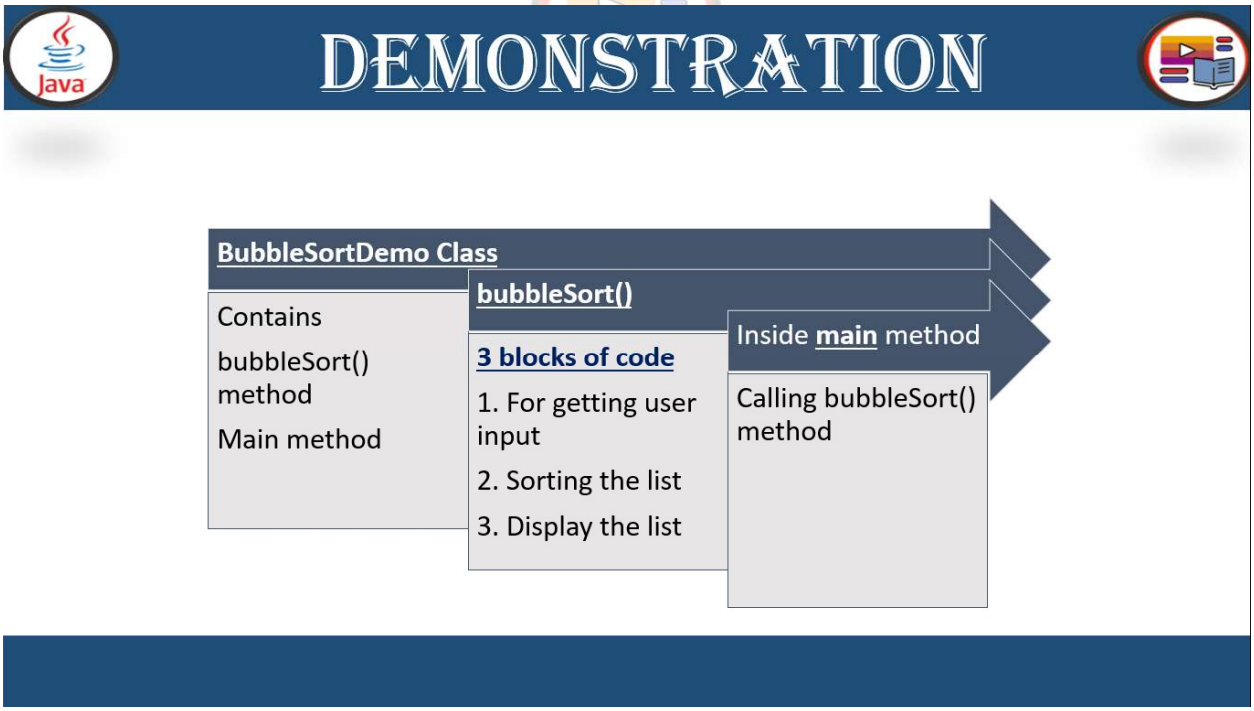
ALGORITHM



In computer graphics it is popular for its capability to detect a very small error (like swap of just two elements) in almost-sorted arrays and fix it with just linear complexity (2n). For example, it is used in a polygon filling algorithm, where bounding lines are sorted by their x coordinate at a specific scan line (a line parallel to x axis) and with incrementing y their order changes (two elements are swapped) only at intersections of two lines

FLOW-CHART


FLOW-CHART
START
ARRAY[i]
ARRAY[temp]=ARRAY[j]
TRUE
ARRAY[i]>ARRAY[i+1]
FALSE
ARRAY[i]
ARRAY[j]


DEMONSTRATION
BubbleSortDemo Class
Contains
bubbleSort() method
Main method
bubbleSort()
3 blocks of code
1. For getting user input
2. Sorting the list
3. Display the list
Inside main method
Calling bubbleSort() method

SOURCE CODE

```
import java.util.Scanner;
public class BinarySearchDemoV1 {
       public void BinarySearch(){
        int c, first, last, middle, n, search, array[];
           @SuppressWarnings("resource")
              Scanner in = new Scanner(System.in);
           System.out.println("Enter number of elements");
           n = in.nextInt();
           array = new int[n];
           System.out.println("Enter " + n + " integers");

           for (c = 0; c < n; c++)
             array[c] = in.nextInt();
           System.out.println("Enter value to find");
           search = in.nextInt();
           first  = 0;
           last   = n - 1;
           middle = (first + last)/2;
           while( first <= last )
           {
             if ( array[middle] < search )
               first = middle + 1;
             else if ( array[middle] == search )
             {
               System.out.println(search + " found at location " + (middle + 1) +
".");

               break;
             }
             else
                last = middle - 1;
             middle = (first + last)/2;
          }
          if (first > last)
             System.out.println(search + " isn't present in the list.\n");
        }
       public static void main(String args[]){
       BinarySearchDemoV1 binary = new BinarySearchDemoV1();
       binary.BinarySearch();
       }
}
```

CONCLUSION

. In this session, we have seen how to implement Bubble Sorting data structure algorithm using Control statements and recursion in Java.

PRACTICE PROJECTS

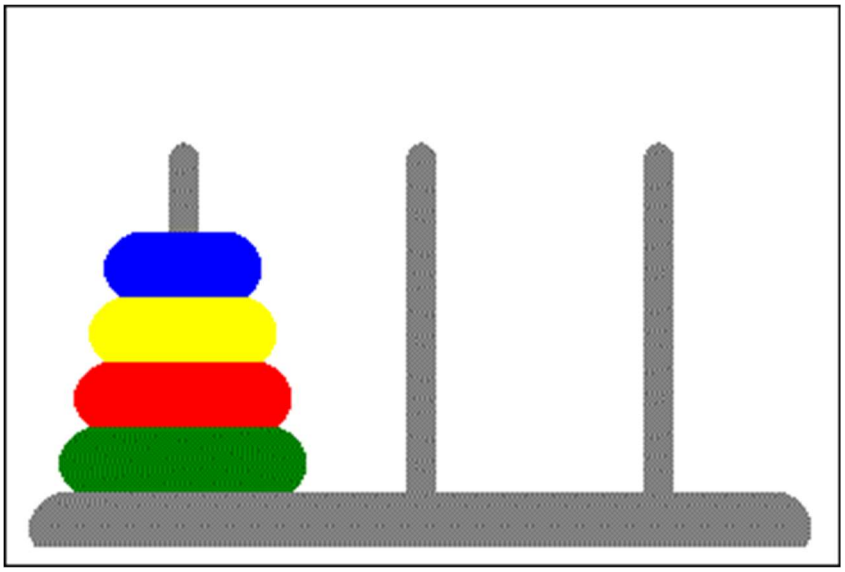
(Java Practice Projects)

Presented by

Anjali Singh

Let's make coding fun!

JAVA PRACTICE PROJECT

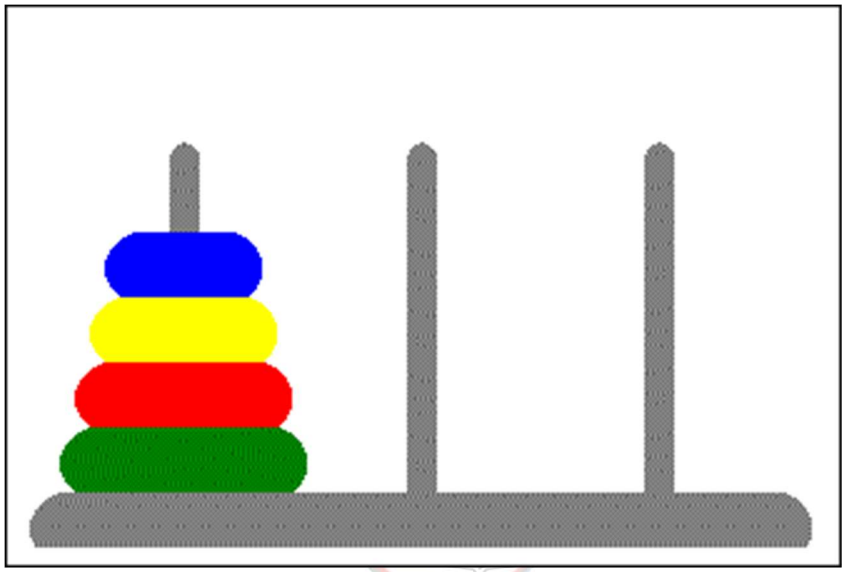


STACK IMPLEMENTATION

Let's begin.....

OBJECTIVE

Tech-Ranch presents how to implement one of the most important data structures, Stack using recursion, control statements and arrays.

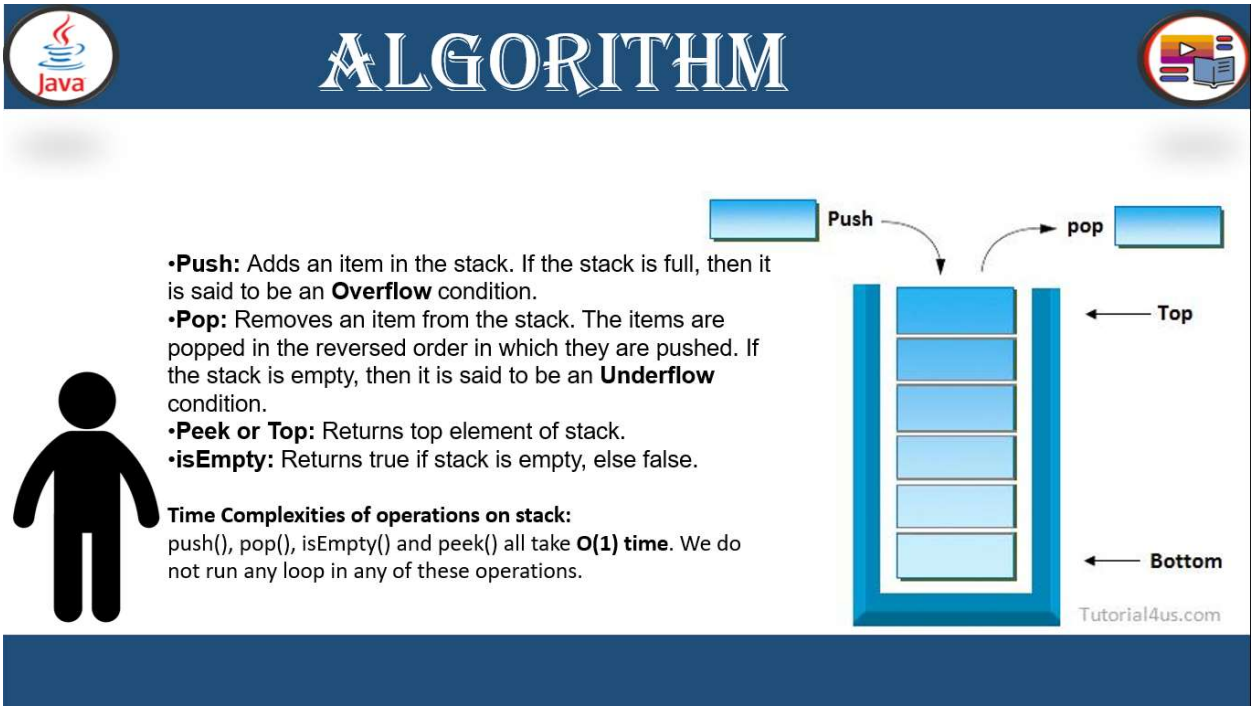


REQUIREMENT

A **stack** is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. A **stack** is a limited access **data structure**- elements can be added and removed from the **stack** only at the top. push adds an item to the top of the **stack**, pop removes the item from the top make logic for implementation. There are so many ways to implement such requirement.

ALGORITHM

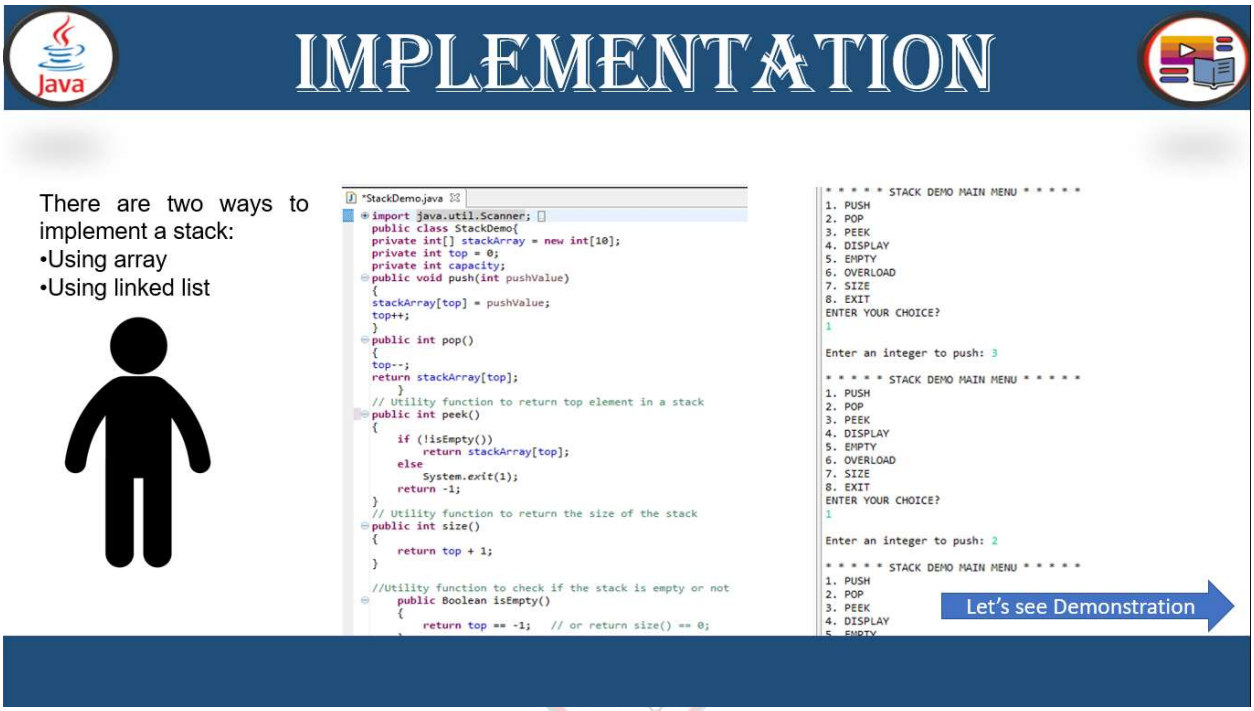
In computer science, a stack is an abstract data type that serves as a collection of elements, with two principal operations: push, which adds an element to the collection, and pop, which removes the most recently added element that was not yet removed.



IMPLEMENTATION

There are many real-life examples of a stack. Consider the simple example of plates stacked over one another in a canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO/FILO order.

There are so many ways to implement Stack. One of the simplest ways is using array and control statements.



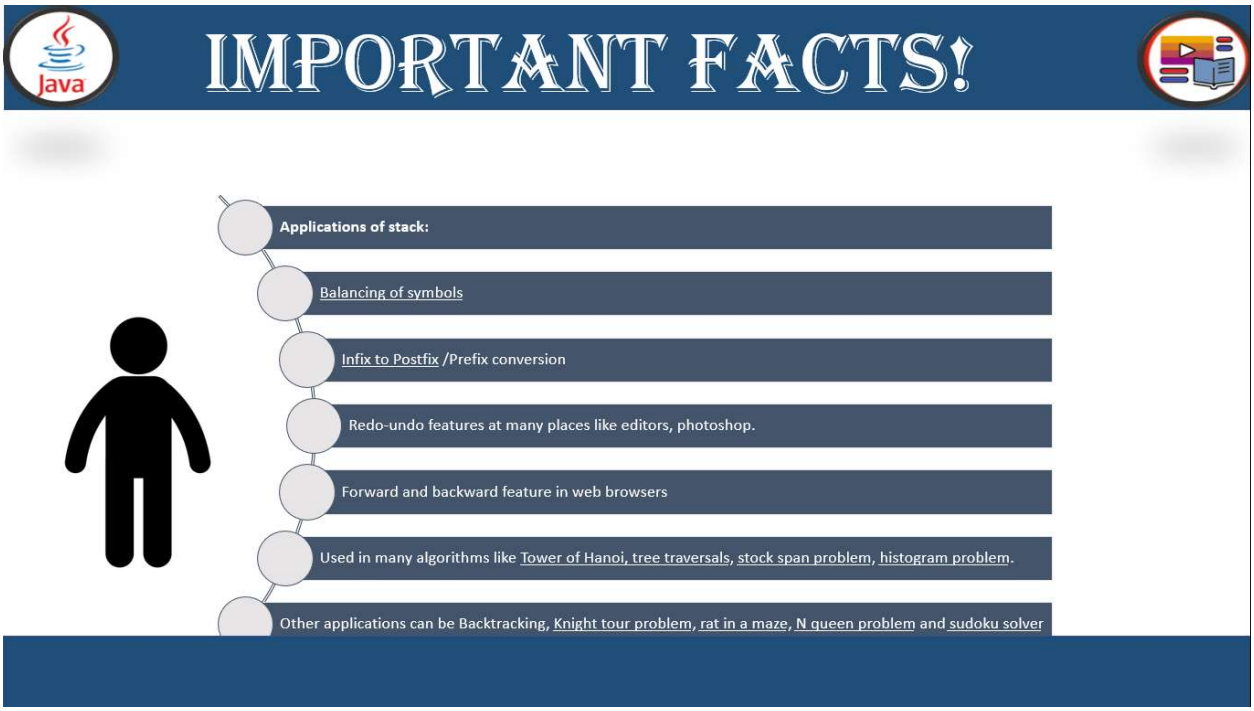
SOURCE CODE

```
import java.util.Scanner;
import java.util.Stack;
public class StackDemo{
private int[] stackArray = new int[10];
private int top = 0;
private int capacity;
public void push(int pushValue)
{
stackArray[top] = pushValue;
top++;
}
```

```
public int pop()
{
top--;
return stackArray[top];
   }
// Utility function to return top element in a stack
public int peek()
{
        if (!isEmpty())
                return stackArray[top];
        else
                System.exit(1);
        return -1;
}
// Utility function to return the size of the stack
public int size()
{
        return top + 1;
}
//Utility function to check if the stack is empty or not
        public Boolean isEmpty()
        {
                return top == -1;   // or return size() == 0;
        }
        // Utility function to check if the stack is full or not
        public Boolean isFull()
        {
                return top == capacity - 1;       // or return size() == capacity;
```

```
        }
@SuppressWarnings("unchecked")
public static void main(String args[]) {
int choice, pushValue;
boolean done = false;
@SuppressWarnings("rawtypes")
Stack stack1 = new Stack();
StackDemo s = new StackDemo();
@SuppressWarnings("resource")
Scanner keyboard = new Scanner(System.in);
while (!done)
{
        System.out.println("* * * * * STACK DEMO MAIN MENU * * * * *");
        System.out.println("1. PUSH");
        System.out.println("2. POP");
        System.out.println("3. PEEK");
        System.out.println("4. DISPLAY");
        System.out.println("5. EMPTY");
        System.out.println("6. OVERLOAD");
        System.out.println("7. SIZE");
        System.out.println("8. EXIT");
        System.out.println("ENTER YOUR CHOICE?");
choice = keyboard.nextInt();
System.out.println();
switch (choice)
{
case 1:
        if(t1 == true)
```

```
            System.out.println("Empty -->"+stack1);
            else
            System.out.println("Overload -->"+stack1);
            break;
case 7 : System.out.println("Size "+ stack1.size());
                    break;
case 8:
done = true;
break;
default:
System.out.println("The number you entered, "+ choice + "," + "is not 1, 2, or 3.
Try again!");
System.out.println();
break;
}
}
System.out.println("...quitting");
}
}
```



CONCLUSION

. In this session, we have seen how to implement Stack data structure algorithm using Control statements and recursion in Java.