DEMO QUESTIONS

1.What is backtracking in Java?

1.A type of sorting algorithm.

2. A technique for finding the shortest path in a graph.

3. A method for solving problems by incrementally building a solution and undoing steps if necessary.

4. A way to optimize recursive algorithms.

2. What does time complexity of an algorithm refer to?

1. The amount of time it takes to write the algorithm.

2. The amount of time the algorithm takes to execute as a function of the input size.

3. The maximum time the algorithm can run before crashing.

4. The number of lines of code in the algorithm.

3. Which data structure in Java is used to store a collection of elements with no duplicate values?

1. ArrayList

2.LinkedList

3.HashSet

4.TreeMap

4. What will be the output of this Java program?

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Stack<Double> myStack = new Stack<>();

myStack.push(2, 3.14);

myStack.push(3, 2.71);

myStack.set(1, 1.0);

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

}

}

1. 3.14
2. 2.71
3. 1
4. Compile Error

5. What will be the output of this Java program?

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

LinkedList<Double> myList = new LinkedList<>();

myList.add(3.14);

myList.add(2.71);

myList.set(1, 1.0);

System.out.println(myList.get(1));

}

}

1. 3.14
2. 2.71
3. 1
4. Compile error

6. What is the time complexity of the following Java code snippet?

int i = 1;

while (i < n) {

i \*= 2;

}

1. O(1)
2. O(log n)
3. O(n)
4. O(n^2)

7. What is the space complexity of the following Java code snippet?

int[] numbers = new int[n];

for (int i = 0; i < n; i++) {

numbers[i] = i;

}

1. O(1)
2. O(n)
3. O(n^2)
4. O(log n)

8. What will be the output of the following backtracking-based code?

public class BacktrackingExample {

public static void main(String[] args) {

int[] nums = {1, 2, 3};

generateSubsets(nums, 0, new ArrayList<>());

}

static void generateSubsets(int[] nums, int index, List<Integer> current) {

System.out.println(current);

for (int i = index; i < nums.length; i++) {

current.add(nums[i]);

generateSubsets(nums, i + 1, current);

current.remove(current.size() - 1);

}

}

}

1. [1, 2, 3]

[1, 2]

[1, 3]

[1]

[2, 3]

[2]

[3]

[]

2. [1]

[2]

[3]

[1, 2]

[1, 3]

[2, 3]

[1, 2, 3]

[]

3. [1, 2, 3]

[1, 2]

[1]

[2]

[2, 3]

[3]

[3, 2]

[]

4.None of the above

9. What will be the output of this Java program?

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Stack<String> myStack = new Stack<>();

myStack.push("Java");

myStack.push("C++");

myStack.push("Python");

boolean found = myStack.search("C++");

System.out.println(found);

}

}

1. TRUE
2. FALSE
3. C++
4. 2

10. What will be the output of this Java program?

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

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

myList.add(10);

myList.add(20);

myList.add(30);

myList.remove(1);

System.out.println(myList.size());

}

}

1. 1
2. 2
3. 3
4. Compile error

11. What will be the output of this Java program?

public class Main {

public static void main(String[] args) {

try {

throw new RuntimeException("Runtime Error");

} catch (IllegalArgumentException e) {

System.out.println("Exception IllegalArgument: " + e.getMessage());

} catch (RuntimeException e) {

System.out.println("Exception Runtime: " + e.getMessage());

}

}

}

1. Exception IllegalArgument: Error
2. Exception Runtime: Runtime Error
3. Error
4. Compile Error

12. What is the time complexity of the following Java code snippet?

int sum = 0;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

sum += i \* j;

}

}

1. O(1)
2. O(n)
3. O(n^2)
4. O(log n)

13. What will be the output of the following backtracking-based code?

public class BacktrackingExample {

public static void main(String[] args) {

System.out.println(countNQueens(4));

}

static int countNQueens(int n) {

int[] colPlacements = new int[n];

return solveNQueens(colPlacements, 0);

}

static int solveNQueens(int[] colPlacements, int row) {

if (row == colPlacements.length) {

return 1;

}

int count = 0;

for (int col = 0; col < colPlacements.length; col++) {

colPlacements[row] = col;

if (isValid(colPlacements, row)) {

count += solveNQueens(colPlacements, row + 1);

}

}

return count;

}

static boolean isValid(int[] colPlacements, int row) {

for (int i = 0; i < row; i++) {

int diff = Math.abs(colPlacements[i] - colPlacements[row]);

if (diff == 0 || diff == row - i) {

return false;

}

}

return true;

}

}

1. 1
2. 2
3. 3
4. 4

14. What will be the output of this Java program?

import java.util.Stack;

public class Main {

public static void main(String[] args) {

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

myStack.push('X');

myStack.push('Y');

myStack.push('Z');

char element = myStack.get(1);

System.out.println(element);

}

}

1. X
2. Y
3. Z
4. Compile error

15. What will be the output of this Java program?

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

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

myList.add("Apple");

myList.add("Banana");

myList.add("Cherry");

String fruit = myList.get(1);

System.out.println(fruit);

}

}

1. Apple
2. Banana
3. Cherry
4. Compile error

16. What will be the output of this Java program?

public class Main {

public static <T> void swapValues(T[] array, int i, int j) {

T temp = array[i];

array[i] = array[j];

array[j] = temp;

}

public static void main(String[] args) {

Integer[] numbers = {5, 10};

swapValues(numbers, 0, 1);

System.out.println(numbers[0] + " " + numbers[1]);

}

}

1. 5 10
2. 10 5
3. 10 5
4. Compile error

17. What is the output of the code?

abstract class Shape {

abstract void draw();

}

class Square extends Shape {

void draw() {

System.out.println("Drawing a square");

}

}

public class Main {

public static void main(String[] args) {

Shape shape = new Square();

shape.draw();

}

}

1. "Drawing a shape"
2. Compilation error
3. Runtime error
4. “Drawing a square”

18. What will be the output of the following backtracking-based code?

public class BacktrackingExample {

public static void main(String[] args) {

int[] nums = {1, 2, 3};

generatePermutations(nums, 0);

}

static void generatePermutations(int[] nums, int index) {

if (index == nums.length) {

for (int num : nums) {

System.out.print(num + " ");

}

System.out.println();

return;

}

for (int i = index; i < nums.length; i++) {

swap(nums, index, i);

generatePermutations(nums, index + 1);

swap(nums, index, i); // Backtrack

}

}

static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

1.

1 2 3

1 3 2

2 1 3

2 3 1

3 2 1

3 1 2

2.

1 2 3

1 3 2

2 1 3

3.

1 2 3

3 1 2

1. 3 1

4.The code will not compile due to an error.

19. What will be the output of this Java program?

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Stack<String> myStack = new Stack<>();

myStack.push("A");

myStack.push("B");

myStack.push("C");

int size = myStack.size();

System.out.println(size);

}

}

1. 0
2. 1
3. 2
4. 3

20. What will be the output of this Java program?

import java.util.LinkedList;

import java.util.Iterator;

public class Main {

public static void main(String[] args) {

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

myList.add(1);

myList.add(2);

myList.add(3);

Iterator<Integer> iterator = myList.iterator();

int sum = 0;

while (iterator.hasNext()) {

sum += iterator.next();

}

System.out.println(sum);

}

}

1. 6
2. 3
3. 12
4. Compile error

21. What will be the output of this Java program?

public class Main {

public static <T> T add(T a, T b) {

return a + b;

}

public static void main(String[] args) {

int result = add(5, 7);

System.out.println(result);

}

}

1. 5
2. 7
3. 12
4. Compiler error

22. What is the output of the code?

abstract class Animal {

abstract void makeSound();

}

class Dog extends Animal {

void makeSound() {

System.out.println("Woof!");

}

}

public class Main {

public static void main(String[] args) {

Animal animal = new Dog();

animal.makeSound();

}

}

1. “woof!”
2. Compile error
3. Runtime error
4. “Animal sound”

23. What will be the output of the following backtracking-based code?

public class BacktrackingExample {

public static void main(String[] args) {

generateSubsets("", "123");

}

static void generateSubsets(String prefix, String remaining) {

System.out.println(prefix);

for (int i = 0; i < remaining.length(); i++) {

generateSubsets(prefix + remaining.charAt(i), remaining.substring(i + 1));

}

}

}

1.

123

12

13

1

23

2

3

2.

1

12

123

13

2

23

3

3. 123

2

3

4.The code will not compile due to an error.

24. What will be the output of this Java program?

import java.util.Stack;

public class Main {

public static void main(String[] args) {

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

myStack.push(1);

myStack.push(2);

myStack.push(3);

myStack.pop();

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

}

}

1. 1
2. 2
3. 3
4. EmptyStackException

25. What will be the output of this Java program?

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

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

myList.add("A");

myList.add("B");

myList.add("C");

myList.remove("B");

System.out.println(myList.size());

}

}

1. 2
2. 3
3. 1
4. Compile error

26. What will be the output of this Java program?

public class Main {

public static void main(String[] args) {

try {

throw new Exception("Error");

} catch (Exception e) {

System.out.println("Exception caught: " + e.getMessage());

}

}

}

1. Exception caught: Error
2. Exception caught: null
3. Error
4. Compile error

27. What is the output of the code?

abstract class Shape {

abstract void draw();

}

class Circle extends Shape {

void draw() {

System.out.println("Drawing a circle");

}

}

public class Main {

public static void main(String[] args) {

Shape shape = new Circle();

shape.draw();

}

}

1. “drawing a circle”
2. Compilation error
3. Runtime error
4. “drawing a shape”

28. What will be the output of the following backtracking-based code?

public class BacktrackingExample {

public static void main(String[] args) {

generatePermutations("", "ABC");

}

static void generatePermutations(String prefix, String remaining) {

if (remaining.length() == 0) {

System.out.println(prefix);

return;

}

for (int i = 0; i < remaining.length(); i++) {

generatePermutations(prefix + remaining.charAt(i),

remaining.substring(0, i) + remaining.substring(i + 1));

}

}

}

1. ABC AB AC BAC BCA CBA CAB
2. ABC BAC CBA ACB
3. ABC ACB BAC BCA CAB CBA
4. The code will not compile due to an error.

29. Which type of binding occurs when using method overloading in Java?

1. static

2. dynamic

3. early

4. late

30. What happens when you attempt to pop an element from an empty stack in Java?

1. The program crashes with an error.
2. The top element of the stack is returned.
3. The stack remains unchanged.
4. An exception is thrown.

31. In a LinkedList, how is the insertion of an element at the beginning typically done?

1. Using the insertFirst() method.
2. Using the addFirst() method.
3. Using the push() method.
4. Linked lists do not support insertion at the beginning.

32. What is the primary advantage of using generics in Java?

1. Generics help improve the overall performance of a program.
2. Generics provide a way to define custom data types.
3. Generics allow you to avoid using objects of any type.
4. Generics help achieve type safety by ensuring type compatibility at compile time.

33. Can an abstract class in Java have non-abstract methods?

1. No, all methods in an abstract class must be abstract.
2. Yes, but at least one method must be abstract.
3. Yes, abstract classes can have both abstract and non-abstract methods.
4. Yes, but abstract methods are optional in an abstract class.

34.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What does encapsulation in Java refer to? | The process of converting objects into primitive data types. | The process of hiding internal implementation details and exposing only necessary features. | The process of converting primitive data types into objects. | The process of creating multiple instances of a class. | 2 |  |

35.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What is the main purpose of using virtual functions in Java? | To ensure that a function cannot be overridden by subclasses. | To provide a way to access private methods from subclasses. | To enable late binding and achieve runtime polymorphism. | To restrict the visibility of methods to the same package. | 3 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What is a virtual function in Java? | A function that can only be accessed by the class that defines it. | A function that is marked as "virtual" to allow it to be overridden by subclasses. | A function that is automatically called when an object is created. | A function that can only be called from within the constructor. | 2 |  |

36.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What is the purpose of exception handling in Java? | To create custom error messages for the user. | To avoid any kind of error or exception from occurring. | To gracefully handle runtime errors and exceptional situations. | To improve the performance of the program. | 3 |  |

37.

38.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What is a LinkedList in Java? | A data structure that stores elements in a fixed-size array. | A linear data structure that stores elements in nodes, where each node points to the next node. | A type of sorting algorithm. | A built-in Java class for handling file input and output. |  |  |

39.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| What is a stack in Java? | A linear data structure that uses a circular buffer to store elements. | A data structure that stores elements in a sorted order. | A data structure that follows the LIFO (Last-In-First-Out) principle. | A data structure that allows elements to be accessed by an index. |  |  |

40.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Which of the following Java classes can be used to implement a stack? | ArrayList | LinkedList | Stack | Queue |  | 2 |

41.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| What is the main idea behind backtracking? | Dividing the problem into smaller subproblems and solving each independently. | Using a stack to keep track of intermediate results. | Enumerating all possible solutions and selecting the best one. | Constructing a solution incrementally and abandoning it when it's determined to be invalid. | 4 |

42.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Which notation is commonly used to describe the upper bound on the time complexity of an algorithm? | O-notation (Big O notation). | T-notation (Time notation). | N-notation (Number notation). | C-notation (Complexity notation). | 1 |

43.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Which data structure in Java provides a last-in, first-out (LIFO) ordering of elements? | Queue | Stack | Set | Map |  | 2 |

44.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In Java, which keyword is used to establish the relationship between a subclass and a superclass? | parent | base | is-a | extends |  | 4 |

45.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In the context of Java, what is polymorphism? | The ability of a class to inherit multiple parent classes. | The process of creating multiple instances of a class. | The ability of a subclass to hide methods of its superclass. | The ability of different classes to be treated as instances of the same class through a common interface. | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In Java, can you use the virtual keyword to define a virtual function like in some other programming languages? | Yes, Java supports the virtual keyword for defining virtual functions. | No, Java does not have a virtual keyword for defining virtual functions. | Yes, but the virtual keyword is used in a different context in Java. | Yes, but it's only supported for abstract classes. | 2 |

46.

47.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In Java, which keyword is used to declare a method as capable of throwing exceptions? | throws | try | catch | throw |  | 1 |

48.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| What is the advantage of a LinkedList over an ArrayList in Java? | LinkedList has better performance for random access operations. | LinkedList uses less memory compared to ArrayList. | LinkedList is faster for adding and removing elements from the middle of the list. | LinkedList is not a part of the Java Collections Framework. | 3 |

49.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Which operations are typically associated with a stack in Java? | Push and Pop | Insert and Delete | Enqueue and Dequeue | Add and Remove | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Which operation can be performed efficiently on a LinkedList compared to an ArrayList in Java? | Removing an element from the middle of the list. | Retrieving an element by index. | Iterating through the list using enhanced for loop. | Resizing the list to accommodate more elements. | 1 |

50.

51.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| When is backtracking an appropriate solution approach? | When the problem can be solved using a simple loop. | When the problem can be divided into subproblems of the same type. | When the problem involves only arithmetic operations. | When the problem can be solved using dynamic programming. | 2 |

52.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| What does O(1) time complexity mean? | The algorithm takes one unit of time to execute. | The algorithm takes a constant amount of time regardless of the input size. | The algorithm takes one time unit for every element in the input. | The algorithm takes a logarithmic amount of time to execute. | 2 |

53.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In Java, which data structure is implemented as a red-black tree and maintains elements in sorted order? | ArrayList | HashMap | HashSet | TreeSet |  | 4 |

54.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| What is polymorphism in Java? | The ability of a class to inherit properties from multiple parent classes. | The ability of a class to create multiple instances. | The ability of a class to have multiple constructors. | The ability of different classes to be treated as instances of the same class through a common interface. | 4 |