1. Write a short on Java class and explain the various components of Java class

→ class is a blueprint or template for creating objects. It defines the properties (fields) and behaviors (methods) that the objects created from the class will have.

→ Components of a Java Class:
Class Name
The name given to the class, usually written in PascalCase.
Example: Student, Employee

Fields / Variables
These are data members that store the state of an object.
Example: int age; String name;

Constructors
Special methods used to initialize objects.
Called automatically when a new object is created.
Can be default or parameterized.

Methods
Define behaviors or functions an object can perform.
Example: void study(), int calculateSalary()

Blocks
Used for static or instance initialization.
Static blocks run once when the class is loaded.
Instance blocks run before every constructor call.

Access Modifiers
Control the visibility of class members (public, private, protected, default).

Example::

```
public class Main {
    int x; // Field (or instance variable)

    // Constructor with a parameter
    public Main(int x) {
        this.x = x;
    }

    // Main method (program entry point)
    public static void main(String[] args) {
```

```
        Main myObj = new Main(5); // Creating an object
        System.out.println("Value of x = " + myObj.x); // Accessing field
using object
    }
}
```

2. Explain the concept of declaring objects

→ In the preceding sample programs, a line similar to the following is used to declare an object of type Box:

```
Box mybox = new Box();
```

• This statement combines the two steps just described. It can be rewritten like this to show each step more clearly:

```
Box mybox; // declare reference to object
mybox = new Box(); // allocate a Box objec
```

→ Write the code of box

3. Explain assigning of object reference variables

In Java, assigning one object reference to another means both will point to the same object in memory — not a new copy.

```
class Student {
    String name;
}

public class Example {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Yashas";

        Student s2 = s1; // s2 refers to same object as s1
        s2.name = "Murthy";

        System.out.println(s1.name); // Output: Murthy
    }
}
```

Changes made using s2 also affect s1 because both refer to the same object.

4. What are ‘methods’ in java. Explain

Methods are blocks of code in Java that perform specific tasks. They define the behavior of a class and help in code reuse.

→ can use box code

→ Explain the below code

It defines a method:
```
double volume() {
    return width * height * depth;
}
```

It shows how to call a method:
b1.volume(), b2.volume(), cube.volume()

Types:
Instance Methods – Need an object to call.
Static Methods – Can be called using the class name.

5. Differentiate between constructors and methods in Java

Feature	Constructor	Method
Purpose	Used to initialize objects	Used to define behavior (actions/tasks)
Name	Same as the class name	Any valid name (not same as class)
Return Type	No return type (not even `void`)	Must have a return type (e.g., `int`, `void`, `String`, etc.)
Invocation	Called automatically when an object is created	Called manually using object name or class name
Types	Default, parameterized, copy constructor *(from your notes)*	Built-in, user-defined, static, abstract, etc. *(from your notes)*
Can be called using	Implicitly or explicitly (via `new` keyword) *(from your notes)*	Called by referencing the method name *(from your notes)*
Can be inherited?	No (constructors are not inherited)	

| Yes (methods can be inherited and overridden)
|
| **Compiler role** | Java compiler provides a default if none is written *(from your notes)* | Not provided automatically by compiler
|

6. Write a short note on: a.‘new’ keyword b.constructors

new Keyword in Java
The new keyword is used for dynamic memory allocation.
It creates an object in the heap memory and returns its reference.
It also automatically calls the constructor of the class.

Example:
ClassName obj = new ClassName();

Constructors in Java
A constructor is a logical entity used to initialize an object in a class.
It has the same name as the class and no return type.
It is called automatically when an object is created using new.

Types of constructors:
Default constructor
Parameterized constructor
Copy constructor (user-defined)

7. Show the working of constructors in Java
→ From Notes – 1
8. Explain the types of constructors in Java
→ From Notes – 1
9. Illustrate ‘this’ keyword in Java with an example
→ From Notes – 1

10. What is garbage collection. Show with an example
Garbage Collection is the process of reclaiming memory used by objects that are no longer in use or have gone out of scope.

It occurs automatically during program execution.
Java does not require manual deletion of objects like in C/C++.
Helps prevent memory leaks and improves performance.

```
class Example {
    // finalize method overridden
    protected void finalize() {
        System.out.println("Object is garbage collected");
```

```
    }

    public static void main(String[] args) {
        Example obj = new Example(); // Object created
        obj = null;                  // Object becomes unreachable

        // Request garbage collection
        System.gc();
        System.out.println("Garbage collection requested");
    }
}
```

11. Illustrate the working of finalise method ()

The finalize() Method:
It is a special method called before garbage collection to perform cleanup tasks, such as releasing non-Java resources (e.g., file handles, database connections) that the object may be holding, before the memory is reclaimed.

Code above

12. Write few inbuilt functions for file I/O operations.

```
File file = new File("example.txt");
file.exists();        // Checks if file exists
file.createNewFile(); // Creates a new file
file.delete();        // Deletes the file
file.getName();       // Returns file name
file.length();        // Returns size of file in bytes
```

13. Write few inbuilt functions on console I/O operations

Component	**Class / Method**	**Purpose / Description**
	`Scanner sc = new Scanner(System.in)`	Reads user input from keyboard
	`sc.nextInt()`	Reads integer input
	`sc.nextLine()`	Reads a full line of text
	`sc.nextDouble()`	Reads a decimal number
	`BufferedReader br = new BufferedReader(new InputStreamReader(System.in))`	Reads input line by line (faster than

```
Scanner) |
|               | `br.readLine()`
              | Reads one full line of input                  |
|               | `System.out.print()`
              | Prints text without moving to a new line      |
|               | `System.out.println()`
              | Prints text and moves to a new line           |
|               | `System.out.printf()`
              | Prints formatted text                         |
Scanner is easy to use and preferred for beginner-level input tasks.
BufferedReader is faster and good for reading large input.
System.out is used for all types of console output.

 14. Write a program to define a class and an object.
Class = Box
Object = myBox
Method = volume()

→ Box code

15. Write a program to instantiate an object and use static numbers

class Box {
    double width, height, depth;

    // Method to calculate volume
    double volume() {
        return width * height * depth;
    }
}

public class Test {
    public static void main(String[] args) {
        Box b = new Box(); // Object creation

        // Static numbers
        b.width = 10;
        b.height = 5;
        b.depth = 2;

        // Calling method
        System.out.println("Volume: " + b.volume());
    }
}
```

16. Write a program to demonstrate nested classes
17. Write a program to demonstrate array of objects
21. Write a program to demonstrate nested array and class of objects

```
public class University {

    // Inner (nested) class
    class Student {
        String name;

        Student(String name) {
            this.name = name;
        }

        void display() {
            System.out.println("Student name: " + name);
        }
    }

    public static void main(String[] args) {
        University uni = new University();

        // Array of inner class objects
        Student[] students = new Student[3];
        students[0] = uni.new Student("Alice");
        students[1] = uni.new Student("Bob");
        students[2] = uni.new Student("Charlie");

        // Display student names
        for (Student s : students) {
            s.display();
        }
    }
}
```

A nested class is a class written inside another class.
An array of objects means a group of objects stored together in one array.

18. Describe the Java class constructor with a sample program
Def
Program: Box code

19. Write a Java Program to demonstrate Default Constructor
→ From Notes - 1
20. Write a program to define a class, describe its constructor with

overloading, instantiate its object and use static members

```
class Box {
    double width, height, depth;
    static int objectCount = 0; // Static member

    // Default constructor
    Box() {
        width = height = depth = 0;
        objectCount++;
    }

    // Constructor with parameters (overloaded)
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
        objectCount++;
    }

    // Cube constructor (another overload)
    Box(double len) {
        width = height = depth = len;
        objectCount++;
    }

    double volume() {
        return width * height * depth;
    }
}

public class Test {
    public static void main(String[] args) {
        Box b1 = new Box();              // Default constructor
        Box b2 = new Box(10, 5, 2);      // 3-arg constructor
        Box b3 = new Box(4);             // Cube constructor

        System.out.println("b2 Volume: " + b2.volume());
        System.out.println("Total Box objects created: " + Box.objectCount);
// static usage
    }
}
```

22. Write a Java program to create a file in Java using File class

→ From Notes - 1

23. All programs on file I/O (create, open, close, read , write)
→ From Notes - 1