Tushar

Properties of Procedure-Oriented Programming (POP)?

- **Function-Based:** Programs are divided into functions or procedures that perform specific tasks.
- **Top-Down Approach:**starts from the top and proceeds through the details
- **Sequential Execution:** Functions are executed in a sequential manner
- **Lack of Reusability:** Code reuse is limited compared to object-oriented programming, leading to duplication

Properties of Object-Oriented Programming (OOP)?

- **Encapsulation:** Combines data and methods into a single unit class, restricting access to some components.
- **Inheritance:** Allows a new class to inherit properties from an existing class, promoting code reuse.
- **Polymorphism:** Enables methods to function differently based on the parameters, through method overloading and overriding.
- **Abstraction:** Displaying on limited things and hiding the implementation details.

Difference Between OOP and Procedural Programming?

- **Approach:**
  OOP: Uses objects and classes, focusing on data and methods.
  Procedural: Uses functions and procedures, focusing on a sequence of tasks.
- **Data Handling:**
  OOP: Encapsulates data within objects, promoting data hiding and protection.
  Procedural: Uses global data that can be accessed by any function, making it less secure.
- **Reusability:**
  OOP: Promotes reusability through inheritance and polymorphism.
  Procedural: Limited reusability, often leading to code duplication.
- **Modularity:**
  OOP: Highly modular, making it easier to manage and maintain large codebases.
  Procedural: Less modular, can become complex and harder to manage as the program grows.
- **Ease of Maintenance:**
  OOP: Easier to maintain and update due to its modular structure.
  Procedural: Maintenance can be more challenging as changes in one part may affect other parts.

What are the key features of OOP?
Encapsulation:
Combines data and methods that manipulate the data into a single unit called a class.
Restricts access to certain components, enhancing security and data integrity.
Abstraction:
Simplifies complex systems by modeling classes based on essential qualities.
Hides implementation details, exposing only necessary functionalities to the user.
Inheritance:
Allows a new class (subclass) to inherit properties and behavior from an existing class (superclass).
Promotes code reuse and establishes a natural hierarchy between classes.
Polymorphism:
Enables methods to perform different functions based on the object they are acting on.
Achieved through method overloading (same method name, different parameters) and method overriding (subclass redefines superclass method).

What are key features of Java?

- **Platform Independence:** Java code can run on any platform with a Java Virtual Machine (JVM).
- **Object-Oriented:** Emphasizes modularity and reusability through classes and objects.
- **Automatic Memory Management:** Garbage collection frees developers from manual memory management.
- **Strongly Typed:** Variables must be declared with a specific data type.
- **Exception Handling:** Robust exception handling mechanism for error detection and recovery.
- **Multi-threading Support:** Built-in support for concurrent programming, allowing multiple threads to run concurrently.
- **Rich Standard Library:** Comprehensive library offering diverse functionalities for various programming tasks.

What is a JVM?
The Java Virtual Machine (JVM) is a crucial component of the Java Runtime Environment (JRE). It interprets Java bytecode, facilitating platform independence by enabling Java programs to run on any device or operating system with a JVM implementation. JVM manages memory, performs garbage collection, and provides runtime environment for Java applications, executing bytecode instructions and translating them into native machine code.

What is JDK?
The Java Development Kit (JDK) is a comprehensive package of tools necessary for Java development. It includes the Java Compiler (javac) for compiling Java source code into bytecode, the Java Runtime Environment (JRE) for executing Java applications, and various development utilities like debugger and documentation generator. JDK is essential for creating, compiling, and running Java programs efficiently.

What is a object?
An object is an instance of a class in object-oriented programming, representing a real-world entity with its properties and behaviors.

What is a class?
A blueprint or template for creating objects, defining their structure (attributes) and behavior (methods).

What are constants in java?
In Java, constants are variables whose values cannot be changed once assigned. They are declared using the `final` keyword, ensuring immutability and enhancing code clarity by providing meaningful names for values that remain constant throughout the program's execution.

What is a variable?
Variables in Java are placeholders used to store data temporarily in computer memory. They are characterized by a data type, which determines the type of data that can be stored in them and the operations that can be performed on them.

Differnce between primitve and non-primitve data types?
Primitive data types are basic data types built into the Java language, storing simple values directly in memory. They include integers, floating-point numbers, characters, and booleans, offering efficient memory usage and faster access. **Non-primitive (reference) data types**, like strings, arrays, and objects, are more complex, storing references to memory locations where the actual data is stored. They allow for more sophisticated data structures and behaviors but may require more memory and processing power to manage.

What is wrapper class in java?
Wrapper classes in Java provide a way to use primitive data types (int, char, etc.) as objects. Examples include Integer for int and Character for char. (import java.lang.Int …)
Example:
```
int num = 5;
Integer numObj = Integer.valueOf(num);
```

What are operators in java?
Operators in Java are special symbols used to perform operations on variables and values. They are categorized into several types:
Arithmetic Operators: +, -, *, /, %
Relational Operators: ==, !=, >, <, >=, <=
Logical Operators: &&, ||, !
Bitwise Operators: &, |, ^, ~, <<, >>, >>>
Assignment Operators: =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=
Unary Operators: +, -, ++, --, !
Ternary Operator: ? :

How to basically take command line arguments in java program?
```
int intValue = Integer.parseInt(args[0]); //To parse in a integer from the command line
String name = args[0]; //To parse a string from the commandd line
```

What is a BufferedReader?
In Java, BufferedReader is a class in the java.io package that reads text from a character-input stream, buffering characters so as to provide efficient reading of characters, arrays, and lines. It improves efficiency by reading data from a stream in chunks rather than one character at a time.
Initiate a buffer reader object
```
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
```
Use the object to take input:
```
 String name = reader.readLine();
```

What is a scanner class?
The Scanner class in Java is a part of the java.util package and is used for parsing primitive types and strings from the input stream. It provides methods to read various types of data, such as integers, doubles, strings, etc., from different input sources like System.in, files, or strings.

Differnence between while loop and do while loop?
While Loop:

- **Condition Evaluation:** The condition is checked before the loop body is executed.
- **Initialization:** Variables used in the loop condition often need to be initialized before the loop begins
- **Usage:** While loops are preferred when the loop may not need to execute at all

Do-While Loop:

- **Condition Evaluation:** The loop body is executed at least once before the condition is checked.
- **Initialization:** Variables can be initialized inside the loop body and used immediately
- **Usage**: Do-while loops are suitable when you want to execute the loop body at least once

Difference between break and continue?
Break:

- **Termination of Loop:** The break statement terminates the loop in which it is used immediately when encountered, regardless of the loop's condition. It exits the loop completely.
- **Usage:** Typically used to exit a loop prematurely based on some condition that is encountered during the loop's execution.
- **Effect:** Once the break statement is executed, control transfers to the statement immediately following the loop.

Continue:

- **Skipping Iteration:** The continue statement skips the current iteration of a loop and proceeds with the next iteration of the loop.
- **Usage:** Used to skip certain iterations of a loop based on a condition, without exiting the loop completely.
- **Effect:** When continue is encountered, the remaining code in the loop body for the current iteration is skipped, and control jumps to the loop's next iteration.

Explain arrays in java?
Arrays in Java are fixed-size containers that hold elements of the same data type. They provide indexed access to elements, facilitating efficient retrieval, modification, and traversal. Array elements can be accessed using their index, starting from 0 to the array length minus one. Arrays can store primitive data types or object references.

Differnce between arrays in C and Java?
Dynamic Memory Allocation:

- C arrays are typically statically allocated, while Java arrays are dynamically allocated objects.

Bounds Checking:

- C arrays lack built-in bounds checking, whereas Java arrays have runtime bounds checking to prevent buffer overflow errors.

Object-oriented Nature:

- C arrays are low-level constructs without built-in methods, while Java arrays are objects with inherited methods.

Multi-dimensional Arrays:

- C implements multi-dimensional arrays as arrays of arrays, while Java natively supports multi-dimensional arrays for efficient access.

Syntax for declarring arrays in java?
1-D Array:

```
int[] arr = new int[5];
```

2-D Array:

```
int[][] matrix = new int[3][4];
```

What are jagged arrays?
In Java, a jagged array is a multi-dimensional array where each row can have a different length. Unlike regular multi-dimensional arrays, which have fixed column lengths for each row, jagged arrays offer flexibility in row lengths. They are useful for representing irregular data structures or matrices where the number of elements in each row varies.
Example of implementation of jagged array:

```
int arr[][] = new int[4][] //Created a array of constant rows and didn’t mention coloumns
arr[1] = new int[2] //for row 1 I can have 2 coloumns
arr[2] = new int[3] //for row 2 I can have 3 coloumns
```

Hence this is why jagged arrays are used

Explain ‘array.length()’ what does it do?
The ‘array.length()’ is a property in Java arrays that returns the number of elements in the array. It provides the size of the array.

Difference between string with and wihtout the new keyword?
String using the new keyword:
```
  String str = new String("YourString")l
```
Explicity creates a new string object in the heap memory regardless of wheterh a string with the same content alread exists in the string pool
String made without using the new keyword:
```
  String varname = "String";
```
If the string arlready exists in the string pool then it will be reused or otherwise a new object willl be created and added to the pool

What is difference between the string constant pool and heap?
String Constant Pool:
- Stores unique string literals to conserve memory.
- Facilitates string interning for efficient memory usage.
- Used for storing compile-time constants and string literals.

String Heap:
- General-purpose memory area for dynamically allocated objects.
- Objects, including strings created with new, are stored here.
- Objects in the heap are managed by the garbage collector and may persist longer than those in the String Constant Pool.

Different methods in String?
- charAt(int index): Returns the character at the specified index in the string.
- length(): Returns the length of the string.
- substring(int beginIndex): Returns a new string that is a substring of the original string starting from the specified index.
- substring(int beginIndex, int endIndex): Returns a new string that is a substring of the original string starting from the beginIndex and ending at endIndex - 1.
- indexOf(String str): Returns the index of the first occurrence of the specified substring within the string, or -1 if not found.
- equals(Object obj): Compares the string to the specified object for equality.
- toUpperCase(): Converts all characters in the string to uppercase.
- toLowerCase(): Converts all characters in the string to lowercase.

Different methods in StringBuffer?
- append(): Appends the specified string, character, or object to the end of the StringBuilder.
- charAt(): Returns the character at the specified index in the StringBuilder.
- length(): Returns the length (number of characters) of the StringBuilder.
- substring(): Returns a new string that is a substring of the StringBuilder, starting from the specified start index to the end of the StringBuilder or up to the specified end index.
- reverse(): Reverses the order of characters in the StringBuilder.
- toString(): Converts the StringBuilder to a String.

Difference between String , StringBuffer and StringBuilder>
String:
- Immutable: Once created, the content of a String cannot be changed.
- Thread-safe: Safe for use in multi-threaded environments due to immutability.
- String concatenation creates new String objects, which can be inefficient for large-scale concatenation operations.

StringBuffer:
- Mutable: Content can be modified after creation.
- Thread-safe: Provides synchronized methods for safe use in multi-threaded environments.
- StringBuffer is slower than StringBuilder due to synchronization.
-

StringBuilder:
- Mutable: Content can be modified after creation.
- Not thread-safe: Not synchronized, making it more efficient in single-threaded environments.
- StringBuilder is faster than StringBuffer due to lack of synchronization

What is collection in java?
Collections in Java are tools for storing and managing groups of objects, offering flexibility and efficiency in programming.

What are Benefits of collection?
Dynamic array with vector: Vector provides automatic resizing of arrays for dynamic storage management.
Thread safe operation with vector: Vector ensures synchronized methods for safe concurrent access in multi-threaded environments.
Indexed access in arrays: Arrays enable efficient element retrieval and modification based on position.
Dynamic sized: Collections like ArrayList and LinkedList dynamically adjust their size to accommodate elements.

What is a arraylist?
An ArrayList in Java is a resizable array implementation, providing dynamic storage for elements and supporting various operations like adding, removing, and accessing elements at specific positions. It offers flexibility and efficiency in managing collections of objects.

What is vector?
A Vector in Java is similar to an ArrayList, providing dynamic resizing and indexed access to elements, but it is synchronized, ensuring thread-safe operations, making it suitable for concurrent environments where multiple threads access the collection simultaneously.

What is the difference between vector and array list?
Thread Safety:
- ArrayList: Not inherently thread-safe, requires external synchronization for concurrent access.
- Vector: Inherently synchronized, ensuring thread-safe operations without external synchronization.

Performance:
- ArrayList: Generally offers better performance in single-threaded scenarios due to lack of synchronization overhead.
- Vector: Synchronization overhead may impact performance, particularly in single-threaded environments.

Usage:
- ArrayList: Preferred in single-threaded scenarios and when explicit synchronization is not required.
- Vector: Suitable for multi-threaded environments or when thread safety is essential, but may incur a performance overhead in single-threaded scenarios.

Methods of collections:
ArrayList Methods:
- add(E element): Appends the specified element to the end of the list.
- get(int index): Returns the element at the specified index in the list.
- remove(int index): Removes the element at the specified index from the list.
- size(): Returns the number of elements in the list.
- clear(): Removes all elements from the list.

Vector Methods:
- addElement(E obj): Adds the specified component to the end of this vector, increasing its size by one.
- elementAt(int index): Returns the component at the specified index.
- removeElementAt(int index): Deletes the component at the specified index.
- size(): Returns the number of components in this vector.
- clear(): Removes all elements from the vector.

Initialize and use Vector:
```
Vector<String> studentnames = new Vector<>();
//Custom for loop
 for(String names : args){
 studentnames.add(names);
 }
```

Initliaze and use ArrayList:
```
ArrayList<Integer> list = new ArrayList<>();
list.add(10);
```

What are Access specifiers in java?
Access specifiers in Java control the visibility and accessibility of classes, methods, and variables, ensuring encapsulation and managing the level of access to members within and outside of classes.
Java access specifiers include:
- public: Accessible from any other class.
- protected: Accessible within the same package or by subclasses.
- default (no specifier): Accessible within the same package.
- private: Accessible only within the same class.

What is static and non-static memebers?
Static members are associated with the class itself and can be accessed without creating an instance, while non-static members are associated with individual instances and require object creation for access.

What is method overloading/static polymorphsm?
Method overloading in Java refers to the ability to define multiple methods in a class with the same name but with different parameters. This allows methods to perform similar tasks but with variations in the types or number of parameters they accept. Overloaded methods are distinguished by their parameter lists, enabling the compiler to determine which method to invoke based on the arguments passed during method invocation.

Different types of variables in java?
Local variables are declared within a method or block and have method-level scope.
Instance variables are declared within a class but outside any method and have object-level scope.
Static variables (class variables) are declared with the static keyword within a class but outside any method, and they have class-level scope.

What is “this” keyword in java?
The "this" keyword in Java refers to the current object instance, commonly used for disambiguating variables, accessing instance methods, or passing the current object as a parameter.

What are types of constructors in java?
In Java, constructors can be classified into two types:

- **Default Constructor:** This is provided by Java if no constructor is explicitly defined in a class. It initializes instance variables to their default values.
- **Parameterized Constructor:** This is a constructor with parameters. It allows for the initialization of instance variables with values passed as arguments when an object is created.
- **Copy Constructor:** A copy constructor is a constructor that creates a new object as a copy of an existing object. It takes an object of the same class as a parameter and initializes the new object's state by copying the state of the existing object.

What are properties of constructors in Java?

- **Same Name as Class:** Constructors have the same name as the class they belong to.
- **No Return Type:** Constructors do not have a return type, not even void.
- **Automatic Invocation:** Constructors are automatically invoked when an object of the class is created using the "new" keyword.
- **Initialization:** Constructors are primarily used for initializing the state of objects, setting initial values to instance variables or performing any necessary setup tasks.

What is constructor overloading in Java?
Constructor overloading in Java refers to the ability to define multiple constructors within a class, each with a different parameter list. This allows objects of the class to be initialized in different ways, depending on the arguments passed during object creation.

Difference Between Method and Constructor:
A method is a block of code within a class that performs a specific task, while a constructor is a special method used for initializing
objects.
Methods have a return type (void or a specific type), whereas constructors do not have a return type.
Methods are called explicitly by name, while constructors are called implicitly when an object is created.
You can define multiple methods in a class, but there can be only one constructor with the same name in the class.

What is inheritance in java?
Inheritance in Java is a fundamental object-oriented programming concept where one class (the subclass or derived class) inherits the fields and methods of another class (the super-class or base class). This allows the subclass to reuse, extend, and modify the behavior of the super-class, promoting code reuse and establishing a hierarchical relationship between classes

What is the role of constructor in inheritance?
Constructors in the super class are invoked when the object of the subclass is created. If no constructors is explicitly defined in the super class, the default constructor is called.
Constructors are not interdicted but are called implicit using the super keyword
Constructors facilitate the initialization of super class field before subclass fields, ensuring proper object initialization.
Super keyword with respect to variable , constructors and methods

What are types of inheritance in java?
In Java, there are four types of inheritance:

- **Single Inheritance:** A class inherits from one superclass.
- **Multilevel Inheritance:** A class inherits from a superclass, and another class inherits from that subclass, forming a chain.
- **Hierarchical Inheritance:** Multiple classes inherit from a single superclass.
- **Hybrid Inheritance (not directly supported):** A combination of two or more types of inheritance, usually achieved through interfaces due to the lack of multiple inheritance of classes in Java.''

What is method overriding?
Method overriding in Java is a feature that allows a subclass to provide a specific implementation of a method that is already defined in its superclass. The method in the subclass must have the same name, return type, and parameters as the method in the superclass. This allows the subclass to offer its own behavior while preserving the interface provided by the superclass.

What is the use of super keyword in java?
The super keyword in Java is used to access methods and constructors of the immediate superclass. It allows a subclass to invoke the superclass's constructor and methods, enabling code reuse and initialization of inherited properties.

What is the use of final keyword in java?
The final keyword in Java is used to restrict modification. It can be applied to variables (preventing reassignment), methods (preventing overriding), and classes (preventing inheritance).

What is meant by static binding and dynamic binding?
Static Binding:

- Occurs at compile-time.
- The method to be called is determined based on the type of the reference variable.
- Typically associated with method overloading.

Dynamic Binding:

- Occurs at runtime.
- The method to be called is determined based on the actual object, not the reference type.
- Typically associated with method overriding.

What is abstract class?
An abstract class in Java is a class that cannot be instantiated directly and may contain abstract methods, which are methods without a body. It is used as a blueprint for other classes to inherit from

```
abstract class AbstractClass {
    abstract void abstractMethod();
} //You can extend this to the required class using the extend keyword
```

What are properties of abstract class?

- They cannot be instantiated directly you can only create instance of their concrete sub class that extended it
- They contain both abstract and concrete method
- Abstract method defined in abstract classes must be implemented by their concrete sub classes
- Abstract classes can have constructors, fields , and regular methods like any other class

What is interface class?
An interface in Java is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types. It cannot contain method implementations or instance variables. Interfaces provide a way to achieve abstraction and multiple inheritance in Java.

```
interface InterfaceName {
           void draw();
 }
```

You use implement this using the implements keyword

What are properties of interface?

- Interface cannot be instantiated they can only provide method declaration and constants
- All methods is an interface are implicitly public and abstract meaning they must be implemented by the class the implements that interfaces
- Interface can contains constants which are implicitly public, static and final
- A class can implement multiple interfaces allowing for polymorphism and flexibility in object behavior

What is the difference between Abstract class and interface class?
Method Implementation:
Abstract classes can contain both abstract and concrete methods, while interfaces can only declare method signatures.
Inheritance:
A Java class can extend only one abstract class but can implement multiple interfaces.
Instance Variables:
Abstract classes can have instance variables, constructors, and method implementations, whereas interfaces cannot contain instance variables or constructors.
Usage:
Abstract classes are used to represent abstract concepts and provide a partial implementation, while interfaces define contracts and are used to achieve multiple inheritance and polymorphism.

What is garbage collection in java?
Garbage collection in Java is the process by which the Java Virtual Machine (JVM) automatically manages memory by reclaiming memory occupied by objects that are no longer in use or reachable by any part of the program. It helps in preventing memory leaks and ensures efficient memory utilization by deallocating memory from objects that are no longer needed
You can call the garbage collector using the System.gc();

What is the finalize keyword?
The finalize() method in Java is a special method that is called by the garbage collector just before an object is garbage collected, giving the object an opportunity to perform any necessary cleanup or resource releasing tasks

What are packages in Java?
In Java, packages are used to organize classes into namespaces, providing a way to group related classes and interfaces together. They help in avoiding naming conflicts and provide modularity, scalability, and code organization benefits

What are advantage of package?

- **Namespace Management:** Packages prevent naming conflicts by organizing classes into distinct namespaces.
- **Modularity:** They group related classes, promoting better code organization and modularity.
- **Access Control:** Packages control access levels, enhancing security and encapsulation.
- **Reusability**: Organized code in packages is easier to reuse across projects.
- **Improved Maintenance:** Packages facilitate easier code maintenance and debugging by grouping related functionality.

What is the difference between error vs exception?
Nature:
- Error: Represents serious system issues.
- Exception: Represents application-level issues.

Hierarchy:
- Error: Subclass of Throwable, not Exception.
- Exception: Subclass of Throwable and Exception.

Recoverability:
- Error: Generally non-recoverable.
- Exception: Often recoverable.

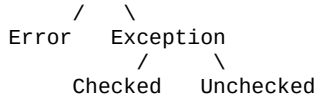
Handling:
- Error: Typically not handled in code.
- Exception: Commonly handled using try-catch blocks.

Important keywords in exception handling?

- **try:**Used to define a block of code that might throw an exception, allowing it to be caught and handled.
- **catch:**Used to handle the exception thrown by the try block, specifying what to do if a particular type of exception occurs.
- **finally:**Used to define a block of code that will always execute after the try and catch blocks, regardless of whether an exception was thrown or caught.
- **Throw:**Used to explicitly throw an exception, either a newly instantiated one or a caught one.
- **throws:**Used in a method signature to declare that the method can throw certain exceptions, informing the caller to handle or propagate them.

Difference between checked and unchecked exception?
Checked Exception:
These are exceptions that are checked at compile-time. They must be either caught or declared in the method signature using throws. Examples include IOException, SQLException, and FileNotFoundException.
Unchecked Exception:
These are exceptions that are not checked at compile-time but at runtime. They do not need to be declared or caught. Examples include NullPointerException, ArrayIndexOutOfBoundsException, and ArithmeticException.

What does throwable class mean?
The `Throwable` class in Java is the superclass of all errors and exceptions. It is the root class for the Java exception hierarchy and has two primary subclasses: `Error` and `Exception`.



What is multithreading in java?
Multithreading in Java is a feature that allows the concurrent execution of two or more threads within a single program, enabling multiple tasks to run simultaneously, thus improving the application's performance and responsiveness. It is achieved by using the Thread class or implementing the Runnable interface.

What is the life cycle of thread?

- **New:** A thread is in this state when it is created but not yet started.
- **Runnable:** The thread is ready to run and waiting for CPU time.
- **Running:** The thread is executing.
- **Blocked/Waiting:** The thread is inactive, waiting for a resource to become available or for another thread to complete.
- **Terminated:** The thread has finished executing.

What are some methods of Thread class?

- **start():** Starts the thread by calling its run() method.
- **run():** Contains the code to be executed by the thread.
- **join():** Waits for the thread to finish.
- **setPriority(int newPriority):** Sets the thread's priority.
- **getPriority():** Returns the thread's priority.

What is a runnable interface?
The Runnable interface in Java defines a single run() method, representing a task to be executed by a thread. It enables classes to implement thread behavior without extending the Thread class.

What is a Threads class?
The Thread class in Java represents a thread of execution in a program. It provides methods to create, start, and manage threads, allowing concurrent execution of code. By extending this class or implementing the Runnable interface, you can define the behavior of a thread and control its lifecycle

What is thread synchronization?
Thread synchronization in Java prevents concurrent threads from executing critical sections simultaneously to avoid data inconsistency and thread interference. It is achieved using the synchronized keyword, applied to methods or code blocks. This ensures that only one thread can access the synchronized code at a time.

What is a synchronized method?
Synchronized methods in Java are methods marked with the synchronized keyword, ensuring that only one thread can execute them at a time, preventing data inconsistency in shared resources. Similarly, synchronized blocks allow for more fine-grained control, synchronizing only specific sections of code. This synchronization mechanism helps maintain thread safety and prevents race conditions in multi-threaded environments.

What are three states of a thread in Java?
The three states of a thread in Java are:
Runnable: The thread is eligible to run but may or may not be currently executing. It's waiting for CPU time.
Blocked (or Waiting): The thread is temporarily inactive, typically waiting for a resource or a monitor lock to become available.
Terminated: The thread has finished executing its task and has permanently exited. Once a thread is terminated, it cannot be restarted.

What are two type of synchronization in java?
Static Synchronization: Involves synchronizing access to a method or block using the static synchronized keyword, which locks the entire class, preventing concurrent execution of static synchronized methods by multiple threads.
Dynamic Synchronization: Involves synchronizing access to a method or block using the synchronized keyword without the static modifier, allowing synchronized methods or blocks to be accessed concurrently by different instances of a class, with each instance having its own lock.

What are the different swing compoenets?
Swing is a GUI (Graphical User Interface) toolkit for Java, providing a rich set of components to create desktop applications. Some of the commonly used components in Swing include:
1. **JFrame:** A top-level container that represents the main window of the application.
2. **Jpanel:** A container used to organize other components within a window.
3. **JButton:** A button component used to trigger actions when clicked.
4. **JLabel:** A non-editable text component used to display text or images.
5. **JTextField:** A text input component where users can enter single-line text.
7. **JCheckBox:** A checkbox component used to represent boolean options.
8. **JRadioButton:** A radio button component used to select one option from a group of options.

What is a swing container?
A Swing container is a component that can hold and organize other components within a Java GUI application. It serves as a structural element for arranging user interface elements such as buttons, text fields, labels, and more. Examples include JFrame, JPanel, JTabbedPane, and JScrollPane. Containers provide layout management, allowing developers to control the positioning and sizing of components within the user interface, facilitating the creation of visually appealing and functional desktop applications in Java.

What is a swing panel?
A Swing panel, represented by the JPanel class, is a lightweight container used in Java GUI applications to organize and group other Swing components. It provides a flexible way to structure the user interface, supporting various layout managers for precise control over component positioning and resizing.

Explain event handling in swing?
Event handling in Swing involves responding to user actions, such as clicking a button or typing in a text field, by implementing event listeners and handlers. Swing provides a range of listener interfaces and adapter classes, allowing developers to register event listeners and define custom event handling logic to create interactive and responsive GUI applications.