🏠_Home_

Java Interview Q&A - 4

01. What are the differences between ‘HashMap’ and ‘TreeMap’ ?

HashMap:
- It may have one null key and multiple null values.
- It maintains no order.

TreeMap:
- It cannot have a null key but can have multiple null values.
- It maintains ascending order.

🔗 MapExample

02. Can you explain the difference between ‘equals’ and ‘==’ ?

Q&A 01 - 39

🔗 EqualsMethod_vs_EqualSymbol

03. What are design patterns? Can you name and describe a few common ones used in Java?

Design Patterns: Elegant solution to repeating problems in software design.

(Software හද්දිදි නැවත නැවත එන ගැටලු වලට තියෙන විසඳුම්)

ex: Singleton, Factory, Facade, Decorator, Observer, Strategy

🔗 AD 2

🔗 TheMIU/Design-Patterns-Impl

04. Can you describe the different types of inner classes in Java?
- **Static Nested Class:** Associated with the outer class but instantiated independently.
- **Non-static Nested Class (Inner Class)**: Associated with an instance of the outer class.
- **Local Inner Class**: Defined inside a block or method.

- **Anonymous Inner Class:** A class without a name, often used for instantiating interfaces or abstract classes.

🔗 inner_class

05. Can you explain Java's garbage collection process?
JVM එකේ තියෙන garbage collector එකෙන් unreferenced objects හඳුනගෙන, heap memory එකෙන් ඉවත් කරයි.

🔗 GarbageCollectorExample

06. What is Java 8's Stream API and how is it different from a collection?
Stream API: Introduced functional-style operations on streams of elements.

Difference from Collection:
Stream doesn't store elements; it processes elements on-the-fly, supporting functional-style operations like **map**, **filter**, and **reduce**.

🔗 stream_api

07. What is the diamond problem in inheritance and how does Java solve it?
Java doesn't support multiple inheritance for classes.

(class එකක් multiple interfaces වලින් implement කරන්න පුලුවන්.)

08. How does Java handle multiple inheritance?
Achieved through interfaces, allowing a class to implement multiple interfaces.
(class එකක් multiple interfaces වලින් implement කරන්න පුලුවන්.)

🔗 DiamondProblem

09. Can you describe the Observer design pattern and its usage in Java?
Object අතර තියෙන one-to-many dependency එකක් අදහස් කරයි.
Subject object එකේ වෙනස්කමක් වුනොත්, ඒකත් එක්ක dependency තියෙන අනිත් observer objects වලටත් ඒ වෙනස්කම notify වෙලා automatically update වෙනෝ.

distributed systems, event handling systems, GUI components වල use වේ.

🔗 ObserverTest

10. What are the differences between a HashSet, TreeSet, and LinkedHashSet?

    HashSet
    - Uses HashMap to store objects.
    - It doesn’t maintain order.

    LinkedHashSet
    - Uses LinkedHashMap to store objects.
    - It maintains order.

    TreeSet
    - Uses TreeMap to store objects.
    - It maintains order (Sorted to ascending by default)

    🔗 Collection Framework

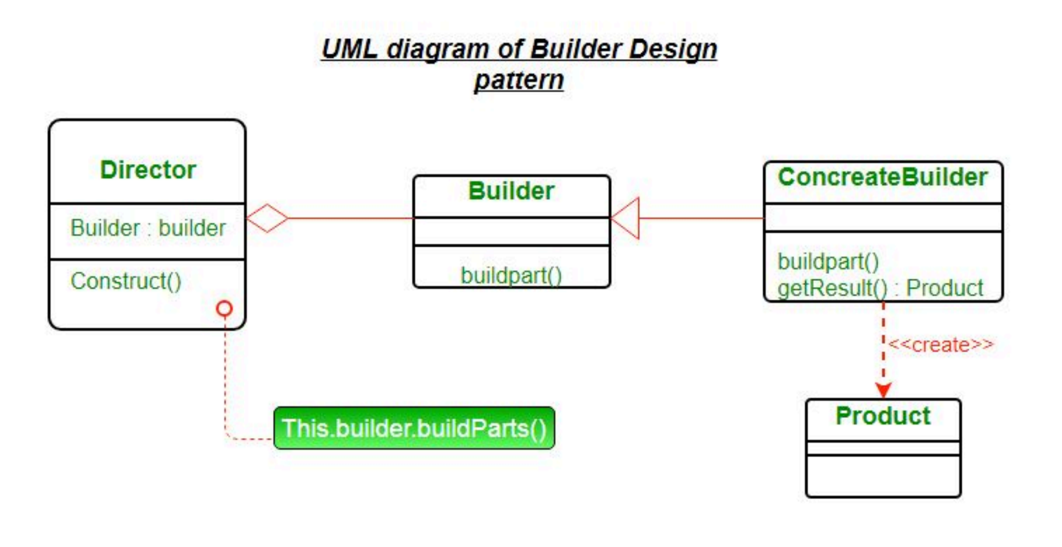
11. How does JIT compilation work in Java?

    Runtime එකේදී, Java bytecode එක, machine code එකට convert කරයි.

12. Can you describe the Builder pattern and its implementation in Java?

    Pattern: Creational pattern used to construct a complex object step by step.

    Implementation: Involves a builder class with methods to set properties and a director class to manage the construction process.



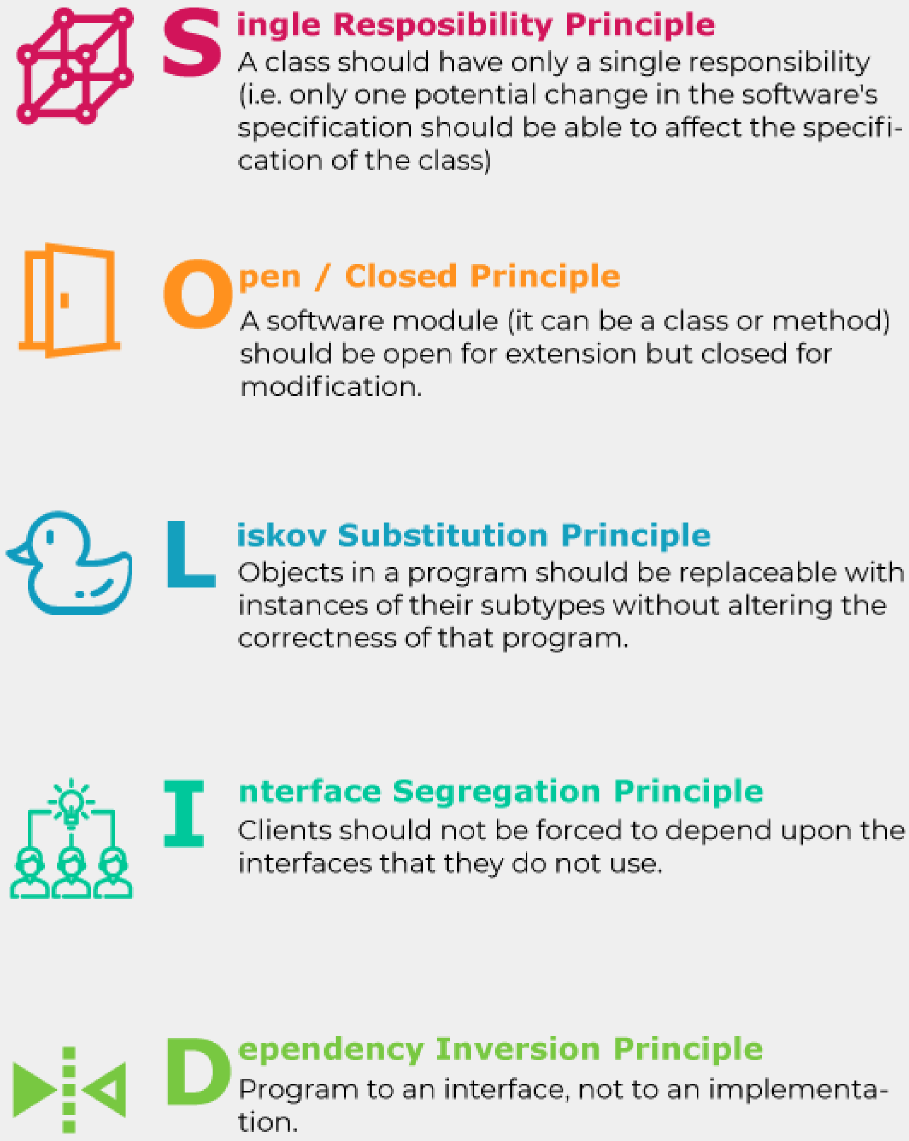
🔗 BuilderPatternExample

13. Can you describe the Factory Method design pattern and its implementation in Java?
    Object creation logic එක hide කරයි.

    Key Components
    1. Products (Product interface & Concrete products)
    2. Factory (Factory interface & Concrete factory)

    Implementation:
    🔗 FactoryTest

14. What are the principles of SOLID in object-oriented programming?



Single Resposibility Principle

A class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)

Open / Closed Principle

A software module (it can be a class or method) should be open for extension but closed for modification.

Liskov Substitution Principle

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

Interface Segregation Principle

Clients should not be forced to depend upon the interfaces that they do not use.

Dependency Inversion Principle

Program to an interface, not to an implementation.

Rules නෙවේ, best practices.

1. **S**ingle Responsibility Principle (SRP)
   එක class එකකට තියෙන්න පුලුවන් එක responsibility එකයි.

2. **O**pen/Closed Principle (OCP)
   Class එකක්, ලේසියෙන් extend කරන්න පුලුවන් වෙන්න ඕන. Modify කරන එක අමාරු වෙන්න ඕන.

3. **L**iskov Substitution Principle (LSP)

ඕනම subclass එකක්, parent class object එකකින් replace කරන්න පුලුවන් වෙන්න ඕන, functionality break වෙන්නෙ නැතුව.

4. **I**nterface Segregation Principle (ISP)
   interface එකක් use කරන කනොට, ඒකෙ හැම function එකක්ම override කරන්න කියල force නෙ◌ාකරන්න ඕන.

5. **D**ependency Inversion Principle (DIP)
   Code එකක, high level module, low level module එක්ක dependency එකක් තියෙන්න බැ කියන එක. (depend වෙන්න ඕන interface එකක් එක්ක)

🔗 ▶ SOLID Principles යනු මෙ◌ානවාද? - SOLID Principles in Sinhala
🔗 SOLID_principles

15. What is meant by loose coupling in programming and how does Java promote it?
    Meaning:
    Reducing dependencies between components or modules.

    Java Promotion:
    Dependency Injection, Interface orientation through Loose coupling promote කරයි.

    🔗 Spring

16. How do you handle transaction management in Java?
    Relational DB වලදි, JDBC වල java.sql.Connection interface එකෙන්.
    Spring වලදි @Transactional annotation එකෙන්.

17. How can you use Java to read and write files?
    Reading: Use FileReader, BufferedReader, Scanner, or Files class.
    Writing: Use FileWriter, BufferedWriter, PrintWriter, or Files class.

18. What is meant by thread safety and how is it ensured in Java?
    Meaning: Ensuring that shared resources can be accessed concurrently without causing data corruption.

Java Methods: Synchronization using the synchronized keyword, or using thread-safe classes from java.util.concurrent package.

🔗 SynchronizationExample

19. What is the difference between a Java library and a Java framework?
    Library: A collection of pre-written code that can be used in various projects.
    (ex: JDBC, Lombok, Jackson, JUnit)

    Framework: An integrated set of software tools and components providing a foundation for building applications.
    (ex: Spring, Spring Boot, Hibernate)

20. What are some popular libraries in Java for handling JSON?
    Jackson: Provides JSON parsing and generation.
    Gson: Google's library for JSON parsing.

    🔗 Spring

21. How does the Java Memory Model work?
    The Java Memory Model (JMM) defines the allowable behavior of multithreaded programs, and therefore describes when such reorderings are possible. It places execution-time constraints on the relationship between threads and main memory in order to achieve consistent and reliable Java applications.

    Definition: Describes how threads interact through memory.
    Visibility: Guarantees visibility of changes made by one thread to other threads.
    Atomicity: Operations like reading and writing are atomic for reference variables.

    🔗 Java Memory Model in 10 minutes

22. How can we ensure that a class is immutable in Java?

Rules: Declare class as final, make fields private and final, avoid mutator methods, and ensure deep immutability for mutable fields.

🔗 ImmutableClass

23. Can you explain what is function currying in Java?
    Definition: Transforming a function that takes multiple arguments into a sequence of functions, each taking a single argument.
    Example: Using functional interfaces and lambda expressions.

    🔗 ▶ 16 Using Currying in Java
    🔗 CurryingExample

24. How is string immutability beneficial in Java?
    Thread Safety:
    In a multi-threaded environment, when different threads access the same String/StringBuffer object, each thread sees the same value, and none can alter the original content.
    (Immutable - වෙනස් කළ නොහැකියි.)

    Caching: Allows caching and optimization of string literals.

25. What are the rules for method overloading and overriding in Java?
    Method overloading:
    * එකම class එකක් ඇතුලෙ
    * same name - different parameter count
    * same name - different parameter type
    * method හදන්න පුලුවන්.

    🔗 MethodOverloadingExample

    Method Overriding:
    * Class 2 ක් extend වලො තියෙද්දි,
    * Same name - Same Parameter count - Same parameter type
    * Method හදන්න පුලුවන්.
    (trough inheritance)

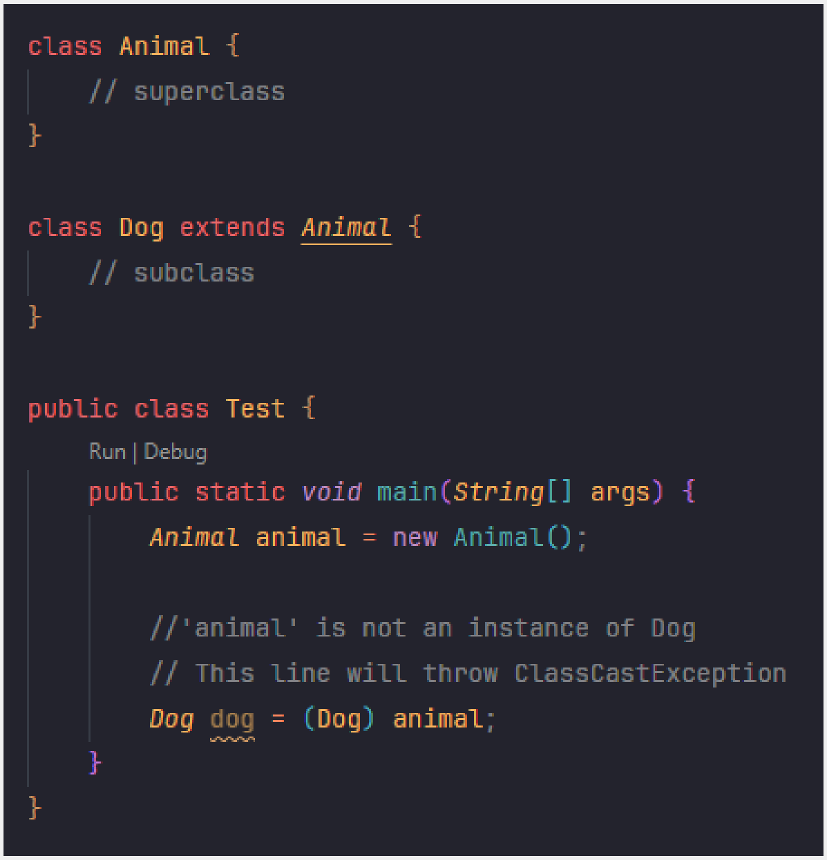
26. What are marker interfaces in Java and why are they used?

Definition: Interfaces with no methods (e.g., **Serializable**, Cloneable).

Usage: Indicate a capability or behavior. They serve as a marker for the compiler or runtime.
(මොකක් හරි විශේෂ behavior එකක් තියෙන class එකක් mark කරන්න use කරයි.)

Q&A 02 - 11

27. Can you explain what is ‘ClassCastException’ in Java?
Definition: Thrown when an object is cast to an incompatible type.
Reasons: Attempting to cast an object of a class to a type it is not a subclass of.



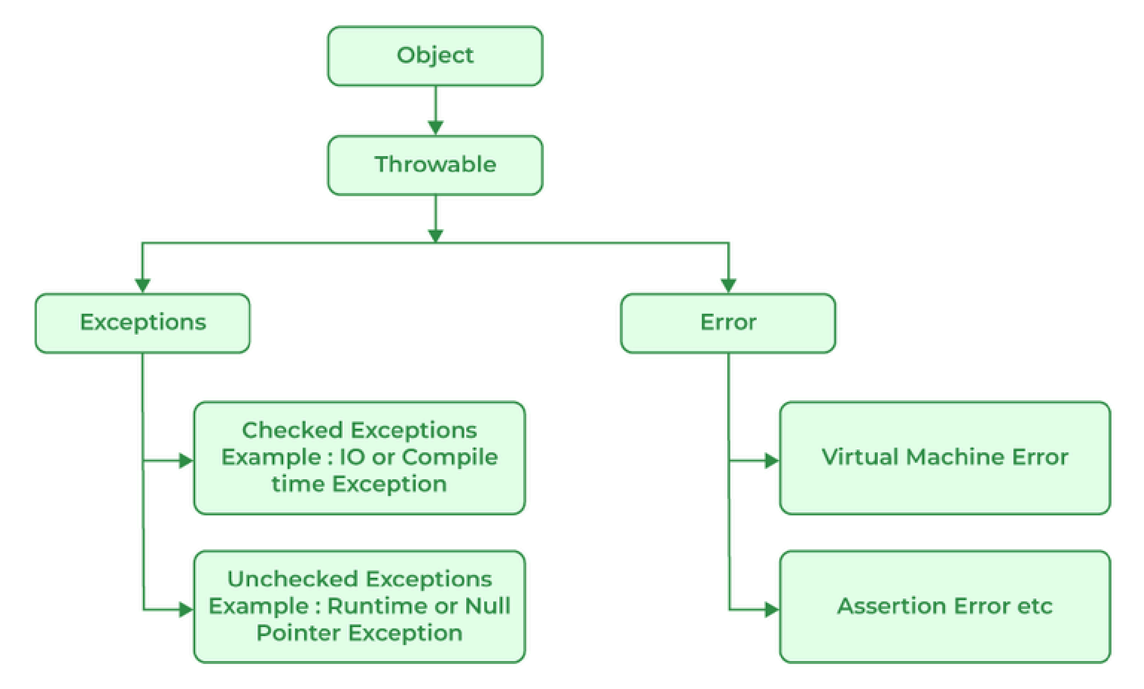
```java
class Animal {
    // superclass
}

class Dog extends Animal {
    // subclass
}

public class Test {
    public static void main(String[] args) {
        Animal animal = new Animal();

        //'animal' is not an instance of Dog
        // This line will throw ClassCastException
        Dog dog = (Dog) animal;
    }
}
```

28. Can you describe Java's exception hierarchy?


Object
Throwable
Exceptions
Error
Checked Exceptions Example : IO or Compile time Exception
Unchecked Exceptions Example : Runtime or Null Pointer Exception
Virtual Machine Error
Assertion Error etc

Throwable (Root): Error and Exception.
Error (Unchecked): Irrecoverable issues (e.g., OutOfMemoryError).
Exception (Checked/Unchecked): Exceptional conditions requiring handling (e.g., IOException).

29. What are the differences between ArrayList and LinkedList in Java?
    ArrayList: Dynamic array, fast random access, slower for insertions and deletions.
    LinkedList: Doubly linked list, fast insertions and deletions, slower random access.

30. How can you create a thread-safe singleton in Java?
    Two ways to do that,
    1. Double-Checked Locking (Lazy Initialization)
    2. Bill Pugh Singleton Pattern (Initialization-on-demand holder idiom)

    🔗 thread_safe_singleton

31. What are the different types of thread states in Java?
    New: Created but not started.
    Runnable: Ready to run, waiting for thread scheduler.
    Blocked: Waiting for a monitor lock.

Waiting: Waiting indefinitely for another thread to perform a specific action.
Timed Waiting: Waiting for a specified amount of time.
Terminated: Execution completed.

Q&A 03 - 08

32. How does the Java 8 Date and Time API improve upon the older date and time classes?
Improvements: Addresses design flaws in older classes (Date and Calendar).
Immutability: New classes like LocalDate and LocalTime are immutable.
Clarity: Separate classes for date, time, and datetime. Improved API for manipulation and formatting.

33. How can you use Regular Expressions in Java?
Usage: Pattern matching and manipulation of strings.
Classes: Pattern for compiling regex, Matcher for matching operations.
Methods: matches(), find(), replaceAll(), etc.

🔗 RegexExample

34. Can you describe the structure and components of a Java class?
Structure: Package declaration, imports, class declaration, fields, methods, constructors.
Components: Members (fields and methods), constructors, initializers, nested classes.

35. What is the purpose of a Java package and how is it used? Can you explain the naming convention associated with it?
Purpose: Group related classes and provide namespace management.
Naming Convention: Lowercase, reverse domain name, followed by project-specific package names (e.g., com.example.project).

36. Can you explain the life cycle of a Java object?
Creation: Object instantiation.
Usage: Object is actively used.

Abandonment: Object becomes unreachable.
Garbage Collection: Memory is reclaimed by the garbage collector.

🔗 ObjectLifeCycleExample

37. What are Java's bitwise and bit shift operators?

Bitwise Operators:

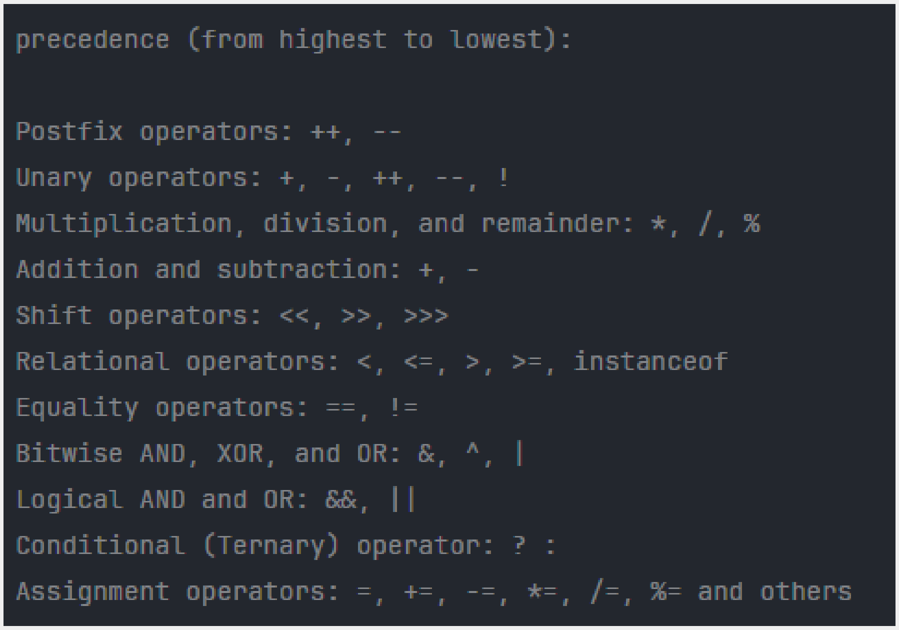
& (AND), | (OR), ^ (XOR), ~ (NOT)

Bit Shift Operators:

<< (left shift), >> (right shift), >>> (unsigned right shift).

38. Can you explain the order of operator precedence and associativity rules in Java?

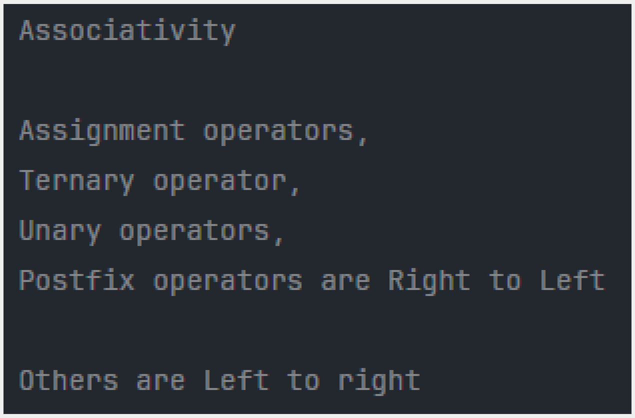
Precedence: Higher precedence means an operator is applied first.



```
precedence (from highest to lowest):

Postfix operators: ++, --
Unary operators: +, -, ++, --, !
Multiplication, division, and remainder: *, /, %
Addition and subtraction: +, -
Shift operators: <<, >>, >>>
Relational operators: <, <=, >, >=, instanceof
Equality operators: ==, !=
Bitwise AND, XOR, and OR: &, ^, |
Logical AND and OR: &&, ||
Conditional (Ternary) operator: ? :
Assignment operators: =, +=, -=, *=, /=, %= and others
```

Associativity: Describes the order in which operators of the same precedence are evaluated (left-to-right or right-to-left).



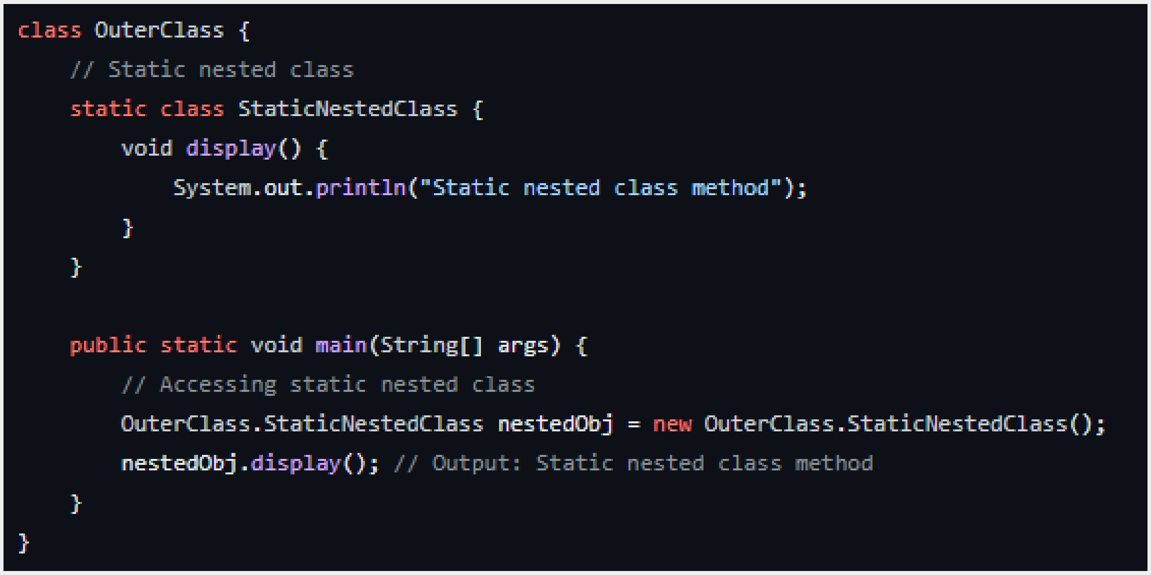
```
Associativity

Assignment operators,
Ternary operator,
Unary operators,
Postfix operators are Right to Left

Others are Left to right
```

🔗 OperatorPrecedenceExample
🔗 Java Operator Precedence - Javatpoint

39. What is a static nested class in Java and how it differs from top-level class?
    Definition: A nested class declared as static within another class.
    Difference: Can be instantiated without an instance of the outer class, while a non-static nested class requires an instance.



```java
class OuterClass {
    // Static nested class
    static class StaticNestedClass {
        void display() {
            System.out.println("Static nested class method");
        }
    }

    public static void main(String[] args) {
        // Accessing static nested class
        OuterClass.StaticNestedClass nestedObj = new OuterClass.StaticNestedClass();
        nestedObj.display(); // Output: Static nested class method
    }
}
```

🔗 StaticNestedClass

40. Can you explain how the Java 8 foreach() method works?
    Q&A 03 - 13

41. What are the differences between the '&' and '&&' operators in Java?

'&' (Bitwise AND):

- Evaluates both operands, even if the left operand is false.

'&&' (Conditional AND):

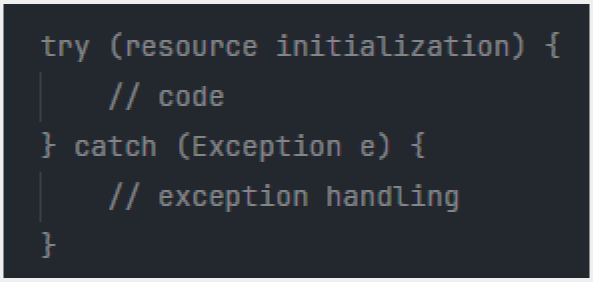
- Uses short-circuit evaluation.
- If the first operand is false, the second operand is not evaluated.
- Good for performance

🔗 Bitwise_vs_Conditional_AND

42. Can you explain how to use Java's try-with-resources feature?

Usage: Automatic resource management in try-catch blocks.

Syntax:



```
try (resource initialization) {
    // code
} catch (Exception e) {
    // exception handling
}
```

Benefit:

- Automatically closes resources like files, sockets, etc., after the try block execution.
- More readable code and easy to write.
- Number of lines of code is reduced.

🔗 TryWithResourcesExample

43. What is the purpose of a constructor in Java?

Purpose: Initializes the object's state when an instance is created.

Constructor characteristics

- Class එකේ නමින් හැදනෙ
- Object creation එකේදි invoke වනෙ
- Return type එකක් නැති
- special method එකක්

44.How does a PriorityQueue work in Java and where is it used?

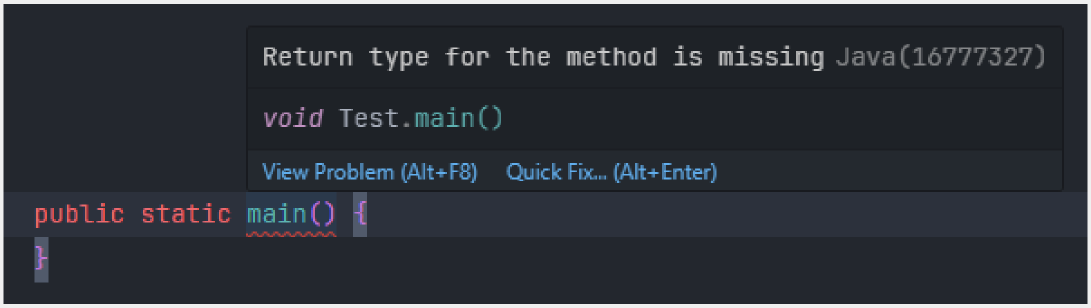
Definition: Implements a priority queue based on the natural ordering or a specified comparator.
Usage: Often used in task scheduling and graph algorithms where elements with higher priority are processed first.

🔗 PriorityQueueExample

45.Can you explain the difference between ' public static void main' and ' public static main ' in Java?

'public static void main': Standard signature for the entry point of a Java application. The method returns no value (void).

'public static main': Invalid syntax. Return type for the method is missing



```
Return type for the method is missing Java(16777327)

void Test.main()
```

```
public static main() {
}
```

46.How do you create a custom annotation in Java?

Syntax: @interface CustomAnnotation { /* elements */ }
Usage: Apply to classes, methods, fields, etc., using @CustomAnnotation.

🔗 custom_annotation

47.Can you explain the Liskov Substitution Principle and its importance in Java programming?

Definition: Subtypes should be substitutable for their base types without affecting program correctness.

Importance: Enables polymorphism and ensures that inheriting classes can be used interchangeably with their base classes.

🔗 SOLID_principles

48. How does bounded type parameters work in Java Generics?
    Usage: Limit the types that can be used as generic arguments.
    Syntax: <T extends SomeType> (upper bound) or <T super SomeType> (lower bound).
    🔗 BoundedTypeExample

49. What are the differences between List<Object> and List<?> in Java Generics?
    List<Object> : allows elements of any type, and type information is lost when retrieving.
    List<?> : allows elements of unknown type, and type information is preserved when retrieving. (cannot add elements except null)

    🔗 ObjectType_vs_UnknownType

50. Can you explain the difference between <? super T> and <? extends T>?
    <? super T>
    - lower-bounded wildcard
    - allowing objects of type T or its supertypes.

    <? extends T>
    - upper-bounded wildcard
    - allowing objects of type T or its subtypes.

    🔗 Upper_Lower_WildcardExample