Great! Let's begin with:

✅ Chapter 1: Introduction to Java

Total Questions: 75
Levels: 25 Easy | 25 Medium | 25 Hard
Answer Key Included

🟢 EASY LEVEL (1–25)

1. Java was developed by:
   a) Dennis Ritchie
   b) Bjarne Stroustrup
   c) James Gosling
   d) Guido van Rossum
   Ans: c

2. Java is considered:
   a) A low-level language
   b) A scripting language
   c) A high-level, object-oriented language
   d) A machine language
   Ans: c

3. The principle of hiding internal details and showing functionality is:
   a) Inheritance
   b) Encapsulation
   c) Abstraction
   d) Polymorphism
   Ans: c

4. In Java, class is:
   a) An instance of an object
   b) A data type
   c) A blueprint for objects
   d) A method
   Ans: c

5. What is the default value of an uninitialized `int` variable in a class?
   a) NULL
   b) 1
   c) 0

d) Undefined
Ans: c

6. Which of these is not a feature of Java?
   a) Platform-independent
   b) Secure
   c) Pointer support
   d) Object-oriented
   Ans: c

7. What is the output file created after Java compilation?
   a) .exe
   b) .obj
   c) .class
   d) .javac
   Ans: c

8. The main function of JVM is to:
   a) Compile code
   b) Interpret bytecode
   c) Store files
   d) Connect to DB
   Ans: b

9. Which of these is a Java primitive type?
   a) String
   b) Integer
   c) boolean
   d) Date
   Ans: c

10. A real-world example of an object is:
    a) method
    b) variable
    c) car
    d) JVM
    Ans: c

11. Java Virtual Machine executes:
    a) .java
    b) .exe
    c) .class
    d) .txt
    Ans: c

12. Platform independence means:
    a) Runs only on Windows
    b) Needs installation on every machine

c) Write once, run anywhere
d) Must compile for each OS
Ans: c

13. The smallest unit in OOP is:
a) Method
b) Class
c) Object
d) Variable
Ans: c

14. The blueprint of an object is called:
a) Object
b) Method
c) Class
d) Structure
Ans: c

15. JVM is:
a) Software
b) Hardware
c) Compiler
d) File system
Ans: a

16. Which is not a type of memory in JVM?
a) Stack
b) Heap
c) Queue
d) Code segment
Ans: c

17. Encapsulation binds:
a) Code only
b) Data only
c) Code and data together
d) Only functions
Ans: c

18. Which is not a pillar of OOP?
a) Inheritance
b) Polymorphism
c) Compilation
d) Encapsulation
Ans: c

19. Which keyword is used to define a class?
a) struct

b) class
c) object
d) define
Ans: b

20. What is used to create an object in Java?
a) malloc
b) new
c) alloc
d) create
Ans: b

21. What keyword represents current object?
a) self
b) this
c) super
d) that
Ans: b

22. Which memory stores objects in Java?
a) Stack
b) Heap
c) Code segment
d) ROM
Ans: b

23. The term “object” refers to:
a) Class instance
b) Function
c) Variable
d) Package
Ans: a

24. OOP concept that allows multiple forms is:
a) Abstraction
b) Polymorphism
c) Encapsulation
d) Association
Ans: b

25. The extension of compiled Java file is:
a) .java
b) .jar
c) .class
d) .exe
Ans: c

Great! Here are the:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 1: Introduction to Java

---

26. Which of the following statements best describes abstraction in Java?
    a) Hiding implementation details and showing essential features
    b) Hiding both data and implementation
    c) Hiding code with passwords
    d) Hiding classes
    Ans: a

27. Which concept helps in managing complexity in large software systems?
    a) Arrays
    b) Abstraction
    c) Looping
    d) Variables
    Ans: b

28. What does the term “Platform Independent” mean in Java?
    a) Can be compiled only on Linux
    b) Can run only on one type of hardware
    c) Compiled bytecode can run on any OS with JVM
    d) Must be installed using Java Setup
    Ans: c

29. What is not true about objects in Java?
    a) Every object has state and behavior
    b) Objects can exist without a class
    c) Objects are instances of classes
    d) Objects interact with other objects
    Ans: b

30. The correct definition of encapsulation is:
    a) Wrapping of data and methods together
    b) Using getter methods only
    c) Wrapping data only
    d) Using classes only
    Ans: a

31. Which part of Java translates `.java` files into `.class` files?
    a) JVM
    b) JDK
    c) JRE
    d) Compiler
    Ans: d

32. Which is **not** a feature of Java?
    a) Machine-dependent
    b) Secure
    c) Multithreaded
    d) Distributed
    Ans: a

33. The memory area where instance variables are stored is:
    a) Stack
    b) Heap
    c) Code
    d) Register
    Ans: b

34. Which concept allows an object to acquire the properties of another object?
    a) Polymorphism
    b) Inheritance
    c) Abstraction
    d) Encapsulation
    Ans: b

35. Which of the following JVM activities **does not** occur at runtime?
    a) Loading code
    b) Executing code
    c) Compiling `.java` files
    d) Providing runtime environment
    Ans: c

36. Java achieves "Compile once, run anywhere" by using:
    a) .exe files
    b) Source code portability
    c) Bytecode interpreted by JVM
    d) XML
    Ans: c

37. How does garbage collection work in Java?
    a) Deletes all files
    b) Deletes unused objects from heap memory
    c) Removes variables from stack
    d) Clears the screen

Ans: b

38. The Mark and Sweep method of garbage collection involves:
    a) Only collecting used memory
    b) Marking reachable objects and sweeping the rest
    c) Manual deletion
    d) Thread-safe destruction
    Ans: b

39. Which of these components verifies and loads Java bytecode?
    a) Java Compiler
    b) JVM
    c) OS
    d) JDK
    Ans: b

40. Which characteristic does **not** belong to OOP?
    a) Inheritance
    b) Encapsulation
    c) Compilation
    d) Polymorphism
    Ans: c

41. Consider `Person p = new Person();`. Here `p` is:
    a) Object
    b) Class
    c) Constructor
    d) Reference variable
    Ans: d

42. Which keyword is used to create an object in Java?
    a) object
    b) create
    c) this
    d) new
    Ans: d

43. Which of the following statements is **true** about class and object in Java?
    a) Class is the instance of an object
    b) Object is the method of class
    c) Class is blueprint, object is instance
    d) Both are same
    Ans: c

44. Which OOP concept promotes code reusability?
    a) Abstraction
    b) Polymorphism

c) Inheritance
d) Encapsulation
Ans: c

45. Which area in memory is used for static variables?
a) Stack
b) Heap
c) Data segment
d) Code segment
Ans: c

46. Which generation of memory is used for newly created objects in Java?
a) Old Generation
b) Nursery / Young Generation
c) Stack
d) Static Area
Ans: b

47. An example of specialization is:
a) Object → Class
b) Manager → Employee
c) Class → Object
d) Data → Code
Ans: b

48. An object in Java is:
a) An entity that holds only methods
b) A real-world entity with structure and behavior
c) A reference only
d) Only a value
Ans: b

49. JVM is responsible for:
a) Compiling Java files
b) Executing machine code
c) Interpreting bytecode and running Java programs
d) Generating class diagrams
Ans: c

50. Which of the following best describes "Generalization"?
a) Combining subclasses into a superclass
b) Creating multiple instances
c) Using more objects
d) Defining constants
Ans: a

Perfect! Here are the:

🔴 HARD LEVEL MCQs (51–75)

Chapter 1: Introduction to Java

---

51. What happens if a class is defined but no object is ever created from it?
    a) Compilation fails
    b) JVM throws error
    c) Class consumes memory
    d) Class exists with no memory allocation
    Ans: d

52. Which OOP concept allows the same method name to behave differently across related classes?
    a) Overloading
    b) Encapsulation
    c) Polymorphism
    d) Inheritance
    Ans: c

53. What distinguishes abstraction from encapsulation?
    a) Abstraction focuses on *how* to implement; encapsulation hides *data*
    b) Abstraction hides internal details; encapsulation binds data and methods
    c) Both are the same
    d) Encapsulation is used only in inheritance
    Ans: b

54. JVM verifies bytecode before execution to:
    a) Reduce memory usage
    b) Prevent logical errors
    c) Prevent illegal access and maintain security
    d) Optimize performance
    Ans: c

55. Which best describes the difference between heap and stack memory?
    a) Stack stores objects, heap stores primitive types
    b) Stack is static, heap is dynamic
    c) Stack is for method calls and local variables, heap is for objects
    d) Stack is for threads, heap is for constants
    Ans: c

56. Which of the following are characteristics of an object?
    a) State and structure

b) Identity and behavior
c) Only identity
d) State, behavior, identity, and responsibility
Ans: d

57. If a class has no methods, what is true about its object?
a) It cannot be created
b) It throws an error
c) It still has a memory reference
d) It executes automatically
Ans: c

58. Consider this structure: `Person → Employee → Manager`. This is an example of:
a) Encapsulation
b) Specialization
c) Polymorphism
d) Multithreading
Ans: b

59. What is a drawback of Java's garbage collection?
a) Slower execution due to unpredictable GC pause
b) Manual memory control
c) Memory leaks
d) Permanent memory allocation
Ans: a

60. Which of the following is not directly managed by the Java Memory Manager?
a) Object allocation
b) Garbage collection
c) Stack overflow detection
d) File system memory
Ans: d

61. Java's garbage collection is most efficient because:
a) It runs periodically
b) It uses both young and old generations
c) It is handled by developers
d) It tracks method calls
Ans: b

62. Which OOP principle does the "is-a" relationship relate to?
a) Encapsulation
b) Polymorphism
c) Inheritance
d) Overriding
Ans: c

63. How is abstraction implemented in Java?
   a) Through constructors
   b) Through object creation
   c) Using abstract classes and interfaces
   d) Using access modifiers
   Ans: c

64. What describes the process of moving up the class hierarchy in generalization?
   a) Creating subclasses
   b) Adding instance variables
   c) Factoring out common elements
   d) Overriding methods
   Ans: c

65. In Java, what ensures that the same class structure works on different platforms?
   a) JVM
   b) JDK
   c) JRE
   d) JavaDoc
   Ans: a

66. JVM memory structure that contains constants and static variables is called:
   a) Code segment
   b) Stack
   c) Data section
   d) Heap
   Ans: c

67. What allows a Java class to function without a `main()` method?
   a) It's an interface
   b) It's a static class
   c) It's not meant to be run standalone (like helper class)
   d) Java does not support this
   Ans: c

68. Which is true about default values of uninitialized member variables?
   a) No default values; throws exception
   b) 0 for all types
   c) Depends on OS
   d) Java sets default values like `0`, `false`, or `null` based on type
   Ans: d

69. Why can't Java objects be accessed directly from outside the class?
   a) Compilation issue
   b) JVM restriction
   c) Principle of encapsulation
   d) Only constructors can access them

Ans: c

70. Why does Java not support multiple inheritance using classes?
    a) Reduces performance
    b) Makes code unreadable
    c) To avoid ambiguity and simplify inheritance hierarchy
    d) JVM limitation
    Ans: c

71. Which of the following is false about bytecode?
    a) Platform-independent
    b) Output of Java compiler
    c) Executed by JVM
    d) Directly understood by hardware
    Ans: d

72. A class can exist without which of the following?
    a) Name
    b) Methods
    c) Constructor
    d) Object
    Ans: d

73. What does “Compile Once, Run Anywhere” imply?
    a) Java is interpreted
    b) Bytecode runs on JVM regardless of OS
    c) Java is lightweight
    d) Java files are executable
    Ans: b

74. Which two concepts together ensure implementation hiding and data protection?
    a) Abstraction and inheritance
    b) Inheritance and encapsulation
    c) Polymorphism and abstraction
    d) Encapsulation and abstraction
    Ans: d

75. Which aspect of Java makes it suitable for internet programming?
    a) Pointer support
    b) Platform dependence
    c) Security and portability via bytecode and JVM
    d) Use of Assembly
    Ans: c

Awesome! Let's now move on to:

✅ Chapter 2: Java Basics

Total Questions: 75
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Operators, Loops, Decision-Making, Access Specifiers, Methods, Constructors, etc.

🟢 EASY LEVEL (1–25)

1. Which of the following is an arithmetic operator in Java?
   a) &&
   b) ++
   c) +=
   d) !=
   Ans: b

2. The `==` operator in Java is used to:
   a) Assign values
   b) Add numbers
   c) Compare equality
   d) Negate a condition
   Ans: c

3. Which operator is used to get the remainder in Java?
   a) /
   b) %
   c) *
   d) //
   Ans: b

4. What is the default value of a boolean variable?
   a) true
   b) 1
   c) false
   d) null
   Ans: c

5. Which loop is guaranteed to execute at least once?
   a) for
   b) while
   c) do...while
   d) switch

Ans: c

6. What is the output of `++i` if i = 5?
   a) 5
   b) 6
   c) 4
   d) Error
   Ans: b

7. Which keyword is used for defining methods that don't return a value?
   a) return
   b) void
   c) static
   d) final
   Ans: b

8. In which loop is the condition checked after executing the body?
   a) for
   b) while
   c) do...while
   d) switch
   Ans: c

9. The access specifier that allows visibility only within the same class is:
   a) public
   b) protected
   c) private
   d) default
   Ans: c

10. A method that does **not** return any value uses:
    a) return null
    b) return 0
    c) void
    d) static
    Ans: c

11. The keyword used to create objects is:
    a) malloc
    b) new
    c) alloc
    d) init
    Ans: b

12. What is the entry point of any Java program?
    a) start()
    b) init()

c) main()
d) run()
Ans: c

13. The loop that executes a known number of times:
a) while
b) do...while
c) for
d) switch
Ans: c

14. Which access specifier allows class members to be accessed from anywhere?
a) default
b) protected
c) private
d) public
Ans: d

15. Java methods are defined using the keyword:
a) def
b) void
c) method
d) return
Ans: b

16. Which of the following is **not** a Java loop?
a) for
b) do...while
c) until
d) while
Ans: c

17. The `else` part of an if statement executes when:
a) Condition is true
b) Condition is false
c) Always
d) Never
Ans: b

18. The `switch` statement works on which data types?
a) float
b) boolean
c) String
d) double
Ans: c

19. `val++` is called:
  a) Post-decrement
  b) Pre-increment
  c) Post-increment
  d) Double increment
  Ans: c

20. `&&` is a:
  a) Bitwise AND
  b) Logical AND
  c) Assignment operator
  d) Relational operator
  Ans: b

21. Which operator increases the value by 1?
  a) ++
  b) +=
  c) +
  d) *
  Ans: a

22. If `x = 10; x += 5;`, then x becomes:
  a) 10
  b) 15
  c) 5
  d) Error
  Ans: b

23. A constructor is a:
  a) Method to return values
  b) Method used to initialize objects
  c) Loop
  d) Final method
  Ans: b

24. Constructors do not have:
  a) Parameters
  b) Body
  c) Return type
  d) Name
  Ans: c

25. What is the return type of the `main()` method in Java?
  a) int
  b) void
  c) String
  d) boolean

Ans: b

Perfect! Let's proceed with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 2: Java Basics

---

26. What will be the output of the following code?

```
int x = 5;

System.out.println(++x);
```

a) 5
b) 6
c) 7
d) Error
Ans: b

---

27. What happens if a method in Java does not specify a return type?
    a) Compilation error
    b) Default return value is assigned
    c) It becomes a constructor
    d) It's treated as `void`
    Ans: a

---

28. Which of the following loops is best used when the number of iterations is known?
    a) while
    b) do...while
    c) for
    d) switch
    Ans: c

---

29. What does `a &= b;` mean in Java?
    a) Assigns `a` to `b`
    b) Performs bitwise AND and assigns to `a`
    c) Compares values
    d) None of the above
    Ans: b

---

30. Which of these statements correctly declares a `char` in Java?
    a) char ch = "A";
    b) char ch = 'A';
    c) char ch = A;
    d) char ch = `A`;
    Ans: b

---

31. What is the output of the following condition:

System.out.println(10 > 5 && 6 < 4);

a) true
b) false
c) 10
d) Error
Ans: b

---

32. Which of the following correctly initializes a float variable?
    a) float f = 10.5;
    b) float f = 10.5d;
    c) float f = 10.5f;
    d) float f = "10.5";
    Ans: c

---

33. Which access specifier allows access within the same package?
    a) protected

b) private
c) default
d) public
Ans: c

---

34. The purpose of the `return` statement is to:
a) Break from a loop
b) Transfer control to caller method
c) Print values
d) Exit the program
Ans: b

---

35. What is the result of the expression: `(5 > 3) || (2 > 4)`?
a) true
b) false
c) Compilation error
d) 1
Ans: a

---

36. Which part of the `for` loop is evaluated first?
a) Condition
b) Initialization
c) Update
d) None
Ans: b

---

37. How many times will this loop run?

```
for (int i = 0; i < 5; i++) {}
```

a) 4
b) 5
c) 6

d) Infinite
Ans: b

---

38. Which operator is used for comparing references of objects?
    a) ==
    b) .equals()
    c) !=
    d) =
    Ans: a

---

39. Which operator reverses the logical value of its operand?
    a) ~
    b) !
    c) ^
    d) -
    Ans: b

---

40. Which method is used to execute code when a Java program starts?
    a) run()
    b) init()
    c) main()
    d) start()
    Ans: c

---

41. Choose the correct syntax to define a method that returns an `int` and takes two `int` parameters:
    a) int method(int x, y)
    b) int method(int x, int y)
    c) method int(int x, int y)
    d) void method(int, int)
    Ans: b

---

42. Which method modifier allows calling a method without an object?
    a) final
    b) public

c) private
d) static
Ans: d

---

43. In a `switch` statement, what happens if no `break` is written?
    a) Compiler error
    b) Only the matched case executes
    c) All subsequent cases execute (fall through)
    d) Program terminates
    Ans: c

---

44. What does the following code do?

```
int a = 10;

int b = 3;

System.out.println(a % b);
```

a) Prints 13
b) Prints 3.3
c) Prints 1
d) Compilation error
Ans: c

---

45. In Java, methods may have:
    a) Only parameters
    b) Only return types
    c) Both parameters and return types
    d) Neither
    Ans: c

---

46. Identify the unary operator in Java:
    a) +
    b) &
    c) ~

d) ++
Ans: d

---

47. Which of these is a valid method signature?
    a) int sum(int a, int b)
    b) return sum(int a, int b)
    c) method sum(int a, int b)
    d) void int sum(a, b)
    Ans: a

---

48. What is the default return type for a constructor?
    a) void
    b) class name
    c) None
    d) Object
    Ans: c

---

49. If a method is declared as `static`, which of the following is true?
    a) It can only be called inside another method
    b) It can be called without creating an object
    c) It can access `this` keyword
    d) It must be public
    Ans: b

---

50. What is the scope of a `private` method?
    a) Entire project
    b) Same package
    c) Same class
    d) Subclass
    Ans: c

Great! Let's wrap up Chapter 2 with the final:

🔴 HARD LEVEL MCQs (51–75)

Chapter 2: Java Basics

---

51. Which of the following best defines the behavior of a `for` loop in Java?
    a) Entry-controlled loop
    b) Exit-controlled loop
    c) Recursion-based loop
    d) Multithreaded loop
    Ans: a

---

52. What will be the output of the following code?

```
int x = 5;

System.out.println(x++ + ++x);
```

a) 11
b) 10
c) 12
d) 13
Ans: c

---

53. Why is `switch` preferred over long `if-else` chains in some cases?
    a) It's faster for string comparison
    b) It uses loops internally
    c) It's more readable and efficient for fixed-value comparisons
    d) It works only with integers
    Ans: c

---

54. Which of the following access specifiers can be used for top-level classes in Java?
    a) public and private
    b) public and protected
    c) public and default
    d) All of the above

Ans: c

---

55. Which of these variables will be accessible inside a static method?
    a) Instance variables
    b) Class (static) variables
    c) Local variables of another method
    d) All of the above
    Ans: b

---

56. What will be the output?

```
int a = 10;

a += (a++) + (++a);

System.out.println(a);
```

a) 32
b) 31
c) 30
d) Undefined
Ans: a

---

57. Which of the following statements about constructors is true?
    a) Constructors cannot be overloaded
    b) Constructors must always have parameters
    c) Constructors can be private
    d) Constructors return values
    Ans: c

---

58. What is the key difference between `==` and `.equals()` in Java?
    a) Both compare object references
    b) `==` compares object references; `.equals()` compares values
    c) `==` compares values only
    d) Both compare hashcodes

Ans: b

---

59. What is the correct order of execution in a `for` loop?
    a) Condition → Initialization → Update
    b) Initialization → Condition → Update
    c) Update → Condition → Initialization
    d) Condition → Update → Initialization
    Ans: b

---

60. Which of the following is a correct way to declare a method without return type and parameters?
    a) public method()
    b) void method
    c) void method()
    d) method(): void
    Ans: c

---

61. Which statement is valid regarding scope of variables in loops?
    a) Variables declared in loop are accessible outside it
    b) Variables are always global
    c) Loop variables are local to the loop block
    d) Loop variables require static declaration
    Ans: c

---

62. A method with the same name and different number/type of parameters is:
    a) Overloaded
    b) Overridden
    c) Reused
    d) Duplicated
    Ans: a

---

63. Consider:

```
int x = 10;

System.out.println(!(x > 5 && x < 20));
```

Output?
a) true
b) false
c) 10
d) Compilation error
Ans: b

---

64. A class with all private fields and public methods to access them follows which OOP principle?
    a) Polymorphism
    b) Abstraction
    c) Inheritance
    d) Encapsulation
    Ans: d

---

65. Which of these is **not** allowed inside a switch statement?
    a) String
    b) int
    c) float
    d) char
    Ans: c

---

66. How does Java treat uninitialized local variables?
    a) Automatically assigns default values
    b) Compiler error
    c) Treated as static
    d) Set to null
    Ans: b

---

67. When can we use `default` in switch statements?
    a) Only at the top
    b) Anywhere, executed if no case matches

c) Only if all other cases are false
d) Never mandatory
Ans: b

---

68. What happens if you define a method with the same name as the class and a return type?
a) It becomes a constructor
b) It becomes a method
c) Compilation fails
d) JVM executes it by default
Ans: b

---

69. Which control structure is **best** for selecting one out of many fixed options?
a) if-else
b) nested if
c) for
d) switch
Ans: d

---

70. What is the key difference between pre-increment (++i) and post-increment (i++)?
a) Post-increment changes value before usage
b) Pre-increment returns old value
c) Pre-increment increases first, post-increment increases after
d) No difference in behavior
Ans: c

---

71. Which of the following is **not** a valid method of loop termination?
a) break
b) return
c) continue
d) exit
Ans: d

---

72. What does the following do?

public static void main(String[] args)

a) Declares main class
b) Is the entry point of execution
c) Returns command line arguments
d) Declares array
Ans: b

---

73. Choose the correct method to return an integer from a method.
a) int result() { return 10; }
b) int result() { }
c) void result() { return 10; }
d) static result() { return 10; }
Ans: a

---

74. Which is a correct behavior of the `continue` statement?
a) Exits the program
b) Skips remaining loop and goes to next iteration
c) Stops compilation
d) Returns value
Ans: b

---

75. Which statement about static methods is **false**?
a) Static methods can be called without objects
b) They can access instance variables directly
c) They can be overloaded
d) They belong to the class, not object
Ans: b

Perfect! Let’s move forward with:

✅ Chapter 3: Language Fundamentals

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics: Arrays, Packages, Wrapper Classes, String, StringBuilder, StringBuffer, Immutable Classes, Autoboxing, etc.

🟢 EASY LEVEL MCQs (1–25)

1. In Java, arrays are:
   a) Objects
   b) Variables
   c) Classes
   d) Methods
   Ans: a

---

2. What is the correct syntax to declare an integer array?
   a) int arr;[]
   b) int arr[];
   c) arr int[];
   d) int[]; arr
   Ans: b

---

3. Which keyword is used to import built-in Java packages?
   a) include
   b) use
   c) import
   d) package
   Ans: c

---

4. Wrapper classes are used to:
   a) Wrap primitive data types into objects
   b) Store data permanently
   c) Create threads
   d) Inherit classes
   Ans: a

---

5. Which of the following is a wrapper class for `int`?
   a) Int
   b) Integer
   c) intWrapper
   d) WrapperInt
   Ans: b

---

6. What is the output type of `String.charAt(2)`?
   a) int
   b) String
   c) char
   d) void
   Ans: c

---

7. Which is not a valid array declaration?
   a) int[] x = new int[5];
   b) int[] x = {1, 2, 3};
   c) int x[] = new int();
   d) int x[] = new int[10];
   Ans: c

---

8. Java Strings are:
   a) Mutable
   b) Immutable
   c) Objects and primitive
   d) Final and static
   Ans: b

---

9. Which method returns the length of a string?
   a) size()
   b) getLength()
   c) length()
   d) len()
   Ans: c

---

10. What is the result of `"hello".toUpperCase()`?
    a) Hello
    b) hello
    c) HELLO
    d) hELLO
    Ans: c

---

11. `StringBuilder` is:
    a) Thread-safe
    b) Synchronized
    c) Mutable but not synchronized
    d) Immutable and thread-safe
    Ans: c

---

12. Which class is synchronized?
    a) StringBuilder
    b) StringBuffer
    c) String
    d) Arrays
    Ans: b

---

13. What does `Integer.parseInt("123")` return?
    a) "123"
    b) 123
    c) Integer object
    d) Error
    Ans: b

---

14. Auto-boxing converts:
    a) Object to primitive
    b) String to primitive
    c) Primitive to object
    d) Method to class
    Ans: c

15. Which is a method of the String class?
   a) append()
   b) reverse()
   c) equals()
   d) insert()
   Ans: c

---

16. What is the output of `"Java".concat("Code")`?
   a) Java Code
   b) JavaCode
   c) CodeJava
   d) Java+Code
   Ans: b

---

17. How do you compare two strings for equality?
   a) ==
   b) equals()
   c) compare()
   d) isEqual()
   Ans: b

---

18. StringBuffer is used when:
   a) We want immutable strings
   b) We want fast string comparisons
   c) We need mutable and thread-safe strings
   d) We don't want synchronization
   Ans: c

---

19. Which method adds characters to the end of a StringBuilder?
   a) add()
   b) join()
   c) append()
   d) concat()
   Ans: c

---

20. What does `String[] arr = new String[5];` do?
    a) Initializes 5 null elements
    b) Initializes 5 empty strings
    c) Assigns 0 to all
    d) Error
    Ans: a

---

21. The keyword to create your own package is:
    a) package
    b) import
    c) define
    d) create
    Ans: a

---

22. Which class provides methods like `toLowerCase()` and `toUpperCase()`?
    a) String
    b) Scanner
    c) Integer
    d) Object
    Ans: a

---

23. Wrapper class for boolean is:
    a) Boolean
    b) Bool
    c) WrapperBoolean
    d) booleanObject
    Ans: a

---

24. What will be stored in memory when `String s = "Hello";` is executed?
    a) Object in heap
    b) Object in stack
    c) Literal in pool
    d) Both a and c
    Ans: d

---

25. The method `split()` in String returns:
    a) char[]
    b) StringBuilder
    c) String[]
    d) List
    Ans: c

Great! Let's move ahead with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 3: Language Fundamentals

---

26. What is the output of the following code?

```
String s1 = "abc";

String s2 = "abc";

System.out.println(s1 == s2);
```

a) true
b) false
c) Compilation error
d) Runtime exception
Ans: a

---

27. What is the output of the code below?

```
String s1 = new String("abc");

String s2 = new String("abc");

System.out.println(s1 == s2);
```

a) true
b) false
c) Compilation error
d) null
Ans: b

---

28. Which of the following methods is used to compare the content of two strings?
    a) equals()
    b) ==
    c) compare()
    d) match()
    Ans: a

---

29. Consider the following code snippet:

```
Integer i = 100;

int j = i;
```

This is an example of:
a) Unboxing
b) Boxing
c) Auto-casting
d) Encapsulation
Ans: a

---

30. Which package contains the `String` class?
    a) java.io
    b) java.util
    c) java.lang
    d) java.string
    Ans: c

---

31. Which wrapper class method converts a string to primitive int?
    a) Integer.valueOf()

b) Integer.parseInt()
c) Integer.convert()
d) Integer.intValue()
Ans: b

---

32. Which is the correct way to create a StringBuilder with initial content?
a) StringBuilder sb = new StringBuilder(); sb.add("Java");
b) StringBuilder sb = new StringBuilder("Java");
c) StringBuilder sb = "Java";
d) new StringBuilder.add("Java");
Ans: b

---

33. What happens when you try to modify a String object?
a) It changes the original string
b) It throws an exception
c) It creates a new object
d) It clears the value
Ans: c

---

34. Which of the following classes is mutable?
a) String
b) StringBuffer
c) Arrays
d) Character
Ans: b

---

35. How is immutability achieved in the String class?
a) Making class `final`
b) Not providing any setters
c) Storing characters in a final array
d) All of the above
Ans: d

---

36. How can you convert a primitive int to Integer object manually?
    a) Integer.valueOf(int)
    b) Integer.parse(int)
    c) new Integer(int)
    d) Both a and c
    Ans: d

---

37. Which of the following is not a valid method of String class?
    a) append()
    b) charAt()
    c) indexOf()
    d) substring()
    Ans: a

---

38. Which of the following converts an int to a String?
    a) String.valueOf(int)
    b) Integer.toString(int)
    c) "" + int
    d) All of the above
    Ans: d

---

39. How can we convert an array of characters into a String?
    a) Using new String(char[]) constructor
    b) Using append()
    c) Using String.join()
    d) Using split()
    Ans: a

---

40. Which statement about the String pool is correct?
    a) It stores all String objects
    b) It is in heap memory
    c) It allows sharing of common strings to save memory
    d) It is part of java.util
    Ans: c

---

41. Given:

```
String s = "Java";

s.concat("Lang");

System.out.println(s);
```

What is printed?
a) JavaLang
b) Lang
c) Java
d) Error
Ans: c

---

42. Which method is used to remove all characters from a StringBuilder?
    a) remove()
    b) deleteAll()
    c) clear()
    d) setLength(0)
    Ans: d

---

43. What is the initial capacity of a StringBuilder object if not specified?
    a) 0
    b) 16
    c) 32
    d) Depends on JVM
    Ans: b

---

44. What is the result of this?

```
StringBuilder sb = new StringBuilder("abc");

sb.reverse();

System.out.println(sb);
```

a) abc
b) cba
c) Compilation error
d) NullPointerException
Ans: b

---

45. Which class is thread-safe and mutable?
    a) String
    b) StringBuffer
    c) StringBuilder
    d) CharSequence
    Ans: b

---

46. Which keyword is used to define a custom package in Java?
    a) include
    b) import
    c) define
    d) package
    Ans: d

---

47. Which method removes characters from a StringBuilder?
    a) delete()
    b) remove()
    c) trim()
    d) erase()
    Ans: a

---

48. What happens if `null` is passed to `String.valueOf()`?
    a) "null" is returned
    b) Error
    c) 0 is returned
    d) NullPointerException
    Ans: a

---

49. What is `Integer.MAX_VALUE`?
   a) 32,767
   b) 2,147,483,647
   c) 65,535
   d) Depends on system
   Ans: b

---

50. Which interface is implemented by String, StringBuffer, and StringBuilder?
   a) Appendable
   b) Serializable
   c) CharSequence
   d) Comparable
   Ans: c

Awesome! Here's the final set for Chapter 3:

🔴 HARD LEVEL MCQs (51–75)

Chapter 3: Language Fundamentals

---

51. What is the effect of the following code?

```
String s1 = "abc";

String s2 = new String("abc");

System.out.println(s1.equals(s2));
```

a) false
b) true
c) Compilation error
d) Runtime error
Ans: b

---

52. Given:

```
StringBuilder sb = new StringBuilder("Java");

sb.insert(2, "123");

System.out.println(sb);
```

What is the output?
a) Java123
b) Ja123va
c) J123ava
d) Java
Ans: b

---

53. What will be the output?

```
StringBuffer sb = new StringBuffer("abc");

sb.delete(1, 3);

System.out.println(sb);
```

a) abc
b) a
c) ab
d) ac
Ans: b

---

54. Which of the following is **true** about `String.intern()` method?
a) Creates a deep copy
b) Allocates new memory
c) Returns a canonical representation from the string pool
d) Reverses the string
Ans: c

---

55. Which of the following is **false** regarding autoboxing?
    a) Converts primitive to wrapper
    b) Introduced in Java 5
    c) Converts object to primitive
    d) Is implicit
    Ans: c

---

56. How many objects are created in the following statement?

```
String s = new String("Java");
```

a) 0
b) 1
c) 2
d) Depends on JVM
Ans: c
(*One in heap, one in pool*)

---

57. Which method can be used to efficiently manipulate strings in a multi-threaded environment?
    a) String
    b) StringBuilder
    c) StringBuffer
    d) Arrays.toString()
    Ans: c

---

58. Given:

```
String s = null;

System.out.println(s + 10);
```

Output?
a) null10
b) 10null
c) Error

d) NullPointerException
Ans: a

---

59. What is the output?

```
char[] ch = {'J','A','V','A'};

String s = new String(ch);

System.out.println(s);
```

a) JAVA
b) JAV
c) Error
d) J,A,V,A
Ans: a

---

60. Which of these statements is valid about wrapper class caching?
    a) Values between -128 and 127 are cached for Integer
    b) All values are cached
    c) Only zero is cached
    d) Wrapper classes are not cached
    Ans: a

---

61. What will be the output?

```
String str1 = "Hello";

String str2 = "Hel" + "lo";

System.out.println(str1 == str2);
```

a) false
b) true
c) Compilation error
d) null

Ans: b
(*Compile-time constants are pooled*)

---

62. Choose the correct statement about `StringBuffer` and `StringBuilder`:
    a) Both are immutable
    b) Both are mutable, but StringBuffer is synchronized
    c) Both are synchronized
    d) StringBuilder is thread-safe
    Ans: b

---

63. Given:

```
String s = "abc";

System.out.println(s.substring(1, 3));
```

Output?
a) bc
b) ab
c) ac
d) Error
Ans: a

---

64. What happens when you do this?

```
Integer i1 = 100;

Integer i2 = 100;

System.out.println(i1 == i2);
```

a) true
b) false
c) Compilation error
d) Runtime error
Ans: a
(*Cached Integer values between -128 to 127*)

---

65. What is returned by `Integer.valueOf("100")`?
    a) 100
    b) "100"
    c) new Integer(100)
    d) Cached Integer object with value 100
    Ans: d

---

66. What happens if you call `charAt()` with index out of bounds?
    a) Returns null
    b) Compilation error
    c) Throws `StringIndexOutOfBoundsException`
    d) Prints last character
    Ans: c

---

67. What is printed?

```
StringBuilder sb = new StringBuilder("abc");

System.out.println(sb.capacity());
```

a) 0
b) 3
c) 16
d) 19
Ans: d
(*Default = 16 + length of input string*)

---

68. Which method of StringBuffer can be used to remove characters between two indices?
    a) erase()
    b) delete(start, end)
    c) cut()
    d) trim()
    Ans: b

69. Which method reverses the characters in a StringBuilder?
a) invert()
b) flip()
c) reverse()
d) reflect()
Ans: c

70. Which of the following operations creates new String objects?
a) append()
b) concat()
c) deleteCharAt()
d) insert()
Ans: b
(*String is immutable, so* `concat()` *creates new object*)

71. What is true about immutability of Strings?
a) Prevents memory leaks
b) Allows caching and pooling
c) Enables thread-safety
d) All of the above
Ans: d

72. Choose the correct output:

```
String s = "abc";

System.out.println(s.replace('a', 'z'));
```

a) zbc
b) abc
c) Error
d) azc
Ans: a

73. Which method converts a wrapper object to primitive explicitly?
   a) parseInt()
   b) unbox()
   c) intValue()
   d) toInt()
   Ans: c

---

74. What happens when calling `String.valueOf(null)`?
   a) "null"
   b) 0
   c) Error
   d) NullPointerException
   Ans: a

---

75. What is the role of `final` in the String class declaration?
   a) Prevents subclassing
   b) Improves performance
   c) Makes it mutable
   d) Allows overriding
   Ans: a

Awesome! Let's begin:

✅ Chapter 4: OOP Concepts

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Inheritance, Polymorphism, Overloading, Overriding, `this`, `super`, `final`, `static`, etc.

🟢 EASY LEVEL MCQs (1–25)

---

1. Which principle of OOP allows a class to use properties of another class?
   a) Encapsulation
   b) Inheritance
   c) Abstraction
   d) Polymorphism
   Ans: b

---

2. The keyword used to inherit a class in Java is:
   a) inherit
   b) base
   c) extends
   d) super
   Ans: c

---

3. Polymorphism means:
   a) Having many variables
   b) Having many methods
   c) One name, many forms
   d) Inheritance
   Ans: c

---

4. Which keyword is used to refer to the current object?
   a) self
   b) this
   c) super
   d) own
   Ans: b

---

5. Method overloading occurs when:
   a) Two methods have the same name but different parameters
   b) Two methods have same name and parameters
   c) One method calls another
   d) Methods are inherited
   Ans: a

---

6. Which of the following is used to call the parent class constructor?
   a) parent()
   b) this()
   c) base()
   d) super()
   Ans: d

---

7. Which class is always the superclass of all Java classes?
   a) Main
   b) Super
   c) Object
   d) Class
   Ans: c

---

8. What type of inheritance does Java **not support** with classes?
   a) Single
   b) Multilevel
   c) Multiple (via classes)
   d) Hierarchical
   Ans: c

---

9. A class declared with `final` keyword:
   a) Can be inherited
   b) Cannot be subclassed
   c) Can be overridden
   d) Is abstract
   Ans: b

---

10. Method overriding happens when:
   a) A subclass defines a method with the same signature as superclass
   b) Two methods have different names
   c) A class extends two parents
   d) Static methods are redefined
   Ans: a

---

11. Which modifier restricts method from being overridden?
    a) abstract
    b) static
    c) final
    d) private
    Ans: c

---

12. What is output of `System.out.println(this);` inside a method?
    a) Class name
    b) Object reference
    c) "this"
    d) Error
    Ans: b

---

13. Which keyword is used to define a static method?
    a) const
    b) static
    c) final
    d) this
    Ans: b

---

14. A method that belongs to the class and not to any object is:
    a) Instance method
    b) Static method
    c) Abstract method
    d) Overloaded method
    Ans: b

---

15. Which class type can't be instantiated directly?
    a) final
    b) static
    c) abstract
    d) subclass
    Ans: c

---

16. The main difference between overloading and overriding is:
  a) Overloading changes name
  b) Overriding changes parameters
  c) Overloading is compile-time, overriding is runtime
  d) Both occur at runtime
  Ans: c

---

17. Which keyword is used to stop inheritance in Java?
  a) static
  b) final
  c) super
  d) private
  Ans: b

---

18. Can constructors be overloaded in Java?
  a) No
  b) Yes
  c) Only in subclasses
  d) Only once
  Ans: b

---

19. Which method is inherited from `Object` class?
  a) equals()
  b) show()
  c) start()
  d) main()
  Ans: a

---

20. What does `super()` do?
  a) Calls the current class constructor
  b) Calls the superclass constructor
  c) Refers to this object
  d) Starts a new thread
  Ans: b

---

21. What is the result of trying to override a `final` method?
    a) Method is hidden
    b) Compilation error
    c) Method is duplicated
    d) Method is skipped
    Ans: b

---

22. Which feature of OOP improves code reusability?
    a) Abstraction
    b) Inheritance
    c) Encapsulation
    d) Polymorphism
    Ans: b

---

23. What type of polymorphism does method overriding represent?
    a) Static
    b) Dynamic
    c) Compile-time
    d) Overloading
    Ans: b

---

24. Can we overload the `main()` method in Java?
    a) No
    b) Yes
    c) Only in interfaces
    d) Only in abstract classes
    Ans: b

---

25. What is the return type of a constructor?
    a) void
    b) class name
    c) Object
    d) No return type
    Ans: d

Great! Let's continue with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 4: OOP Concepts

---

26. What will be the output of the following code?

```
class A {
    void show() {
        System.out.println("Class A");
    }
}
class B extends A {
    void show() {
        System.out.println("Class B");
    }
}
public class Test {
    public static void main(String[] args) {
        A obj = new B();
        obj.show();
    }
}
```

a) Class A
b) Class B
c) Compilation error

d) Runtime error
Ans: b

---

27. If a method is both `static` and `final`, what does it mean?
    a) It can be overridden
    b) It can be inherited but not overridden
    c) It is abstract
    d) It is private
    Ans: b

---

28. What will happen if a subclass has a method with the same signature as a `private` method of the superclass?
    a) It overrides the method
    b) It hides the method
    c) It is a compilation error
    d) It creates a new method in subclass
    Ans: d

---

29. What happens if `super()` is not called explicitly in a subclass constructor?
    a) Compiler adds `super()` automatically
    b) Constructor fails
    c) Object is not created
    d) Error occurs
    Ans: a

---

30. Which of the following statements is true regarding method overloading?
    a) Return type must be different
    b) Number or type of parameters must differ
    c) Both method name and parameters must differ
    d) Method name must change
    Ans: b

---

31. Given:

```
class Parent {

    static void greet() {

        System.out.println("Hello from Parent");

    }

}

class Child extends Parent {

    static void greet() {

        System.out.println("Hello from Child");

    }

}
```

Calling `Child.greet();` will output:
a) Hello from Parent
b) Hello from Child
c) Compilation error
d) Runtime error
Ans: b

---

32. Which concept binds data and methods into a single unit?
    a) Inheritance
    b) Abstraction
    c) Encapsulation
    d) Association
    Ans: c

---

33. Can a constructor be overridden?
    a) Yes
    b) No
    c) Only in abstract class
    d) Only in static class
    Ans: b

---

34. What happens when a child class object is assigned to a parent class reference?
   a) Compile-time error
   b) Only parent class methods can be called
   c) Only child class methods can be called
   d) Both can be accessed
   Ans: b

---

35. What is the term for using the same method name in multiple classes of the same hierarchy?
   a) Overloading
   b) Overriding
   c) Overcasting
   d) Abstraction
   Ans: b

---

36. What is the advantage of method overriding?
   a) Compile-time performance
   b) Run-time polymorphism
   c) Less memory usage
   d) Shorter syntax
   Ans: b

---

37. Which class can use `super` to call a superclass method?
   a) Abstract class
   b) Final class
   c) Subclass
   d) Object class
   Ans: c

---

38. What is the purpose of using `this()` in a constructor?
   a) Call a method
   b) Refer current object
   c) Call another constructor of the same class
   d) Refer to parent class
   Ans: c

---

39. Which statement is true for static methods in Java?
    a) Can override instance methods
    b) Can be overridden
    c) Belong to class, not instances
    d) Can access `this`
    Ans: c

---

40. When is a constructor called?
    a) During method execution
    b) At compile time
    c) When an object is created
    d) After finalizer
    Ans: c

---

41. Choose the correct syntax to prevent a method from being overridden:
    a) `private void method()`
    b) `static void method()`
    c) `final void method()`
    d) `abstract void method()`
    Ans: c

---

42. What happens if you try to override a static method?
    a) Method is hidden
    b) Method is overridden
    c) Compile-time error
    d) Runtime exception
    Ans: a

---

43. Which method from the `Object` class is often overridden in custom classes?
    a) clone()
    b) wait()
    c) notify()
    d) equals()

Ans: d

---

44. What will the following code print?

```
class A {
   int x = 10;
}
class B extends A {
   int x = 20;
   void print() {
      System.out.println(super.x);
   }
}
```

a) 10
b) 20
c) 0
d) Error
Ans: a

---

45. Which access modifier allows a subclass to override a method from a different package?
    a) private
    b) default
    c) protected
    d) static
    Ans: c

---

46. The `Object` class is part of which package?
    a) java.lang
    b) java.util

c) java.object
d) java.core
Ans: a

---

47. How many times can a constructor be called during object creation?
    a) 0
    b) 1
    c) Unlimited
    d) Only if it’s static
    Ans: b

---

48. Which type of method resolution does Java use for overridden methods?
    a) Static binding
    b) Early binding
    c) Dynamic binding
    d) Late compilation
    Ans: c

---

49. If a method in superclass is `protected`, what is the minimum access modifier you can use when overriding?
    a) private
    b) public
    c) default
    d) protected
    Ans: d

---

50. What is the primary goal of inheritance?
    a) Reduce memory
    b) Eliminate classes
    c) Code reuse
    d) Improve constructor chaining
    Ans: c

Excellent! Now, let’s finish Chapter 4 with the final:

🔴 HARD LEVEL MCQs (51–75)

Chapter 4: OOP Concepts

51. What will be the output of the following code?

```
class A {
    void display() {
        System.out.println("Class A");
    }
}
class B extends A {
    void display(int x) {
        System.out.println("Class B");
    }
}
public class Test {
    public static void main(String[] args) {
        B obj = new B();
        obj.display();
    }
}
```

a) Class B
b) Compilation error
c) Class A
d) Runtime error

Ans: c
(`display()` *from superclass is inherited and called; overloading, not overriding.*)

---

52. Which of the following conditions must be met for a method to be overridden?
    a) Method must be static
    b) Method must be final
    c) Method signature must be identical
    d) Method must be private
    Ans: c

---

53. What is the output?

```
class A {

   A() {

      System.out.println("A");

   }

}

class B extends A {

   B() {

      System.out.println("B");

   }

}

public class Test {

   public static void main(String[] args) {

      B obj = new B();

   }

}
```

a) A
b) B
c) AB
d) BA
Ans: c

---

54. Which of the following is true about `final` methods in Java?
    a) They can be overridden
    b) They are inherited but not overridden
    c) They can only be used inside interfaces
    d) They must be static
    Ans: b

---

55. Which of the following is not inherited by a subclass?
    a) public method
    b) protected variable
    c) constructor
    d) static method
    Ans: c

---

56. Which scenario will cause ambiguity in method overloading?
    a) Same method name with different parameter types
    b) Same method name with same number of parameters and same types
    c) Same method name with different return types
    d) Same method name with varying access modifiers
    Ans: b

---

57. Can a constructor call another constructor of the same class?
    a) No
    b) Yes, using `super()`
    c) Yes, using `this()`
    d) Only if it's static
    Ans: c

---

58. Which is true about polymorphism in Java?

a) It applies only to interfaces
b) It occurs only during compile time
c) Method overriding is an example of runtime polymorphism
d) Java does not support polymorphism
Ans: c

---

59. What happens if `super()` is called after `this()` in a constructor?

a) Constructor chaining works
b) Compilation error
c) Runs normally
d) Only `super()` executes
Ans: b
(*Either `super()` or `this()` must be first statement.*)

---

60. Can `final` variables be inherited?

a) No
b) Yes, but cannot be modified
c) Yes, and can be modified
d) Only in abstract classes
Ans: b

---

61. Which method provides object comparison based on content in overridden form?

a) toString()
b) clone()
c) equals()
d) hashCode()
Ans: c

---

62. Which keyword allows dynamic method dispatch?

a) static
b) new
c) super
d) override
Ans: d

63. What will be the result of the following?

```
class A {
  void show() {
    System.out.println("A");
  }
}
class B extends A {
  void show() {
    super.show();
    System.out.println("B");
  }
}
```

Calling `new B().show();` prints:
a) A
b) B
c) AB
d) BA
Ans: c

---

64. Which of these best represents method overriding?

```
class A {
  void greet() { }
}
class B extends A {
  void greet() { }
```

}

a) Overloading
b) Overriding
c) Hiding
d) None
Ans: b

---

65. Why can’t Java support multiple inheritance with classes?
    a) Compiler limitation
    b) JVM restriction
    c) To avoid ambiguity (Diamond problem)
    d) Because of garbage collection
    Ans: c

---

66. What is the result of executing the following?

```
class A {

   A() {

      System.out.println("A");

   }

}

class B extends A {

   B(int x) {

      super();

      System.out.println("B");

   }

}
```

Creating `new B(5);` outputs:
a) B

b) A B
c) B A
d) Compilation error
Ans: b

---

67. Which of the following cannot be used to achieve polymorphism in Java?
    a) Interfaces
    b) Method Overloading
    c) Method Overriding
    d) Constructor Overriding
    Ans: d

---

68. Which is the correct statement for method hiding?
    a) Overriding of static methods
    b) Overriding of final methods
    c) Inheriting private methods
    d) Abstract method implementation
    Ans: a

---

69. What happens if a subclass method has a broader access modifier than the superclass method being overridden?
    a) Compilation error
    b) Allowed
    c) Runtime error
    d) Method hiding
    Ans: b

---

70. What does the following do?

```
public final class A { }
```

a) A can be subclassed
b) A can’t be instantiated
c) A cannot be inherited
d) A must be abstract
Ans: c

---

71. What is printed?

```
class A {
   void display() { System.out.println("A"); }
}
class B extends A {
   void display() {
      System.out.println("B");
   }
}
class C extends B {
   void display() {
      System.out.println("C");
   }
}
A obj = new C();
obj.display();
```

a) A
b) B
c) C
d) Compilation error
Ans: c

---

72. Which of the following methods must be overridden from Object class to use objects in a `HashMap` properly?
    a) equals() and hashCode()
    b) toString() and equals()
    c) clone() and finalize()
    d) wait() and notify()

Ans: a

---

73. What happens if you call `this()` in a constructor but it's not the first line?
    a) Executes after other lines
    b) Skipped
    c) Compilation error
    d) Runtime exception
    Ans: c

---

74. Which allows different classes to respond to the same method call in different ways?
    a) Overloading
    b) Overriding
    c) Polymorphism
    d) Encapsulation
    Ans: c

---

75. What best explains the relationship between `this` and `super`?
    a) Both refer to the same object
    b) `this` refers to current class, `super` to parent
    c) `super` is a static reference
    d) Both are used only in constructors
    Ans: b

Perfect! Let's now dive into:

✅ Chapter 5: Exception Handling

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Try-catch-finally, `throw`, `throws`, custom exceptions, exception hierarchy, checked vs unchecked exceptions, etc.

🟢 EASY LEVEL MCQs (1–25)

---

1. What is an exception in Java?
   a) Syntax error
   b) Compilation error
   c) Runtime error that disrupts normal flow
   d) Memory leak
   Ans: c

---

2. Which keyword is used to handle exceptions in Java?
   a) catch
   b) throw
   c) try
   d) All of the above
   Ans: d

---

3. Which block must be used to catch exceptions?
   a) try
   b) catch
   c) finally
   d) throw
   Ans: b

---

4. Which of the following is a checked exception?
   a) ArithmeticException
   b) IOException
   c) NullPointerException
   d) ArrayIndexOutOfBoundsException
   Ans: b

---

5. What is the superclass of all exceptions?
   a) Object
   b) Throwable
   c) Error
   d) Exception

Ans: b

---

6. Which block is **always** executed whether an exception occurs or not?
   a) try
   b) catch
   c) finally
   d) throw
   Ans: c

---

7. What does the `throw` keyword do?
   a) Catches an exception
   b) Declares an exception
   c) Creates and throws an exception
   d) Returns from a method
   Ans: c

---

8. Which class is the parent of all runtime exceptions?
   a) Throwable
   b) Exception
   c) Error
   d) RuntimeException
   Ans: d

---

9. What happens if no catch block is found for an exception?
   a) Error is ignored
   b) Program continues normally
   c) JVM handles it and terminates the program
   d) Compiler fixes it
   Ans: c

---

10. What does `try` block contain?
   a) Code that might throw an exception
   b) Only catch blocks
   c) Only variable declarations

d) Only print statements
Ans: a

---

11. What is the output of the following?

```
try {

   int x = 5 / 0;

} catch (ArithmeticException e) {

   System.out.println("Error");

}
```

a) 0
b) Compilation error
c) Error
d) Runtime crash
Ans: c

---

12. Which of the following exceptions is thrown when array index is out of range?
    a) NullPointerException
    b) ArrayIndexOutOfBoundsException
    c) IOException
    d) ArithmeticException
    Ans: b

---

13. What will the following print?

```
try {

   System.out.println("A");

} finally {

   System.out.println("B");

}
```

a) A
b) AB
c) B
d) Compilation error
Ans: b

---

14. How can we define our own exceptions?
    a) By extending RuntimeException
    b) By implementing Throwable
    c) By using if-else
    d) Not possible
    Ans: a

---

15. What is the return type of `finally` block?
    a) int
    b) void
    c) It has no return type
    d) boolean
    Ans: c

---

16. Can we have multiple `catch` blocks?
    a) No
    b) Yes
    c) Only one
    d) Only if finally is used
    Ans: b

---

17. Which exception is thrown by `Integer.parseInt("abc")`?
    a) NullPointerException
    b) ClassCastException
    c) NumberFormatException
    d) IllegalArgumentException
    Ans: c

---

18. What is the purpose of `throws` keyword?
    a) Handle exception
    b) Define exception
    c) Declare exception
    d) Return exception
    Ans: c

---

19. Is `finally` block always executed?
    a) Yes
    b) Only if catch runs
    c) Only for runtime errors
    d) No
    Ans: a

---

20. What is the correct syntax to throw an exception manually?
    a) throw new Exception();
    b) throws Exception();
    c) new Exception();
    d) Exception throw();
    Ans: a

---

21. What is the type of `NullPointerException`?
    a) Checked
    b) Unchecked
    c) IO Exception
    d) Compilation error
    Ans: b

---

22. Can a `catch` block catch multiple exception types?
    a) No
    b) Yes, using `|`
    c) Yes, using `,`
    d) Only in JDK < 7
    Ans: b

---

23. Which class must be extended to create a checked exception?
    a) Throwable
    b) Exception
    c) RuntimeException
    d) Error
    Ans: b

---

24. Which keyword is used to declare a method that might throw an exception?
    a) throw
    b) try
    c) throws
    d) finally
    Ans: c

---

25. What is the output of this code?

```
try {

   return;

} finally {

   System.out.println("Finally block");

}
```

a) Nothing
b) Compilation error
c) Finally block
d) Runtime error
Ans: c

Great! Let’s move on to:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 5: Exception Handling

---

26. Which of the following statements is valid?

```
try {

   int x = 5 / 0;

} catch (Exception e) {

   System.out.println("Caught");

} catch (ArithmeticException e) {

   System.out.println("Arithmetic");

}
```

a) Caught
b) Arithmetic
c) Compilation error
d) Runtime error
Ans: c
(*More specific catch must come before general one.*)

---

27. What will happen if `System.exit(0)` is called inside a `try` block?

```
try {

    System.exit(0);

} finally {

    System.out.println("Finally");

}
```

a) Prints "Finally"
b) Terminates silently
c) Throws exception

d) Compilation error
Ans: b

---

28. Which of the following can **only be handled** at runtime and not checked at compile-time?
    a) FileNotFoundException
    b) ClassNotFoundException
    c) NullPointerException
    d) SQLException
    Ans: c

---

29. What is the purpose of having a `finally` block?
    a) Execute default code
    b) Handle fatal errors
    c) Ensure resource cleanup
    d) Handle null exceptions
    Ans: c

---

30. How do you create a custom checked exception?
    a) Extend `Throwable` directly
    b) Extend `RuntimeException`
    c) Extend `Exception`
    d) Use `throws` only
    Ans: c

---

31. What happens when a `try` block throws an exception and there is **no matching catch block**?
    a) Program continues
    b) JVM handles it
    c) Compilation error
    d) Program crashes
    Ans: b

---

32. Which of these is a valid multi-catch statement?

a) `catch (IOException, SQLException e)`

b) `catch (IOException | SQLException e)`

c) `catch (IOException; SQLException e)`

d) `catch IOException | SQLException e`

Ans: b

---

33. In which situation is the `finally` block **not** executed?

a) Exception thrown

b) Normal execution

c) JVM crashes

d) None

Ans: c

---

34. What is the output?

```
try {

   int x = 1 / 0;

} catch (ArithmeticException e) {

   throw e;

} finally {

   System.out.println("Cleanup");

}
```

a) Cleanup

b) Exception only

c) Cleanup followed by exception

d) Compile-time error

Ans: c

---

35. What is the correct way to declare a method that may throw two exceptions?

void myMethod() _______

a) throws IOException or SQLException
b) throw IOException, SQLException
c) throws IOException, SQLException
d) throws (IOException, SQLException)
Ans: c

---

36. Which of these exception types can be caught using a `catch (Exception e)` block?
    a) All exceptions
    b) Only checked exceptions
    c) Only unchecked exceptions
    d) Only RuntimeException
    Ans: a

---

37. Which exception is thrown when a thread is sleeping and gets interrupted?
    a) InterruptedException
    b) IllegalThreadStateException
    c) ThreadDeath
    d) RuntimeException
    Ans: a

---

38. Which block must **directly follow** a `try` block?
    a) finally
    b) catch or finally
    c) catch
    d) throw
    Ans: b

---

39. Is it possible to rethrow the same exception caught in a catch block?
    a) No
    b) Yes, using `throw e`
    c) Only for checked exceptions
    d) Only in Java 8

Ans: b

---

40. Which method is used to retrieve the exception message?
    a) e.printStackTrace()
    b) e.toString()
    c) e.getMessage()
    d) e.message()
    Ans: c

---

41. Which of the following exceptions is **not** a subclass of RuntimeException?
    a) ArrayIndexOutOfBoundsException
    b) NumberFormatException
    c) ClassCastException
    d) IOException
    Ans: d

---

42. Consider:

```
try {

   // code

} catch (IOException | SQLException e) {

   // handler

}
```

Which is true?
a) `e` must be final or effectively final
b) `e` can be reassigned
c) `IOException` must come before `SQLException`
d) `e` can be null
Ans: a

---

43. Which exception is thrown when casting an object of one type to an incompatible type?
  a) ClassCastException
  b) IllegalArgumentException
  c) UnsupportedOperationException
  d) TypeMismatchException
  Ans: a

---

44. Which of the following is **not true** about the `finally` block?
  a) It always executes
  b) It executes after `catch`
  c) It only executes if an exception occurs
  d) It's used for cleanup
  Ans: c

---

45. What happens when an exception is thrown inside `catch` block?
  a) Control passes to `finally`
  b) Program terminates
  c) Control skips `finally`
  d) Catch executes again
  Ans: a

---

46. How does Java differentiate between checked and unchecked exceptions?
  a) Based on whether they are subclasses of Exception
  b) Checked = compile-time, Unchecked = runtime
  c) Checked extends Exception but not RuntimeException
  d) All of the above
  Ans: d

---

47. How can you ensure that a resource (like file or DB connection) is closed, regardless of exceptions?
  a) Use finally
  b) Use try-with-resources
  c) Use both a and b
  d) Use catch block

Ans: c

---

48. Which of the following would you choose for a **custom unchecked** exception?
    a) Extends Exception
    b) Extends IOException
    c) Extends RuntimeException
    d) Implements Throwable
    Ans: c

---

49. What does `e.printStackTrace()` print?
    a) The cause only
    b) Line number only
    c) Complete exception hierarchy and trace
    d) Message only
    Ans: c

---

50. If a method is declared with `throws IOException`, what must the caller do?
    a) Ignore the exception
    b) Catch or declare it
    c) Declare only
    d) Catch only
    Ans: b

Awesome! Let’s now complete:

🔴 HARD LEVEL MCQs (51–75)

Chapter 5: Exception Handling

---

51. What happens in the following scenario?

try {

```
    throw new Exception("Try");

} catch (Exception e) {

    throw new Exception("Catch");

} finally {

    System.out.println("Finally");

}
```

a) Only "Finally" is printed
b) Exception with message "Try"
c) "Finally" is printed, then "Catch" is thrown
d) Compilation error
Ans: c

---

52. If an exception is thrown in `finally` block, and there was an exception already thrown in `try`, what happens?
    a) First exception is suppressed
    b) Both are printed
    c) Second exception replaces the first
    d) Compilation fails
    Ans: c

---

53. Which of the following is the **best practice** for custom exception classes?
    a) Extend `Object`
    b) Override `toString()` only
    c) Extend `Exception` or `RuntimeException` and define constructors
    d) Use `Throwable` directly
    Ans: c

---

54. Consider the following:

```
public class MyException extends Exception {

    MyException(String msg) {
        super(msg);
    }
}
```

How do you throw it?
a) `throw new MyException;`
b) `throw MyException("Error");`
c) `throw new MyException("Error");`
d) `throws MyException("Error");`
Ans: c

---

55. What is the output?

```
try {
    int a = 5 / 0;
} catch (ArithmeticException e) {
    System.out.println("AE");
} finally {
    System.out.println("Finally");
}
```

a) AE
b) AE Finally
c) Only Finally
d) Exception
Ans: b

---

56. Which exception will be thrown if you try to access a method on a `null` object?
    a) IllegalArgumentException
    b) NullPointerException
    c) ClassCastException
    d) IllegalAccessException

Ans: b

---

57. Which of the following exceptions is **checked**?
    a) NullPointerException
    b) IOException
    c) NumberFormatException
    d) ArithmeticException
    Ans: b

---

58. What is the difference between `throw` and `throws`?
    a) `throw` declares, `throws` creates
    b) `throws` is used to handle, `throw` to declare
    c) `throw` creates/throws object; `throws` declares exceptions in method signature
    d) No difference
    Ans: c

---

59. What will the code print?

```
try {

   System.out.println("A");

   return;

} finally {

   System.out.println("B");

}
```

a) A
b) AB
c) B
d) Compilation error
Ans: b

---

60. Which is true about try-with-resources in Java?
   a) Available since Java 6
   b) Can be used with any object
   c) Used for AutoCloseable objects
   d) Doesn't close resources automatically
   Ans: c

---

61. Which of the following ensures that exceptions are propagated to the caller?
   a) throw
   b) throws
   c) catch
   d) return
   Ans: b

---

62. Given:

```
try {
   FileReader fr = new FileReader("data.txt");
} catch (FileNotFoundException e) {
   System.out.println("File not found");
}
```

This requires handling because:
a) FileReader is unchecked
b) FileNotFoundException is checked
c) JVM checks all exceptions
d) Catch block must have finally
Ans: b

---

63. What is a **chained exception** in Java?
   a) Catching multiple exceptions
   b) Linking new exception with original cause
   c) Rethrowing exception using super
   d) Using multiple `finally` blocks

Ans: b

---

64. In try-with-resources, which interface must the resource implement?
    a) Closeable
    b) AutoCloseable
    c) Serializable
    d) Appendable
    Ans: b

---

65. What happens if no exception is thrown in the try block?
    a) Catch block runs
    b) Finally block runs
    c) Catch and finally both run
    d) None runs
    Ans: b

---

66. Which exception occurs when accessing a method of an incompatible class cast?
    a) ClassCastException
    b) IllegalStateException
    c) TypeMismatchException
    d) UnsupportedOperationException
    Ans: a

---

67. What is the signature of the `Throwable` constructor that allows chained exceptions?
    a) `Throwable(String)`
    b) `Throwable(String, Throwable)`
    c) `Throwable(Throwable)`
    d) Both b and c
    Ans: d

---

68. Which of these can be thrown but not caught in normal `try-catch` blocks?
    a) Exception
    b) Throwable
    c) Error

d) CheckedException
Ans: c

---

69. Which exception must **always** be caught or declared in a method?
a) RuntimeException
b) IOException
c) NullPointerException
d) ArithmeticException
Ans: b

---

70. What is the purpose of `addSuppressed()` method in Java exception handling?
a) Add multiple messages to one exception
b) Combine logs
c) Link secondary exceptions from try-with-resources
d) Replace stack trace
Ans: c

---

71. Which of the following does not belong to `java.lang` package?
a) Throwable
b) Exception
c) Error
d) IOException
Ans: d

---

72. Can you override a method that declares an exception with a broader exception?
a) Yes
b) No
c) Only if it's unchecked
d) Depends on compiler
Ans: b

---

73. Consider:

```
public void test() throws IOException {

    throw new IOException("Test IO");

}
```

How must a caller handle this?
a) Catch it or declare `throws IOException`
b) Ignore it
c) Use `return`
d) Use `final`
Ans: a

---

74. What happens when you override a method but throw a **narrower** exception?
    a) Compile-time error
    b) Allowed
    c) Ignored
    d) JVM handles it
    Ans: b

---

75. If a method in superclass throws `IOException`, what can subclass throw?
    a) IOException
    b) FileNotFoundException
    c) No exception
    d) All of the above
    Ans: d

Great! Let's move ahead to:

✅ Chapter 6: Java Collection Framework

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: List, Set, Map, Iterator, Enumeration, Collection interfaces, Generics, etc.

🟢 EASY LEVEL MCQs (1–25)

---

1. Which package contains Java Collection classes?
   a) java.lang
   b) java.io
   c) java.net
   d) java.util
   Ans: d

---

2. Which of the following is an interface in Java Collections?
   a) ArrayList
   b) HashSet
   c) List
   d) Hashtable
   Ans: c

---

3. Which collection allows duplicate elements?
   a) Set
   b) List
   c) Map
   d) TreeSet
   Ans: b

---

4. Which of the following is ordered?
   a) HashSet
   b) TreeSet
   c) ArrayList
   d) HashMap
   Ans: c

---

5. Which class implements the List interface?
   a) HashMap
   b) TreeSet
   c) ArrayList
   d) HashSet

Ans: c

---

6. Which method adds an element to a list?
   a) insert()
   b) put()
   c) add()
   d) set()
   Ans: c

---

7. Which class implements the Set interface?
   a) ArrayList
   b) HashSet
   c) HashMap
   d) Vector
   Ans: b

---

8. Which of the following is not part of the Collection interface hierarchy?
   a) List
   b) Set
   c) Map
   d) Queue
   Ans: c

---

9. Which collection does not allow duplicates?
   a) ArrayList
   b) LinkedList
   c) HashSet
   d) Vector
   Ans: c

---

10. Which collection is best suited for FIFO ordering?
    a) List
    b) Queue
    c) Set

d) Stack
Ans: b

---

11. The root interface of the collection framework is:
    a) List
    b) Collection
    c) Set
    d) Iterable
    Ans: d

---

12. Which of these allows key-value pairs?
    a) Set
    b) Map
    c) List
    d) Collection
    Ans: b

---

13. Which method removes all elements from a collection?
    a) removeAll()
    b) delete()
    c) clear()
    d) empty()
    Ans: c

---

14. Which class maintains insertion order and allows duplicates?
    a) TreeSet
    b) LinkedHashSet
    c) LinkedList
    d) HashSet
    Ans: c

---

15. Which of the following implements Map?
    a) HashMap
    b) TreeMap

c) LinkedHashMap
d) All of the above
Ans: d

---

16. Which class is synchronized by default?
a) ArrayList
b) Vector
c) HashSet
d) HashMap
Ans: b

---

17. Which method is used to get the size of a collection?
a) count()
b) size()
c) length()
d) getSize()
Ans: b

---

18. Which interface provides access to elements in forward direction only?
a) ListIterator
b) Iterator
c) Enumerator
d) Scanner
Ans: b

---

19. What is the default capacity of an `ArrayList`?
a) 5
b) 10
c) 16
d) 0
Ans: b

---

20. What happens if you insert duplicate keys in a `HashMap`?
a) Compile error

b) Runtime exception
c) Value is overwritten
d) Both values stored
Ans: c

---

21. Which method retrieves a value in a `Map`?
a) get()
b) find()
c) getValue()
d) search()
Ans: a

---

22. Which is not a valid implementation of List interface?
a) ArrayList
b) LinkedList
c) Vector
d) TreeSet
Ans: d

---

23. Which collection guarantees sorting in natural order?
a) ArrayList
b) HashMap
c) TreeSet
d) HashSet
Ans: c

---

24. Which method checks if a collection is empty?
a) isEmpty()
b) size() == 0
c) hasNext()
d) a and b
Ans: d

---

25. Which data structure uses LIFO order?
    a) Queue
    b) Deque
    c) Stack
    d) PriorityQueue
    Ans: c

Excellent! Let's proceed with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 6: Java Collection Framework

---

26. What is the output?

```
List<String> list = new ArrayList<>();

list.add("A");

list.add("B");

list.add(1, "C");

System.out.println(list);
```

a) [A, B, C]
b) [C, A, B]
c) [A, C, B]
d) Compilation error
Ans: c

---

27. Which of these is the correct syntax for creating a generic list of integers?
    a) List list = new ArrayList<>();
    b) List list = new ArrayList<>();
    c) List list = new ArrayList();
    d) ArrayList list = new ArrayList<>();

Ans: b

---

28. Which data structure should be used for constant-time performance on basic operations like `add()`, `remove()`, `contains()`?
    a) TreeSet
    b) ArrayList
    c) HashSet
    d) LinkedList
    Ans: c

---

29. Which collection class would you use for fast retrieval using unique keys?
    a) ArrayList
    b) HashMap
    c) TreeSet
    d) HashSet
    Ans: b

---

30. What is the time complexity of `get()` in `ArrayList`?
    a) O(1)
    b) O(log n)
    c) O(n)
    d) O(n log n)
    Ans: a

---

31. Which of the following statements about `HashSet` is true?
    a) Maintains insertion order
    b) Is sorted
    c) Allows duplicate elements
    d) Does not allow duplicates
    Ans: d

---

32. Which class provides a **fail-fast** iterator?
    a) Vector
    b) HashMap

c) ArrayList
d) Hashtable
Ans: c

---

33. Which interface supports **bidirectional iteration**?
a) Iterator
b) Iterable
c) ListIterator
d) Enumeration
Ans: c

---

34. What happens if you modify a collection while iterating using `Iterator`?
a) Exception is thrown
b) Value is ignored
c) Loop skips element
d) Allowed silently
Ans: a

---

35. What is true about generics in Java Collections?
a) They increase performance
b) They allow type safety
c) They are optional
d) Both b and c
Ans: d

---

36. What is the difference between `ArrayList` and `LinkedList`?
a) LinkedList allows duplicates, ArrayList doesn't
b) ArrayList is faster for random access
c) LinkedList is better for index-based access
d) ArrayList maintains a linked chain
Ans: b

---

37. Which of the following is **thread-safe** and part of legacy collections?
a) ArrayList

b) HashMap
c) Vector
d) HashSet
Ans: c

---

38. What does the `retainAll()` method do in collections?
a) Removes all elements
b) Adds new elements
c) Keeps only common elements
d) Clears duplicates
Ans: c

---

39. In `HashMap`, which method is used to iterate key-value pairs efficiently?
a) keys()
b) entrySet()
c) values()
d) keyList()
Ans: b

---

40. How does `TreeSet` maintain elements?
a) Hashing
b) Random order
c) Natural or comparator-based order
d) Insertion order
Ans: c

---

41. What happens if you try to add a duplicate key in `TreeMap`?
a) Replaces the old value
b) Allows both values
c) Throws exception
d) Ignores second value
Ans: a

---

42. What is the default ordering of elements in a `TreeSet`?
   a) Insertion
   b) Sorted ascending (natural)
   c) Random
   d) None
   Ans: b

---

43. How can you safely iterate and remove elements from a collection?
   a) Use index-based loop
   b) Use enhanced for loop
   c) Use Iterator and its `remove()` method
   d) Use `removeAll()`
   Ans: c

---

44. Which map maintains insertion order?
   a) HashMap
   b) TreeMap
   c) LinkedHashMap
   d) ConcurrentHashMap
   Ans: c

---

45. What is the output?

```
Map<String, Integer> map = new HashMap<>();

map.put("A", 1);

map.put("B", 2);

map.put("A", 3);

System.out.println(map.get("A"));
```

a) 1
b) 3
c) 2

d) null
Ans: b

---

46. Which of these allows duplicate keys?
    a) HashMap
    b) TreeMap
    c) Hashtable
    d) None of these
    Ans: d

---

47. What is returned by `containsKey()` in Map interface?
    a) Key
    b) Value
    c) boolean
    d) Object
    Ans: c

---

48. Which method is used to remove a specific mapping from a map?
    a) delete(key)
    b) remove(key)
    c) clear(key)
    d) discard(key)
    Ans: b

---

49. How can you make a collection **immutable**?
    a) Use `Collections.unmodifiableList()`
    b) Use final keyword
    c) Use private constructors
    d) You can't
    Ans: a

---

50. Which interface does not allow null keys or null values?
    a) HashMap
    b) TreeMap
    c) Hashtable

d) LinkedHashMap
Ans: c

Perfect! Let’s complete:

🔴 HARD LEVEL MCQs (51–75)

Chapter 6: Java Collection Framework

---

51. What is the result of iterating over a `HashMap` using `entrySet()` and modifying it within the loop?

```
for (Map.Entry<String, String> entry : map.entrySet()) {
   map.put("key", "value");
}
```

a) Works fine
b) Compiles but throws `ConcurrentModificationException`
c) Compilation error
d) Updates existing entries
Ans: b

---

52. Why does `HashSet` use `HashMap` internally?
a) To store duplicates
b) For insertion order
c) For constant-time performance
d) For type safety
Ans: c

---

53. Which of the following correctly describes the difference between `Hashtable` and `HashMap`?
a) Hashtable is non-synchronized

b) HashMap allows null keys and values
c) HashMap is slower
d) Hashtable supports generics
Ans: b

---

54. What is the time complexity of insertion in a `HashMap` in average case?
a) O(n)
b) O(log n)
c) O(1)
d) O(n log n)
Ans: c

---

55. Which of the following interfaces supports both key-value mapping and ordering by key?
a) HashMap
b) Hashtable
c) TreeMap
d) HashSet
Ans: c

---

56. Which method in the Collection interface is used to convert collection to an array?
a) array()
b) convert()
c) toArray()
d) toList()
Ans: c

---

57. Which of the following allows **null key** and **multiple null values**?
a) HashMap
b) TreeMap
c) Hashtable
d) ConcurrentHashMap
Ans: a

---

58. How does Java handle hash collisions in `HashMap`?
a) Uses another map
b) Overwrites entries
c) Uses a linked list or red-black tree (Java 8+)
d) Crashes the program
Ans: c

---

59. Which structure should be used for implementing a **priority queue**?
a) HashSet
b) TreeMap
c) PriorityQueue
d) LinkedList
Ans: c

---

60. What will the following code print?

```
List<Integer> list = new ArrayList<>();

list.add(1); list.add(2); list.add(3);

list.remove(1);

System.out.println(list);
```

a) [1, 3]
b) [2, 3]
c) [1, 2]
d) Error
Ans: a
(*Removes element at index 1*)

---

61. Which Map implementation is best suited for **concurrent access with thread safety**?
a) HashMap
b) TreeMap
c) LinkedHashMap
d) ConcurrentHashMap
Ans: d

---

62. Why is `EnumSet` more efficient than other sets when used with enums?
    a) Uses Hashing
    b) Backed by bit vectors
    c) Uses Trees
    d) Is mutable
    Ans: b

---

63. What happens if you try to store a custom object in `TreeSet` without implementing `Comparable` or providing a `Comparator`?
    a) Compiles and runs
    b) Runtime error
    c) ClassCastException
    d) No effect
    Ans: c

---

64. Which statement about `LinkedHashMap` is **false**?
    a) Maintains insertion order
    b) Allows null keys
    c) Is sorted by keys
    d) Is not thread-safe
    Ans: c

---

65. Which method in `Iterator` is used to avoid `ConcurrentModificationException` when removing elements?
    a) delete()
    b) clear()
    c) remove()
    d) exclude()
    Ans: c

---

66. Which is true about `ListIterator` but not `Iterator`?
    a) Only moves forward
    b) Allows element deletion
    c) Allows bidirectional traversal

d) Used only with Maps
Ans: c

---

67. What does this code return?

```
Set<String> set = new HashSet<>();

set.add("A");

set.add("B");

set.add("A");

System.out.println(set.size());
```

a) 3
b) 2
c) 1
d) 0
Ans: b

---

68. Which of the following methods returns a **view** of keys in a `Map`?
a) entrySet()
b) keySet()
c) getKeys()
d) values()
Ans: b

---

69. How can you sort a `Map` by values?
a) Use `TreeMap`
b) Use `Collections.sort()` directly
c) Convert to list and sort using a comparator
d) Sort during put
Ans: c

---

70. What does the `spliterator()` method in the Collection interface return?

a) Iterator
b) Stream
c) Spliterator
d) Comparator
Ans: c

---

71. When should you prefer `LinkedList` over `ArrayList`?

a) For frequent random access
b) For faster sorting
c) For frequent insertions/deletions
d) When thread-safety is needed
Ans: c

---

72. Which of the following is true about `ConcurrentHashMap`?

a) It allows null keys
b) It is fully synchronized
c) It uses segment-based locking
d) It allows duplicates
Ans: c

---

73. Which interface provides the foundation for queue-based classes?

a) List
b) Map
c) Queue
d) Iterator
Ans: c

---

74. Which method is used to insert an element at the beginning of a `LinkedList`?

a) addStart()
b) insertFirst()
c) addFirst()
d) prepend()
Ans: c

---

75. What’s the effect of calling `Collections.unmodifiableList(list)`?
a) Prevents additions only
b) Makes the list immutable
c) Clears the list
d) Sorts the list
Ans: b

Great! Let's now begin:

✅ Chapter 7: Functional Programming & Streams

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Lambda Expressions, Functional Interfaces, Method References, Streams, Terminal & Intermediate operations, etc.

🟢 EASY LEVEL MCQs (1–25)

---

1. Which Java version introduced Lambda expressions?
a) Java 6
b) Java 7
c) Java 8
d) Java 11
Ans: c

---

2. A lambda expression is used to implement which type of interface?
a) Normal
b) Functional
c) Abstract
d) Runnable only
Ans: b

---

3. A functional interface must have:
   a) Two abstract methods
   b) One abstract method
   c) No methods
   d) Static methods only
   Ans: b

---

4. Which annotation is used to define a functional interface?
   a) @Lambda
   b) @Override
   c) @Function
   d) @FunctionalInterface
   Ans: d

---

5. Which of the following is a built-in functional interface in Java?
   a) Predicate
   b) Consumer
   c) Supplier
   d) All of the above
   Ans: d

---

6. What is the return type of `Supplier<T>`?
   a) T
   b) boolean
   c) void
   d) int
   Ans: a

---

7. Which functional interface represents a function with one input and one output?
   a) Predicate
   b) Function
   c) Consumer
   d) Runnable
   Ans: b

---

8. Which interface is used for filtering in streams?
   a) Consumer
   b) Predicate
   c) Supplier
   d) Runnable
   Ans: b

---

9. What does a stream represent?
   a) A file
   b) A sequence of elements supporting sequential or parallel operations
   c) A database
   d) A thread
   Ans: b

---

10. What is the output type of `filter()` method in streams?
   a) List
   b) Collection
   c) Stream
   d) Optional
   Ans: c

---

11. What type of method is `forEach()` in streams?
   a) Intermediate
   b) Terminal
   c) Static
   d) Constructor
   Ans: b

---

12. Which interface does `Runnable` implement?
   a) Supplier
   b) Consumer
   c) Functional interface
   d) Predicate
   Ans: c

---

13. What is the return type of `Predicate<T>.test(T t)`?
    a) int
    b) boolean
    c) T
    d) void
    Ans: b

---

14. What does the `map()` method in streams do?
    a) Filters values
    b) Converts elements
    c) Combines elements
    d) Removes elements
    Ans: b

---

15. Lambda expressions are a replacement for:
    a) Anonymous classes
    b) Constructors
    c) Static blocks
    d) Packages
    Ans: a

---

16. Which of these is a valid lambda expression?
    a) x => x + 1
    b) (x) -> x + 1
    c) function(x) { return x + 1 }
    d) lambda(x) x + 1
    Ans: b

---

17. What does `Optional.ofNullable(value)` return if value is null?
    a) null
    b) Exception
    c) Empty Optional
    d) Compilation error
    Ans: c

---

18. The `collect()` method is:
    a) Used to consume values
    b) A terminal operation
    c) Filters values
    d) An intermediate operation
    Ans: b

---

19. What does `peek()` do in streams?
    a) Filters values
    b) Performs action without consuming
    c) Returns the count
    d) Ends the stream
    Ans: b

---

20. What does `distinct()` in streams return?
    a) Stream with duplicate values
    b) Stream with only unique elements
    c) Null values only
    d) Nothing
    Ans: b

---

21. What is the purpose of `reduce()`?
    a) To multiply all values
    b) To return the count
    c) To accumulate values into one result
    d) To remove duplicates
    Ans: c

---

22. What is the output of: `Stream.of(1, 2, 3).count();`
    a) 0
    b) 1
    c) 3
    d) Error
    Ans: c

---

23. Can streams be reused once operated upon?
    a) Yes
    b) Only if parallel
    c) No
    d) Only for primitive types
    Ans: c

---

24. `Optional.empty()` returns:
    a) Null
    b) Exception
    c) An empty Optional
    d) Undefined
    Ans: c

---

25. Which of the following is a terminal operation in Java streams?
    a) map()
    b) filter()
    c) collect()
    d) peek()
    Ans: c

Great! Let's proceed with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 7: Functional Programming & Streams

---

26. What will the following code output?

```
Stream<String> stream = Stream.of("java", "python", "java");

long count = stream.distinct().count();
```

System.out.println(count);

a) 3
b) 2
c) 1
d) Compilation error
Ans: b

---

27. What does the `Optional.ifPresent(Consumer<? super T> action)` method do?
    a) Always executes the action
    b) Executes only if value is present
    c) Throws an exception if empty
    d) Returns boolean
    Ans: b

---

28. What is the result of calling `findFirst()` on an empty stream?
    a) Null
    b) Empty Optional
    c) Exception
    d) Zero
    Ans: b

---

29. Which of the following stream operations is **lazy**?
    a) forEach()
    b) collect()
    c) filter()
    d) count()
    Ans: c

---

30. In streams, `mapToInt()` returns:
    a) List
    b) IntStream
    c) Map
    d) OptionalInt

Ans: b

---

31. Which operation in streams is best suited for chaining transformation functions?
    a) collect()
    b) peek()
    c) map()
    d) reduce()
    Ans: c

---

32. Which of the following can **not** be passed as a lambda target?
    a) Runnable
    b) Comparator
    c) Thread
    d) Predicate
    Ans: c

---

33. What is the purpose of method reference `ClassName::staticMethod`?
    a) Calls a constructor
    b) Refers to a non-static method
    c) Binds to a static method
    d) Overrides methods
    Ans: c

---

34. Which statement is **true** about `Function<T, R>`?
    a) Accepts two arguments
    b) Has `apply()` method
    c) Returns boolean
    d) Used only in sorting
    Ans: b

---

35. Which of the following is the **correct way** to define a `Predicate<String>` that checks string length > 5?
    a) `Predicate<String> p = s -> s.length() > 5;`

b) `Predicate<String> p = String -> String.length > 5;`
c) `Predicate<String> p = (s) -> { return s > 5; };`
d) `Predicate<String> p = s.length > 5;`
Ans: a

---

36. What will this stream operation do?

Stream.of(1, 2, 3, 4).filter(i -> i % 2 == 0).map(i -> i * 10).forEach(System.out::print);

a) 20 40
b) 10 30
c) 1 2 3 4
d) 2 4
Ans: a

---

37. Which of these is **not** a built-in functional interface?
a) BiConsumer
b) Function
c) Stream
d) Supplier
Ans: c

---

38. What is the correct way to convert a list to a stream and get the max element?

List<Integer> list = Arrays.asList(3, 1, 4);

Optional<Integer> max = ________

a) `list.stream().max()`
b) `list.toStream().max()`
c) `list.stream().max(Comparator.naturalOrder())`
d) `max(list)`
Ans: c

---

39. Which of the following is a **primitive specialization** of `Stream`?
   a) Stream
   b) ObjectStream
   c) IntStream
   d) GenericStream
   Ans: c

---

40. What is true about `flatMap()` in streams?
   a) It transforms one element into multiple streams
   b) It maps elements into optional values
   c) It filters based on condition
   d) It is terminal
   Ans: a

---

41. Which interface is used to define custom sorting in streams?
   a) Filter
   b) Consumer
   c) Comparator
   d) Function
   Ans: c

---

42. Which of the following terminal operations returns a single summary result from a stream?
   a) forEach()
   b) reduce()
   c) map()
   d) filter()
   Ans: b

---

43. What is the return type of `reduce(T identity, BinaryOperator<T>)`?
   a) Stream
   b) Optional
   c) T
   d) void
   Ans: c

---

44. What will `anyMatch()` return if the condition is false for all elements?
    a) false
    b) true
    c) null
    d) Optional.empty
    Ans: a

---

45. What does this code print?

```
Stream.of("a", "b", "c").limit(2).forEach(System.out::print);
```

a) abc
b) ab
c) bc
d) a
Ans: b

---

46. What is the effect of this?

```
Stream.empty().forEach(System.out::println);
```

a) Prints null
b) Prints nothing
c) Throws exception
d) Prints 0
Ans: b

---

47. How can we convert a stream to a list?

```
Stream<String> s = Stream.of("a", "b");

List<String> list = ___________;
```

a) `s.collect(Collectors.toList())`
b) `s.toList()`
c) `List.of(s)`
d) `Collectors.list(s)`
Ans: a

---

48. What is the output type of `groupingBy()` collector?
    a) List
    b) Set
    c) Map
    d) Stream
    Ans: c

---

49. Which of these functional interfaces accepts two arguments and returns a result?
    a) Function
    b) BiFunction
    c) Predicate
    d) Supplier
    Ans: b

---

50. What will this code print?

```
Optional<String> opt = Optional.of("Hello");

System.out.println(opt.orElse("World"));
```

a) Hello
b) World
c) null
d) Optional
Ans: a

Awesome! Let’s wrap up:

🔴 HARD LEVEL MCQs (51–75)

Chapter 7: Functional Programming & Streams

---

51. What will be the output of the following?

```
List<String> list = Arrays.asList("abc", "def", "ghi");

String result = list.stream().reduce("", (a, b) -> a + b);

System.out.println(result);
```

a) abcdefghi
b) abc def ghi
c) ""
d) Compilation error
Ans: a

---

52. What is the purpose of `flatMap()` in Java Streams?
    a) Replaces each value with its mapped equivalent
    b) Maps each element to multiple elements and flattens
    c) Collects results
    d) Filters duplicates
    Ans: b

---

53. What is the key difference between `map()` and `flatMap()`?
    a) `map()` works only on primitives
    b) `flatMap()` flattens nested streams
    c) `map()` returns Optional
    d) Both are terminal
    Ans: b

---

54. Which of the following lambdas can be used as a `BiFunction<String, Integer, String>`?
    a) (s, i) -> s.length()

b) (s, i) -> s + i
c) (s) -> s.toUpperCase()
d) (s, i) -> i
Ans: b

---

55. What will the following return?

```
Stream.of(2, 4, 6).anyMatch(n -> n % 2 != 0);
```

a) true
b) false
c) 0
d) Exception
Ans: b

---

56. What is the purpose of `Optional.orElseGet(Supplier<? extends T>)`?
a) Always returns default value
b) Lazily returns default if value is absent
c) Immediately invokes the supplier
d) Never used
Ans: b

---

57. What does the following code output?

```
Optional<String> o = Optional.ofNullable(null);

System.out.println(o.isPresent());
```

a) true
b) false
c) null
d) Exception
Ans: b

---

58. Which of the following is true about `Collectors.toMap()`?
   a) It allows duplicate keys
   b) It throws exception on key collision
   c) It returns a TreeMap
   d) It ignores nulls by default
   Ans: b

---

59. What happens if you call `get()` on an empty Optional?
   a) null is returned
   b) Exception is thrown
   c) false is returned
   d) Default value
   Ans: b
   (*Throws `NoSuchElementException`*)

---

60. Which of these methods is used to provide an alternative Optional when the original is empty?
   a) orElse()
   b) orElseGet()
   c) orElseThrow()
   d) or()
   Ans: d

---

61. Which collector counts the number of elements?
   a) toList()
   b) groupingBy()
   c) counting()
   d) summarizingInt()
   Ans: c

---

62. Which statement is **true** about stream **parallelism**?
   a) Streams are sequential by default
   b) Streams run in parallel unless specified
   c) All collectors are thread-safe
   d) Only filter() is parallel
   Ans: a

---

63. What is returned by `Collectors.joining(", ")`?
    a) Set
    b) List
    c) String
    d) Stream
    Ans: c

---

64. What happens if a lambda throws a checked exception?
    a) Compiles normally
    b) Must be handled or declared
    c) Automatically handled
    d) Skipped silently
    Ans: b

---

65. Which lambda matches `Supplier<Double>`?
    a) () -> 10
    b) () -> 10.0
    c) (x) -> x
    d) x -> Math.random()
    Ans: b

---

66. How do you refer to an instance method of an arbitrary object?
    a) ClassName::methodName
    b) this::method
    c) object::staticMethod
    d) () -> method()
    Ans: a

---

67. What is the output?

```
Optional<String> o = Optional.of("test");

System.out.println(o.map(String::toUpperCase).get());
```

a) TEST
b) test
c) null
d) Optional
Ans: a

---

68. Which of the following is a **pure function** in Java?
    a) Random number generator
    b) Logging function
    c) String::toUpperCase
    d) Thread.sleep()
    Ans: c

---

69. What happens if you reuse a stream after a terminal operation?

```
Stream<String> s = Stream.of("a", "b");

s.forEach(System.out::println);

s.forEach(System.out::println);
```

a) Prints a b twice
b) Compiles and runs
c) Throws IllegalStateException
d) Ignores second call
Ans: c

---

70. What will this return?

```
Stream.of(1, 2, 3).mapToInt(i -> i).sum();
```

a) 6
b) 3
c) 1

d) Error
Ans: a

---

71. Which is **not** a characteristic of functional programming?
    a) Pure functions
    b) Immutability
    c) Shared mutable state
    d) Stateless computations
    Ans: c

---

72. Which collector would you use to group stream elements by a key?
    a) groupingBy()
    b) toSet()
    c) counting()
    d) joining()
    Ans: a

---

73. What is the output?

```
Stream.of(1, 2, 3, 4).skip(2).forEach(System.out::print);
```

a) 1234
b) 12
c) 34
d) 234
Ans: c

---

74. Which of the following creates an infinite stream?
    a) Stream.of()
    b) Stream.generate()
    c) Stream.empty()
    d) Stream.builder()
    Ans: b

---

75. What is the result of:

```
Stream.of("a", "b", "c").limit(0).collect(Collectors.toList()).size();
```

a) 3
b) 0
c) 1
d) Exception
Ans: b

Great choice! Let's begin:

✅ Chapter 8: Multithreading

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Thread creation, lifecycle, synchronization, inter-thread communication, `Runnable`, `Thread`, `wait()`, `notify()`, `sleep()`, etc.

🟢 EASY LEVEL MCQs (1–25)

---

1. Which package contains the `Thread` class?
   a) java.util
   b) java.lang
   c) java.io
   d) java.thread
   Ans: b

---

2. Which interface is used to define a thread in Java?
   a) Runnable
   b) Threadable
   c) Callable
   d) Executable

Ans: a

---

3. Which class can be used to create a thread in Java?
   a) Thread
   b) Runnable
   c) Object
   d) Timer
   Ans: a

---

4. What is the method used to start a thread?
   a) run()
   b) execute()
   c) start()
   d) call()
   Ans: c

---

5. What happens if `run()` is called instead of `start()` on a thread object?
   a) Thread runs concurrently
   b) Thread runs sequentially
   c) Compile-time error
   d) JVM crashes
   Ans: b

---

6. Which method is inherited by threads to define the task?
   a) execute()
   b) task()
   c) run()
   d) do()
   Ans: c

---

7. Which thread method causes the currently executing thread to pause for a specified time?
   a) wait()
   b) pause()

c) sleep()
d) yield()
Ans: c

---

8. What does `Thread.yield()` do?
a) Kills the thread
b) Sends thread to sleep
c) Gives chance to other threads of same priority
d) Restarts the thread
Ans: c

---

9. What is the default priority of a thread in Java?
a) 1
b) 5
c) 10
d) 0
Ans: b

---

10. Which method returns a reference to the currently executing thread?
a) getCurrent()
b) Thread.get()
c) Thread.currentThread()
d) runThread()
Ans: c

---

11. Threads in Java are:
a) Processes
b) Lightweight processes
c) Abstract classes
d) Events
Ans: b

---

12. Which of the following methods can stop a thread?
a) suspend()

b) stop()
c) interrupt()
d) All of the above
Ans: d

---

13. Which method forces one thread to wait for another to complete?
    a) wait()
    b) join()
    c) notify()
    d) sleep()
    Ans: b

---

14. What will happen if `sleep()` is called inside a thread?
    a) Terminates the thread
    b) Suspends it temporarily
    c) Pauses the JVM
    d) Causes an exception
    Ans: b

---

15. What is the return type of `isAlive()` method?
    a) boolean
    b) void
    c) int
    d) Thread
    Ans: a

---

16. What happens if you call `start()` on a thread that has already completed?
    a) Restarts the thread
    b) Throws IllegalThreadStateException
    c) Runs the thread again
    d) Silently fails
    Ans: b

---

17. The `run()` method of a thread is:
   a) Called automatically by JVM
   b) Called by the programmer
   c) Never used
   d) Used to terminate a thread
   Ans: a (when `start()` is used)

---

18. How many times can a thread be started?
   a) Once
   b) Twice
   c) Infinite
   d) None
   Ans: a

---

19. Which method is used to check if a thread is still running?
   a) isRunning()
   b) isAlive()
   c) getState()
   d) running()
   Ans: b

---

20. Which thread method releases the lock but keeps thread alive?
   a) wait()
   b) notify()
   c) sleep()
   d) yield()
   Ans: a

---

21. What is the parent class of all threads?
   a) ThreadGroup
   b) Object
   c) Runnable
   d) Thread
   Ans: d

---

22. Which keyword is used to prevent thread interference?
    a) final
    b) private
    c) synchronized
    d) static
    Ans: c

---

23. Which method wakes up a waiting thread?
    a) sleep()
    b) wake()
    c) notify()
    d) interrupt()
    Ans: c

---

24. Which of these methods is not recommended for controlling threads?
    a) stop()
    b) suspend()
    c) resume()
    d) start()
    Ans: a (deprecated)

---

25. How do you ensure only one thread accesses a method at a time?
    a) static
    b) volatile
    c) synchronized
    d) private
    Ans: c

Awesome! Let's continue with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 8: Multithreading

---

26. What is printed by the following code?

```
Thread t = new Thread(() -> System.out.println(Thread.currentThread().getName()));

t.start();
```

a) Thread-0
b) main
c) Runnable
d) Compilation error
Ans: a

---

27. Which of the following methods will **not** throw `InterruptedException`?
   a) sleep()
   b) join()
   c) wait()
   d) yield()
   Ans: d

---

28. What is the result of calling `wait()` without acquiring the object's monitor?
   a) Program sleeps
   b) Program waits
   c) IllegalMonitorStateException
   d) Compilation error
   Ans: c

---

29. Which statement is true about thread priorities?
   a) Thread with lower priority always runs first
   b) Priority affects execution order but is not guaranteed
   c) JVM ignores priorities
   d) Priority is used only on Linux
   Ans: b

---

30. What is the effect of calling `interrupt()` on a sleeping thread?
   a) It wakes up the thread with an exception

b) Silently skips
c) Terminates the thread
d) Stops immediately
Ans: a

---

31. What is the output order of this code?

```
Thread t1 = new Thread(() -> System.out.print("A"));

Thread t2 = new Thread(() -> System.out.print("B"));

t1.start();

t2.start();
```

a) Always AB
b) Always BA
c) A or B first – order is not guaranteed
d) Compilation error
Ans: c

---

32. What does the following code do?

```
synchronized (this) {

   // critical section

}
```

a) Locks object forever
b) Locks the object’s class
c) Acquires lock on current instance
d) Acquires lock on parent class
Ans: c

---

33. Which method is used to **resume** a suspended thread?
a) start()
b) resume()

c) notify()
d) run()
Ans: b *(deprecated)*

---

34. What does `Thread.setDaemon(true)` do?
    a) Makes thread non-blocking
    b) Makes thread background (doesn't prevent JVM from exiting)
    c) Prevents thread from terminating
    d) Makes thread high-priority
    Ans: b

---

35. When is a thread considered dead?
    a) After start()
    b) After sleep()
    c) After run() completes
    d) After yield()
    Ans: c

---

36. What happens if multiple threads access a non-synchronized method?
    a) Only one runs
    b) Thread-safe by default
    c) Thread interference is possible
    d) JVM handles order
    Ans: c

---

37. What does this code do?

```
Thread t = new Thread(() -> {

   synchronized(Thread.class) {

      System.out.println("Locked by: " + Thread.currentThread().getName());

   }

});
```

```
t.start();
```

a) Locks current object
b) Locks the Thread class object
c) Throws exception
d) Runs asynchronously
Ans: b

---

38. What does `volatile` keyword ensure?
    a) Faster performance
    b) Variable is thread-safe
    c) Updates to variable are visible to all threads
    d) No need for synchronization
    Ans: c

---

39. What is the role of the `join()` method?
    a) Kills current thread
    b) Waits for another thread to die
    c) Executes two threads simultaneously
    d) Joins multiple thread outputs
    Ans: b

---

40. What happens if a thread calls `join()` on itself?
    a) Waits forever
    b) Returns instantly
    c) Compilation error
    d) Terminates the thread
    Ans: a

---

41. What does the following code produce?

```
Thread t = new Thread();

System.out.println(t.getState());
```

a) RUNNABLE
b) TERMINATED
c) NEW
d) WAITING
Ans: c

---

42. Which statement is true about `synchronized static` methods?
    a) Lock is on object
    b) Lock is on class object
    c) Cannot be overridden
    d) Lock is on JVM
    Ans: b

---

43. When two threads call `wait()` on the same object, what must happen to resume them?
    a) call wait() again
    b) call notify() or notifyAll() on the same object
    c) call sleep()
    d) call run()
    Ans: b

---

44. Which of the following allows safe data exchange between threads?
    a) HashMap
    b) Vector
    c) ConcurrentHashMap
    d) TreeMap
    Ans: c

---

45. What will be the result of calling `notify()` without `wait()`?
    a) Nothing happens
    b) Thread is blocked
    c) Compilation error
    d) Runtime error
    Ans: a

46. Which of the following allows interruption of blocking threads?
   a) interrupt()
   b) notify()
   c) resume()
   d) reset()
   Ans: a

---

47. What happens if two threads call a synchronized method at the same time on **different objects**?
   a) One blocks, one runs
   b) Both wait
   c) Both can execute
   d) JVM crashes
   Ans: c

---

48. What does `Thread.sleep(0)` do?
   a) Makes thread sleep forever
   b) Causes runtime error
   c) Yields control to another thread
   d) No effect
   Ans: c

---

49. Which method checks if a thread is interrupted?
   a) isRunning()
   b) isInterrupted()
   c) hasInterrupted()
   d) checkInterrupt()
   Ans: b

---

50. What's the correct way to make a method thread-safe?
   a) Use static
   b) Use private variables
   c) Use `synchronized` keyword
   d) Avoid objects

Ans: c

Excellent! Let's finish:

🔴 HARD LEVEL MCQs (51–75)

Chapter 8: Multithreading

---

51. What will happen in this scenario?

```
Thread t = new Thread(() -> {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        System.out.println("Interrupted");
    }
});
t.start();
t.interrupt();
```

a) Always prints "Interrupted"
b) Never prints
c) May or may not print "Interrupted" depending on timing
d) Compilation error
Ans: c

---

52. Which method is **safest** to use for stopping a thread in modern Java?
    a) stop()
    b) destroy()
    c) Use a volatile flag and exit loop

d) finalize()
Ans: c

---

53. What happens if you call `notify()` on an object with **multiple waiting threads**?
    a) All are notified
    b) Only one is notified
    c) All are blocked
    d) None are resumed
    Ans: b

---

54. What is the role of `Thread.UncaughtExceptionHandler`?
    a) Terminates thread forcefully
    b) Handles uncaught exceptions in a thread
    c) Logs exceptions
    d) Converts checked to unchecked exception
    Ans: b

---

55. Which of the following scenarios can cause a **deadlock**?
    a) Threads never synchronize
    b) Threads run sequentially
    c) Threads wait for each other’s locks
    d) Threads yield
    Ans: c

---

56. How do you avoid **livelock** in multithreading?
    a) Use synchronized blocks everywhere
    b) Never release a lock
    c) Introduce delay or backoff strategies
    d) Avoid using multiple threads
    Ans: c

---

57. What does this code output?

```
Thread t = new Thread(() -> {});

t.start();

t.join();

System.out.println(t.isAlive());
```

a) true
b) false
c) null
d) Compilation error
Ans: b

---

58. Why is `volatile` not a complete replacement for `synchronized`?
    a) It does not affect visibility
    b) It is slower
    c) It cannot ensure atomicity
    d) It affects compilation only
    Ans: c

---

59. What is **atomicity** in threading context?
    a) Memory leak control
    b) Operations completed in a single step
    c) Exception handling
    d) Thread delay
    Ans: b

---

60. Which thread state describes a thread that's **blocked waiting for a monitor lock**?
    a) WAITING
    b) SLEEPING
    c) BLOCKED
    d) TIMED_WAITING
    Ans: c

---

61. Which tool is used to detect deadlocks in a Java program?
    a) javac

b) jps
c) jstack
d) jconsole
Ans: c

---

62. What does `ThreadGroup` provide in Java?
a) Execution control
b) Logical grouping of threads
c) Memory management
d) Stream handling
Ans: b

---

63. What will happen if two threads are trying to **synchronize on different objects** inside the same method?
a) Deadlock
b) They wait for each other
c) No blocking; both execute
d) JVM crash
Ans: c

---

64. What is **false sharing** in multithreading?
a) Threads sharing variables intentionally
b) Threads accessing adjacent memory causing performance issues
c) Synchronization on null
d) Threads modifying constant variables
Ans: b

---

65. What does the following cause?

```
synchronized (lock1) {

   synchronized (lock2) {

      // critical section

   }
```

}

a) Deadlock always
b) Fine if lock order is consistent
c) Race condition
d) Exception
Ans: b

---

66. What does `ForkJoinPool` primarily optimize?
    a) Blocking I/O
    b) Divide-and-conquer parallelism
    c) Thread scheduling
    d) UI rendering
    Ans: b

---

67. In Java, which method **can be overridden** to customize thread behavior?
    a) start()
    b) run()
    c) join()
    d) sleep()
    Ans: b

---

68. Which class provides **reentrant locks** in Java?
    a) LockObject
    b) AtomicLock
    c) ReentrantLock
    d) LockSync
    Ans: c

---

69. What is the benefit of `ReentrantLock` over `synchronized`?
    a) More readable
    b) Supports tryLock(), fairness, and interruptible lock
    c) Always faster
    d) Does not require unlocking
    Ans: b

---

70. Which class is **thread-safe** and allows **blocking retrieval** of elements?
    a) ArrayList
    b) HashMap
    c) LinkedBlockingQueue
    d) Stack
    Ans: c

---

71. Which method allows trying to acquire a lock without blocking?
    a) tryLock()
    b) getLock()
    c) checkLock()
    d) attemptLock()
    Ans: a

---

72. What does `CyclicBarrier` do?
    a) Coordinates threads to meet at a common point
    b) Prevents thread creation
    c) Locks shared variables
    d) Waits until one thread finishes
    Ans: a

---

73. Which of the following is true about `ThreadLocal`?
    a) All threads share the same value
    b) Each thread has its own isolated copy
    c) Used for locking
    d) Avoids thread interference using volatile
    Ans: b

---

74. Which method is used in `Executors` to create a fixed thread pool?
    a) newThreadPool(int)
    b) newFixedPool(int)
    c) newFixedThreadPool(int)
    d) newExecutor()
    Ans: c

---

75. What is a **race condition**?
    a) When threads run in sequence
    b) When threads wait too long
    c) When multiple threads access shared data without synchronization
    d) When thread is terminated before start
    Ans: c

Perfect! Let's move forward to:

✅ Chapter 9: File I/O and Serialization

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: File handling (`File`, `FileReader`, `BufferedReader`, `FileWriter`, etc.), Serialization (`Serializable`, `transient`, `ObjectInputStream`, etc.)

🟢 EASY LEVEL MCQs (1–25)

---

1. Which package is primarily used for file handling in Java?
   a) java.util
   b) java.io
   c) java.net
   d) java.file
   Ans: b

---

2. Which class is used to represent file and directory paths?
   a) FileSystem
   b) Path
   c) File
   d) Reader
   Ans: c

---

3. What is the correct way to create a file in Java?
   a) `File f = File("file.txt");`
   b) `File f = new File("file.txt");`
   c) `file = create("file.txt");`
   d) `File.make("file.txt");`
   Ans: b

---

4. What method checks whether a file exists?
   a) exists()
   b) isAvailable()
   c) available()
   d) hasFile()
   Ans: a

---

5. Which method creates a new file in the filesystem?
   a) makeFile()
   b) createFile()
   c) newFile()
   d) createNewFile()
   Ans: d

---

6. What does the `delete()` method of File class do?
   a) Deletes contents of a file
   b) Deletes the file or directory
   c) Truncates a file
   d) Marks file for deletion
   Ans: b

---

7. What does `isDirectory()` return for a directory?
   a) true
   b) false
   c) null
   d) Error
   Ans: a

8. Which stream class is used to write characters to a file?
   a) FileOutputStream
   b) FileWriter
   c) ObjectOutputStream
   d) PrintStream
   Ans: b

---

9. Which stream is used to **read characters** from a file?
   a) FileInputStream
   b) BufferedInputStream
   c) FileReader
   d) FileChannel
   Ans: c

---

10. What is the purpose of `BufferedReader`?
    a) Writes text to file
    b) Reads raw bytes
    c) Reads text with buffering
    d) Compresses file
    Ans: c

---

11. What is the method to close a stream?
    a) finish()
    b) end()
    c) close()
    d) done()
    Ans: c

---

12. Which class is used to serialize an object to a file?
    a) ObjectOutput
    b) ObjectWriter
    c) ObjectOutputStream
    d) SerializableWriter
    Ans: c

13. What interface must be implemented to serialize a class?
   a) Writable
   b) Serializable
   c) Streamable
   d) Transferable
   Ans: b

---

14. Which keyword prevents a variable from being serialized?
   a) static
   b) final
   c) transient
   d) volatile
   Ans: c

---

15. Which stream is used for object serialization?
   a) FileInputStream
   b) ObjectInputStream
   c) ByteArrayOutputStream
   d) DataOutputStream
   Ans: b

---

16. What happens if a non-serializable class is written to `ObjectOutputStream`?
   a) Runtime exception
   b) Compilation error
   c) Nothing
   d) Skips writing
   Ans: a *(NotSerializableException)*

---

17. What is the return type of `readLine()` in `BufferedReader`?
   a) int
   b) char
   c) String
   d) Object
   Ans: c

18. Can static variables be serialized?
   a) Yes
   b) No
   c) Only final ones
   d) Only public ones
   Ans: b

19. What is the extension of a typical serialized object file?
   a) .txt
   b) .ser
   c) .obj
   d) .data
   Ans: b

20. What does `flush()` do in output streams?
   a) Deletes content
   b) Writes remaining buffer to output
   c) Closes the stream
   d) Resets the stream
   Ans: b

21. Which of these classes is not used for character stream I/O?
   a) FileReader
   b) FileWriter
   c) FileInputStream
   d) BufferedWriter
   Ans: c

22. Which exception is commonly thrown during file reading?
   a) NullPointerException
   b) IOException
   c) ClassNotFoundException
   d) FileFormatException
   Ans: b

---

23. To write formatted output to a file, use:
    a) FileWriter
    b) PrintWriter
    c) BufferedWriter
    d) OutputStream
    Ans: b

---

24. What method reads a single character from a `FileReader`?
    a) readChar()
    b) readByte()
    c) read()
    d) nextChar()
    Ans: c

---

25. Which class can read objects from a file?
    a) FileReader
    b) ObjectReader
    c) ObjectInputStream
    d) FileInputStream
    Ans: c

Excellent! Let's continue with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 9: File I/O and Serialization

---

26. What will be the output of the following?

File file = new File("test.txt");

file.createNewFile();

```
System.out.println(file.exists());
```

Assuming no exceptions are thrown and file doesn't already exist.
a) false
b) true
c) Compilation error
d) null
Ans: b

---

27. What is the default buffer size used in `BufferedReader` if not specified?
    a) 512 bytes
    b) 8192 characters
    c) 1024 bytes
    d) 256 characters
    Ans: b

---

28. How many objects are created in this code?

```
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("data.ser"));
```

a) 1
b) 2
c) 3
d) 0
Ans: b *(FileOutputStream + ObjectOutputStream)*

---

29. Which method writes a string into a file using `FileWriter`?
    a) insert()
    b) write()
    c) add()
    d) println()
    Ans: b

---

30. What is required to deserialize an object from file?
    a) Only ObjectInputStream

b) Serializable interface
c) Class must be available in classpath
d) All of the above
Ans: d

---

31. Which method checks if a path refers to a file and not a directory?
a) isFile()
b) isDirectory()
c) exists()
d) canRead()
Ans: a

---

32. Which of these classes supports random access file reading and writing?
a) BufferedReader
b) Scanner
c) RandomAccessFile
d) FileChannel
Ans: c

---

33. What does `mark()` and `reset()` methods in `BufferedReader` allow?
a) Marking line numbers
b) Skipping characters
c) Returning to a previously marked position
d) Formatting text
Ans: c

---

34. What will happen if you try to serialize a class containing a non-serializable object?
a) Nothing
b) Entire object is skipped
c) NotSerializableException
d) The object is serialized partially
Ans: c

---

35. What is the default serialVersionUID value if not declared manually?
a) 0L
b) JVM generated based on class structure
c) Random UUID
d) It's required to be declared
Ans: b

---

36. What is the correct way to read all lines from a file using `Files` class?
a) `Files.readAll("file.txt")`
b) `Files.read("file.txt")`
c) `Files.readAllLines(Path)`
d) `Files.getLines("file.txt")`
Ans: c

---

37. What does `transient` keyword do during serialization?
a) Skips method execution
b) Skips variable from serialization
c) Makes variable thread-safe
d) Makes class non-final
Ans: b

---

38. What is the return type of `File.listFiles()`?
a) List
b) ArrayList
c) File[]
d) String[]
Ans: c

---

39. What is the result of calling `deleteOnExit()`?
a) File is deleted immediately
b) File is deleted after program ends
c) File is archived
d) File is skipped
Ans: b

---

40. Which stream should be closed first when using ObjectOutputStream wrapped over FileOutputStream?
    a) ObjectOutputStream
    b) FileOutputStream
    c) Both simultaneously
    d) Doesn't matter
    Ans: a *(Always close the outer stream first)*

---

41. What does the `available()` method return in InputStreams?
    a) Total file size
    b) Number of characters
    c) Estimate of bytes available to read
    d) Remaining lines
    Ans: c

---

42. What is the purpose of `flush()` in output stream?
    a) Clears memory
    b) Closes the stream
    c) Ensures data is written from buffer to destination
    d) Deletes file
    Ans: c

---

43. If a class implements Serializable, but its superclass does not, what is serialized?
    a) Nothing
    b) Only subclass fields
    c) All fields
    d) Error occurs
    Ans: b

---

44. How do you skip characters in a `Reader`?
    a) skip(n)
    b) next(n)
    c) jump(n)
    d) ignore(n)

Ans: a

---

45. What does `File.createTempFile()` do?
    a) Creates and deletes a file
    b) Creates a new directory
    c) Creates a temporary file in system temp dir
    d) Overwrites existing file
    Ans: c

---

46. What is true about `ObjectInputStream.readObject()`?
    a) Returns byte array
    b) Requires type casting
    c) Returns only primitive types
    d) Returns InputStream
    Ans: b

---

47. What does `ObjectOutputStream.reset()` do?
    a) Resets the file
    b) Clears internal object cache to avoid object sharing
    c) Rewinds stream
    d) Deletes the file
    Ans: b

---

48. Which interface allows objects to define their own serialization logic?
    a) Externalizable
    b) Serializable
    c) Cloneable
    d) Writable
    Ans: a

---

49. What is the main difference between `Serializable` and `Externalizable`?
    a) Serializable is faster
    b) Serializable is manual
    c) Externalizable requires overriding `readExternal` and `writeExternal`

d) Serializable supports XML
Ans: c

---

50. What is the primary role of `ObjectStreamClass`?
    a) File operations
    b) Class metadata for serialization
    c) Reading strings
    d) Managing threads
    Ans: b

Great! Let's now complete:

🔴 HARD LEVEL MCQs (51–75)

Chapter 9: File I/O and Serialization

---

51. What is the result of writing the same object multiple times to `ObjectOutputStream` without resetting?

ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("data.ser"));

MyClass obj = new MyClass();

oos.writeObject(obj);

oos.writeObject(obj);

a) Two separate objects are stored
b) Only one instance is stored due to object reference caching
c) Throws an error
d) Overwrites file
Ans: b

---

52. How can you force `ObjectOutputStream` to write the same object again as new (not shared)?

a) Reopen the stream
b) Call `oos.flush()`
c) Call `oos.reset()`
d) Assign null before second write
Ans: c

---

53. What is the result of trying to read a serialized object with a different `serialVersionUID` than written?

a) Runtime warning
b) Object is deserialized
c) `InvalidClassException` is thrown
d) Null is returned
Ans: c

---

54. What happens if a file does not exist when `FileReader` is initialized?

a) File is created
b) Empty file is read
c) `FileNotFoundException` is thrown
d) Compilation error
Ans: c

---

55. What is the impact of marking a static field as transient?

a) Skips it from serialization
b) Causes error
c) No effect — static fields are not serialized anyway
d) Prevents compilation
Ans: c

---

56. If a class implements `Serializable` but one of its fields is not serializable, how can you handle it?

a) Mark the field `transient`
b) Use `Externalizable`
c) Remove the field
d) Catch IOException
Ans: a

---

57. In what order should streams be closed when chained?

```
BufferedWriter bw = new BufferedWriter(new FileWriter("file.txt"));
```

a) FileWriter first
b) BufferedWriter first
c) Any order
d) Don’t need to close
Ans: b

---

58. Which of these can store **binary data**?

a) FileReader
b) BufferedWriter
c) FileOutputStream
d) PrintWriter
Ans: c

---

59. What is the best method to **read and write binary data** to a file?

a) BufferedReader and BufferedWriter
b) DataInputStream and DataOutputStream
c) FileReader and FileWriter
d) PrintStream
Ans: b

---

60. How do you write an object to a file only if it's not null?

a) `if (object != null) oos.writeObject(object);`
b) `oos.write(object != null);`
c) Java handles it automatically
d) `oos.writeNull(object);`
Ans: a

---

61. What exception is thrown if a class is not found during deserialization?

a) FileNotFoundException
b) IOException
c) ClassNotFoundException
d) IllegalArgumentException
Ans: c

---

62. How can you manually define `serialVersionUID`?

a) Declare a static final long field in the class
b) Use an annotation
c) Define in `ObjectOutputStream`
d) It's not possible
Ans: a

---

63. When deserializing, which constructor is called?

a) Default constructor
b) No constructor is called
c) Public constructor
d) Only the constructor of Serializable interface
Ans: b

---

64. What does the `flush()` method ensure when working with output streams?

a) The stream is closed
b) Data is sent to disk immediately
c) Stream pointer resets
d) Nothing — it's redundant
Ans: b

65. What will happen if `flush()` is not called?

a) Data is lost
b) Data is not written immediately
c) File gets deleted
d) Exception is thrown
Ans: b

---

66. What's the difference between `BufferedWriter` and `FileWriter`?

a) FileWriter is buffered
b) BufferedWriter provides efficient writing via buffer
c) FileWriter doesn't exist
d) BufferedWriter is slower
Ans: b

---

67. How can you make a Java object **not serializable**, even if its class implements `Serializable`?

a) Make all fields static
b) Use a custom `writeObject()` method that throws exception
c) Use final fields
d) Set UID to 0
Ans: b

---

68. What is the result of writing a large object graph (many references) without resetting the ObjectOutputStream?

a) Performance degrades due to cache
b) Program crashes
c) All references are lost
d) All objects are re-serialized
Ans: a

---

69. Which encoding is used by `FileWriter` by default?

a) UTF-8
b) ASCII
c) JVM platform default
d) ISO-8859-1
Ans: c

---

70. What is the purpose of the `writeExternal()` method in Externalizable interface?

a) Define manual deserialization logic
b) Write object header
c) Handle exception
d) Write constant value
Ans: a

---

71. When should you use `FileChannel` over streams?

a) For reading large files with better performance
b) For character input
c) For writing text only
d) Only for console I/O
Ans: a

---

72. What is the main benefit of `try-with-resources` in file I/O?

a) Faster execution
b) Better performance
c) Automatic resource closing
d) Thread safety
Ans: c

---

73. What does this code do?

try (BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {

```
    br.readLine();
}
```

a) Closes the file automatically
b) Needs manual close
c) Causes exception
d) File is not read
Ans: a

---

74. What is the disadvantage of using `ObjectOutputStream` for appending to a file?

a) It overwrites the entire file
b) It adds stream headers again and causes corruption
c) It is slower than PrintWriter
d) It can only write strings
Ans: b

---

75. Which stream types are used in `Serializable` object transmission over network?

a) FileReader / FileWriter
b) InputStreamReader / OutputStreamWriter
c) ObjectInputStream / ObjectOutputStream
d) Scanner / PrintWriter
Ans: c

Great! Let's begin:

✅ Chapter 10: Java Networking

Total: 75 MCQs
Levels: 25 Easy | 25 Medium | 25 Hard
Topics Covered: Sockets, ServerSocket, URL, HttpURLConnection, DatagramSocket, InetAddress, TCP/UDP communication

---

🟢 EASY LEVEL MCQs (1–25)

---

1. Which package contains networking classes in Java?
   a) java.util
   b) java.net
   c) java.io
   d) java.network
   Ans: b

---

2. Which class is used for TCP client communication?
   a) DatagramSocket
   b) URL
   c) Socket
   d) InetAddress
   Ans: c

---

3. Which class is used to implement a TCP server in Java?
   a) DatagramSocket
   b) Socket
   c) ServerSocket
   d) URLConnection
   Ans: c

---

4. Which of the following is used for **UDP** communication?
   a) Socket
   b) DatagramSocket
   c) ServerSocket
   d) MulticastSocket
   Ans: b

---

5. What does `InetAddress.getLocalHost()` return?
   a) Local IP address
   b) Public IP
   c) Hostname only
   d) MAC address

Ans: a

---

6. What is the default port range for TCP/IP?
   a) 0–255
   b) 1024–65535
   c) 0–1023
   d) 0–65535
   Ans: d

---

7. Which class is used to connect to a URL and read from it?
   a) URLReader
   b) URLConnection
   c) URLScanner
   d) URLStream
   Ans: b

---

8. Which method of `Socket` returns the input stream?
   a) getInput()
   b) getInputStream()
   c) receive()
   d) readInput()
   Ans: b

---

9. What is the default port number of HTTP?
   a) 21
   b) 80
   c) 443
   d) 25
   Ans: b

---

10. What is the use of `DatagramPacket` in Java?
    a) It's for TCP packets
    b) It stores data for UDP communication
    c) It manages HTTP

d) It represents a socket
Ans: b

---

11. Which of these is **connectionless** protocol?
    a) TCP
    b) HTTP
    c) UDP
    d) FTP
    Ans: c

---

12. Which Java class provides host name resolution?
    a) Socket
    b) URL
    c) InetAddress
    d) HostManager
    Ans: c

---

13. Which method of `ServerSocket` waits for a client?
    a) wait()
    b) accept()
    c) listen()
    d) receive()
    Ans: b

---

14. What is the return type of `InetAddress.getByName(String host)`?
    a) IPAddress
    b) InetAddress
    c) String
    d) Socket
    Ans: b

---

15. What does `getPort()` return for a `Socket`?
    a) Client port
    b) Remote port

c) Server port
d) Error code
Ans: b

---

16. What exception must be handled when opening a socket?
a) FileNotFoundException
b) IOException
c) SocketClosedException
d) InterruptedException
Ans: b

---

17. What does `setSoTimeout(int timeout)` in `Socket` do?
a) Sets delay for data transfer
b) Sets connection timeout
c) Sets read timeout
d) Prevents server shutdown
Ans: c

---

18. Which stream reads bytes from a TCP connection?
a) FileInputStream
b) SocketInputStream
c) InputStream
d) StreamReader
Ans: c

---

19. Which method is used to send data using UDP?
a) write()
b) send(DatagramPacket)
c) push()
d) stream()
Ans: b

---

20. What protocol does Java `Socket` class use?
a) UDP

b) HTTP
c) TCP
d) FTP
Ans: c

---

21. Which Java class is used for creating URLs?
a) URLBuilder
b) HttpUrl
c) URL
d) URI
Ans: c

---

22. What is the result of calling `getHostAddress()` on an `InetAddress` object?
a) Domain name
b) MAC address
c) IP address as String
d) Port number
Ans: c

---

23. Which class is used for sending and receiving datagrams?
a) ServerSocket
b) Socket
c) DatagramSocket
d) FileSocket
Ans: c

---

24. Which class provides support for HTTP URL connections?
a) HttpServer
b) URL
c) HttpURLConnection
d) ServerSocket
Ans: c

---

25. What does `getOutputStream()` do on a socket?
    a) Reads from socket
    b) Sends output to console
    c) Returns stream for writing to socket
    d) Opens a new connection
    Ans: c

Great! Let's now proceed with:

🟡 MEDIUM LEVEL MCQs (26–50)

Chapter 10: Java Networking

---

26. What happens if a client tries to connect to a server on a port where no process is listening?
    a) `SocketTimeoutException`
    b) `ConnectException`
    c) `IOException`
    d) Program hangs
    Ans: b

---

27. Which of these steps is required to receive data using `DatagramSocket`?
    a) Accept the socket
    b) Read stream
    c) Use `receive(DatagramPacket)`
    d) Connect first
    Ans: c

---

28. How do you open a connection to a URL?

URL url = new URL("http://example.com");

URLConnection conn = _________;

a) `url.read()`
b) `url.open()`
c) `url.openConnection()`
d) `url.connect()`
Ans: c

---

29. Which method sends data over a TCP connection using `OutputStream`?
    a) send()
    b) write()
    c) flush()
    d) push()
    Ans: b

---

30. Which method is used to bind a `DatagramSocket` to a port?
    a) setPort()
    b) new DatagramSocket(port)
    c) bind()
    d) register()
    Ans: b

---

31. What is returned by `getInputStream()` from a `URLConnection` object?
    a) FileInputStream
    b) InputStream
    c) URLReader
    d) StreamConnection
    Ans: b

---

32. Which Java class is best for resolving a hostname to an IP address?
    a) InetAddress
    b) URL
    c) HostResolver
    d) SocketAddress
    Ans: a

---

33. What does `accept()` method of `ServerSocket` return?
    a) Socket
    b) URL
    c) Port
    d) ServerSocket
    Ans: a

---

34. What happens when you call `Socket.close()`?
    a) Closes the output stream only
    b) Shuts down the JVM
    c) Closes both input and output streams
    d) Flushes only
    Ans: c

---

35. Which protocol guarantees packet delivery?
    a) UDP
    b) HTTP
    c) TCP
    d) FTP
    Ans: c

---

36. What is a valid way to create a URL with protocol, host, and port?

```
new URL("http", "example.com", 8080, "/index.html");
```

a) Valid
b) Invalid — URL has no constructor
c) Invalid — missing IP
d) Invalid — must use https
Ans: a

---

37. What is the default timeout for a `Socket` connection?
    a) 0 (infinite)

b) 10s
c) 1s
d) 60s
Ans: a

---

38. What is the purpose of `HttpURLConnection.setRequestMethod("POST")`?
a) Reads data
b) Writes headers
c) Changes the HTTP method
d) Sends a ping
Ans: c

---

39. What is the purpose of `DatagramPacket(byte[] buf, int length)`?
a) Create a TCP packet
b) Store outgoing UDP data
c) Encrypt buffer
d) Create connection
Ans: b

---

40. What is the result of sending a large UDP packet over its maximum limit (~64KB)?
a) Fragments automatically
b) Packet is dropped
c) Delivered in parts
d) Written to disk
Ans: b

---

41. Which method of `Socket` is used to close the input stream only?
a) closeInput()
b) shutdownInput()
c) end()
d) setClosed(true)
Ans: b

---

42. How to send a string using TCP?

Socket s = new Socket("localhost", 1234);

OutputStream os = s.getOutputStream();

os.________;

a) send("Hello")
b) write("Hello")
c) write("Hello".getBytes())
d) println("Hello")
Ans: c

---

43. How does UDP ensure **faster** delivery than TCP?
a) Uses compression
b) Avoids acknowledgments
c) Has higher priority
d) Queues data
Ans: b

---

44. What method of `HttpURLConnection` is used to get response code?
a) getResponseStatus()
b) getStatusCode()
c) getResponseCode()
d) getCode()
Ans: c

---

45. What type of address is `127.0.0.1`?
a) External IP
b) Loopback address
c) MAC address
d) Private IPv6
Ans: b

---

46. Which of the following is **true** for `DatagramSocket` and `Socket`?

a) Both are for TCP
b) DatagramSocket uses connection-oriented protocol
c) Socket is for TCP, DatagramSocket is for UDP
d) Both support `accept()`
Ans: c

---

47. Which Java method sends data over a `DatagramSocket`?

a) transmit()
b) send()
c) write()
d) push()
Ans: b

---

48. What happens when `InetAddress.getByName("localhost")` is called?

a) Connects to the internet
b) Resolves to `127.0.0.1`
c) Creates a socket
d) Returns a URL
Ans: b

---

49. In client-server architecture, what role does `ServerSocket` play?

a) Sends packets
b) Receives datagrams
c) Listens for client connections
d) Resolves IP
Ans: c

---

50. Which method allows setting timeouts on `Socket` reads?

a) setTimeout()
b) setSoTimeout()
c) setDelay()
d) setReadLimit()
Ans: b

Excellent! Let’s now wrap up:

🔴 HARD LEVEL MCQs (51–75)

Chapter 10: Java Networking

---

51. What happens if you use `Socket.getInputStream().read()` on a closed socket?
    a) Returns -1
    b) Blocks forever
    c) Throws IOException
    d) Reopens the socket
    Ans: c

---

52. Which of the following can cause a `SocketTimeoutException`?
    a) Too many connections
    b) Server not listening
    c) `setSoTimeout()` triggered during read
    d) Port in use
    Ans: c

---

53. How does TCP ensure data is received in correct order?
    a) Checksums
    b) Sequence numbers
    c) Timestamps
    d) Buffer size
    Ans: b

---

54. What does this code output?

```
InetAddress local = InetAddress.getLocalHost();

System.out.println(local.getHostName());
```

a) IP address
b) Loopback address
c) Hostname of local machine
d) Domain name
Ans: c

---

55. Which class is used to parse query parameters in a URL?
    a) URLConnection
    b) URLEncoder
    c) URI
    d) URLDecoder
    Ans: d

---

56. Why is `flush()` important when writing to a socket stream?
    a) Closes the socket
    b) Forces buffered output to be sent immediately
    c) Clears all written content
    d) Creates a socket connection
    Ans: b

---

57. What is the result of using a port number below 1024 in `ServerSocket` without admin/root permissions?
    a) Nothing
    b) Binds successfully
    c) Throws `BindException`
    d) Skips port
    Ans: c

---

58. How do you create a UDP client that sends data to port 9090?

DatagramSocket socket = new DatagramSocket();

byte[] data = "Hello".getBytes();

DatagramPacket packet = new DatagramPacket(data, data.length, _________);

a) new URL("localhost", 9090)
b) new Socket("localhost", 9090)
c) InetAddress.getByName("localhost"), 9090
d) new InetAddress("localhost", 9090)
Ans: c

---

59. What is `getOutputStream()` used for in `HttpURLConnection`?
    a) Receiving data from server
    b) Sending request headers
    c) Sending data (like POST body) to server
    d) Reading response code
    Ans: c

---

60. How does `MulticastSocket` differ from `DatagramSocket`?
    a) Used only for TCP
    b) Used for broadcasting data to multiple receivers
    c) Slower
    d) It uses port 80
    Ans: b

---

61. How do you ensure a client sends a large file over TCP without blocking the server?

a) Use threads or async I/O
b) Use UDP
c) Flush constantly
d) Set larger port
Ans: a

---

62. Which method is used to **forcefully close** a `DatagramSocket`?
    a) disconnect()
    b) close()
    c) kill()
    d) stop()
    Ans: b

---

63. Which method returns the **remote IP address** connected to a socket?
   a) getLocalAddress()
   b) getPort()
   c) getInetAddress()
   d) getRemoteHost()
   Ans: c

---

64. How does Java determine the protocol (http, ftp, etc.) for a `URL`?
   a) Based on DNS
   b) From constructor's first argument
   c) Based on port
   d) From headers
   Ans: b

---

65. What does `URL.openStream()` return?
   a) URLConnection
   b) InputStream to read from URL
   c) Socket stream
   d) OutputStream
   Ans: b

---

66. Which is true about TCP?
   a) Connectionless
   b) Unreliable
   c) Ensures delivery, ordering, and error-checking
   d) Cannot send files
   Ans: c

---

67. What will this code do?

```
ServerSocket server = new ServerSocket(0);

System.out.println(server.getLocalPort());
```

a) Binds to port 0
b) Random available port is assigned
c) Port 80 is used
d) Throws exception
Ans: b

---

68. Which of the following allows **non-blocking I/O** operations in Java?
    a) DatagramSocket
    b) java.nio.channels.SocketChannel
    c) BufferedReader
    d) Scanner
    Ans: b

---

69. Why is `try-with-resources` recommended for networking code?
    a) It is faster
    b) It auto-closes streams and sockets
    c) It flushes automatically
    d) It ensures encryption
    Ans: b

---

70. How do you detect if a TCP connection is still alive in Java?
    a) getAlive()
    b) send urgent data or read with timeout
    c) call isConnected()
    d) isClosed() == false
    Ans: b

---

71. Which exception is thrown if two `ServerSocket` instances try to bind to the same port?
    a) IOException
    b) SocketTimeoutException
    c) BindException
    d) IllegalPortException
    Ans: c

---

72. Which of the following HTTP methods does Java support in `HttpURLConnection`?
  a) GET, POST
  b) PUT, DELETE
  c) HEAD
  d) All of the above
  Ans: d

---

73. Which method returns a `URLConnection` from a `URL` object?
  a) getURLConnection()
  b) openStream()
  c) openConnection()
  d) connect()
  Ans: c

---

74. Which of the following can be used to encode a URL query string?
  a) URLReader
  b) URLEncoder.encode(String, charset)
  c) URI.getQuery()
  d) HttpURLConnection
  Ans: b

---

75. Which Java class provides information like content type, length, and date of a URL resource?
  a) URL
  b) HttpURLConnection
  c) URI
  d) InetAddress
  Ans: b