CORE JAVA MCQS

✓✓✓

Session 1 & 2: Java Language Basics, JDK/JRE/JVM, Data Types, Variables, Operators, Control Statements

1. Which of the following can be operands of arithmetic operators?
a) Numeric
b) Boolean
c) Characters
d) Both Numeric & Characters

Ans: D) Both Numeric and characters

2. Which of the following is the primary purpose of the Java Virtual Machine (JVM)?

a) To compile Java source code (.java files) into bytecode.
b) To provide a runtime environment for executing Java bytecode.
c) To manage project dependencies and build artifacts.
d) To write platform-specific operating system commands.

Ans: B) To provide a runtime environment for executing Java bytecode.

3. What is the correct file extension for a compiled Java class file?

a) .java
b) .jar
c) .class
d) .exe

Ans: C) .class

4. Which component is responsible for providing the development tools (compiler, debugger, etc.) along with the runtime environment?

a) JRE
b) JVM
c) JDK
d) JIT

Ans: C) JDK

5. Which of the following is NOT a primitive data type in Java?

a) int
b) boolean
c) String
d) char

Ans: C) String

6. What is the default value of an int variable in Java if it's declared as a class member (instance variable) but not explicitly initialized?

a) null
b) 0
c) false
d) Undefined; compilation error

Ans: B) 0

7. Which operator is used for logical OR in Java?

a) &
b) ||
c) &&
d) !

Ans: B) ||

8. What will be the value of x after the following code executes?

Java

```
int x = 10;
x += 5;
x--;
```

a) 15
b) 14
c) 16
d) 10

Ans: B) 14 (10 + 5 = 15; 15 - 1 = 14)

9. Which control statement is best suited for executing a block of code repeatedly a fixed number of times?

a) if-else
b) while loop
c) do-while loop
d) for loop

Ans: D) for loop

10. Modulus operator, %, can be applied to which of these?

a) Integers
b) Floating – point numbers
c) Both Integers and floating – point numbers
d) None of the mentioned

Ans: C) Both Integers and floating-point numbers

11. With x = 0, which of the following are legal lines of Java code for changing the value of x to 1?

1. x++;
2. x = x + 1;
3. x += 1;
4. x =+ 1;

a) 1, 2 & 3
b) 1 & 4
c) 1, 2, 3 & 4
d) 3 & 2

Ans: C) 1, 2, 3, & 4

12. What will be the output of the following Java program?

```
1.  class increment
2.  {
3.     public static void main(String args[])
4.     {
5.        double var1 = 1 + 5;
6.        double var2 = var1 / 4;
7.        int var3 = 1 + 5;
8.        int var4 = var3 / 4;
9.        System.out.print(var2 + " " + var4);
10.
11.    }
12. }
```

a) 1 1
b) 0 1
c) 1.5 1
d) 1.5 1.0

Ans: C) 1.5 1

13. Which of these have highest precedence?

a) ()
b) ++
c) *
d) >>

Ans: A) ()

14. Which of these selection statements test only for equality?

a) if
b) switch
c) if & switch
d) none of the mentioned

Ans: B) switch

15. Which of the following loops will execute the body of loop even when condition controlling the loop is initially false?

a) do-while
b) while
c) for
d) none of the mentioned

Ans: A) do-while

16. What is it called where child object gets killed if parent object is killed?

a) Aggregation
b) Composition
c) Encapsulation
d) Association

Ans: B) Composition

17. Which component of the JVM is responsible for loading class files into memory?

a) Garbage Collector
b) Class Loader
c) Execution Engine
d) JIT Compiler
Ans: B) Class Loader

18. Which of the following is ONLY contained within the JDK and not the JRE?
a) The `rt.jar` file containing core Java libraries
b) The Java Virtual Machine (JVM)
c) `java.exe` (or `java` on Unix-based systems)
d) `javac.exe` (or `javac` on Unix-based systems)
Ans: D) `javac.exe` (or `javac` on Unix-based systems)

19. Which of the following best describes the relationship between JDK, JRE, and JVM?

a) JDK, JRE, and JVM are all independent of each other.
b) JDK is a subset of JRE, which is a subset of JVM.
c) JVM is a subset of JRE, which is a subset of JDK.
d) JRE is a subset of JDK, which is a subset of JVM.
Ans: C) JVM is a subset of JRE, which is a subset of JDK.

20. In which JVM memory area are local variables of a method stored?
a) Heap
b) Native Method
c) Stack
d) Method
Ans: C) Stack

21. Which of the following is a key responsibility of the JVM's Execution Engine?
a) Executing the instructions in the bytecode.
b) Loading class files from the disk.
c) Allocating memory for new objects.
d) Providing native method implementations.

22. Can you have multiple JREs installed on the same machine?
a) No, because it would cause conflicts with the system's PATH variable.
b) Yes, but only if they are all the same version.
c) No, only one JRE can be installed at a time.
d) Yes, and you can specify which JRE to use for a particular application.
Ans: Yes, and you can specify which JRE to use for a particular application.

23. Consider the following switch statement:

Java

```
int day = 3;
String dayName;
switch (day) {
    case 1: dayName = "Monday";
    case 2: dayName = "Tuesday";
    case 3: dayName = "Wednesday";
    case 4: dayName = "Thursday"; break;
    default: dayName = "Invalid Day";
}
System.out.println(dayName);
```

What will be the output, and why?

a) Wednesday, because day is 3.

b) Thursday, because of fall-through until break;.

c) Invalid Day, because there's no break after case 3.

d) Compilation Error, missing break statements.

Ans: B) Thursday, because of fall-through until break;

24. Which of the following correctly defines a long literal in Java?

a) long bigNumber = 12345678901;

b) long bigNumber = 12345678901L;

c) long bigNumber = (long) 12345678901;

d) long bigNumber = "12345678901";

Ans: B) long bigNumber = 12345678901L;

25. Why is Java often referred to as "write once, run anywhere" (WORA)?

a) Because Java programs are compiled directly into machine code for all platforms.

b) Because the JVM acts as an abstraction layer, translating bytecode into platform-specific instructions.

c) Because Java source code can be executed directly without compilation.

d) Because Java only runs on specific operating systems.

Ans: B) Because the JVM acts as an abstraction layer, translating bytecode into platform-specific instructions.

26. You have a Java program that compiles successfully on your Windows machine. You then take the generated .class files and try to run them on a Linux machine that has only the JRE installed. Will the program run? Why or why not?

a) No, because the JRE on Linux cannot execute Windows-compiled .class files.

b) Yes, because Java bytecode is platform-independent, and JRE provides the runtime environment.

c) No, because you need the JDK on the Linux machine for execution.

d) Yes, but only if the Linux machine has the same architecture as the Windows machine.

Ans: B) Yes, because Java bytecode is platform-independent, and JRE provides the runtime environment.

27. A developer wants to create a constant in Java that represents the value of PI. Which of the following is the most appropriate way to declare it?

a) final double PI = 3.14159;

b) double PI = 3.14159;

c) static double PI = 3.14159;

d) const double PI = 3.14159;

Ans: A) final double PI = 3.14159;

28. Why is Java often referred to as "write once, run anywhere" (WORA)?

a) Because Java programs are compiled directly into machine code for all platforms.

b) Because the JVM acts as an abstraction layer, translating bytecode into platform-specific instructions.

c) Because Java source code can be executed directly without compilation.

d) Because Java only runs on specific operating systems.

Ans: B) Because the JVM acts as an abstraction layer, translating bytecode into platform-specific instructions.

Session 3 & 4: Object-Oriented Concepts, Classes, Objects, Constructors, Access Specifiers

29. Which access specifier allows a member to be accessed only by subclasses and classes in the same package?

a) public

b) private

c) protected

d) default

Ans: c) protected

30. Which of the following best describes an "object" in Java?
a) A blueprint or template for creating data.
b) An instance of a class, representing a real-world entity.
c) A collection of static methods.
d) A primitive data type.
Ans: B) An instance of a class, representing a real-world entity.

31. If a class does not explicitly define any constructor, what happens?
a) A compilation error occurs.
b) The class cannot be instantiated.
c) The Java compiler provides a default no-argument constructor.
d) Only parameterized constructors can be called.
Ans: C) The Java compiler provides a default no-argument constructor.

32. What is the primary role of a constructor in Java?
a) To destroy objects when they are no longer needed.
b) To initialize the state (instance variables) of an object when it's created.
c) To define the behavior (methods) of a class.
d) To declare static variables.
Ans: B) To initialize the state (instance variables) of an object when it's created.

33. Which access specifier makes a member visible only within its own class?
a) public
b) protected
c) default (package-private)
d) private
Ans: D) private

34. Which of the following keywords is used to define a class in Java?
a) object
b) create
c) class
d) define
Ans: C) class

35. What does the new operator do in Java?
a) Declares a new class.
b) Creates a new instance (object) of a class.
c) Defines a new method.
d) Assigns a new value to a variable.
Ans: B) Creates a new instance (object) of a class.

36. In Java, "tokens" are the smallest individual units of a program. Which of the following is considered a token?
a) Keywords (public, class, int)
b) Identifiers (myVariable, ClassName)
c) Operators (+, =, ==)
d) All of the above
Ans: D) All of the above

37. You are designing a Student class. You want to ensure that every Student object is created with a unique studentId. Which type of constructor would be most appropriate to enforce this, and why?
a) A no-argument constructor, because it's simplest.
b) A private constructor, to prevent direct instantiation.
c) A parameterized constructor that accepts studentId as an argument, forcing the caller to provide it at creation.
d) A copy constructor, to duplicate existing student IDs.
Ans: C) A parameterized constructor that accepts studentId as an argument, forcing the caller to provide it at creation.

38. If a class has multiple constructors, what is this OOP concept called?
a) Constructor Overriding
b) Constructor Hiding
c) Constructor Overloading
d) Constructor Abstraction
Ans: C) Constructor Overloading

39. You have two classes in the same package: ClassA and ClassB. ClassA has a method doSomething() with default (package-private) access. Can ClassB call this method?

a) No, default access is only for the defining class.

b) Yes, because they are in the same package.

c) Only if ClassB is a subclass of ClassA.

d) Only if doSomething() is declared static.

Ans: B) Yes, because they are in the same package.

40. Consider the following class:

Java

```
public class Product {
    private String name;
    private double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public double getPrice() {
        return price;
    }
}
```

If you create an object: Product p = new Product("Laptop", 1200.00);, how can you access the name of this product?

a) System.out.println(p.name);

b) System.out.println(Product.name);

c) You cannot directly access name because it is private, a getter method is needed.

d) System.out.println(p.getName()); (assuming a getName() method exists)

Ans: C) You cannot directly access name because it is private, a getter method is needed.

41. What is the key difference between Java's concept of "namespaces" (often managed by packages) and access specifiers?

a) Namespaces control object creation, while access specifiers control memory allocation.

b) Namespaces prevent naming conflicts for classes, while access specifiers control visibility of members within and across packages/classes.

c) Namespaces are only for variables, while access specifiers are for methods.

d) They are essentially the same concept with different names.

Ans: B) Namespaces prevent naming conflicts for classes, while access specifiers control visibility of members within and across packages/classes.

Session 6: Abstraction, Data Hiding, Encapsulation, Polymorphism (Runtime/Compile-time), Inheritance

42. You are designing a BankAccount class. You want to ensure that the balance field can only be modified through deposit() and withdraw() methods, and not directly accessed from outside the class. Which OOP concept would you primarily use, and how would you implement it?
a) Abstraction; declare balance as abstract.
b) Polymorphism; overload the balance variable.
c) Encapsulation; declare balance as private and provide public deposit() and withdraw() methods.
d) Inheritance; extend BankAccount from a FinancialObject class.
Ans: C) Encapsulation; declare balance as private and provide public deposit() and withdraw() methods.

43. What is the primary benefit of using inheritance in Java?
a) To achieve data hiding
b) To enable garbage collection
c) To promote code reusability
d) To prevent method overriding
Ans: C) To promote code reusability

171. Which OOP principle involves representing essential features without including background details or explanations?
a) Encapsulation
b) Polymorphism
c) Abstraction
d) Inheritance
Ans: C) Abstraction

172. Encapsulation combines data and the methods that operate on the data into a single unit. What is the primary benefit of this?
a) To enable multiple inheritance.
b) To achieve data hiding and protect the internal state of an object.
c) To allow all members to be accessed directly from anywhere.
d) To make the code faster.
Ans: B) To achieve data hiding and protect the internal state of an object.

173. How is data hiding typically achieved in Java?
a) By declaring fields as public.
b) By using protected access modifier.
c) By declaring fields as private and providing public accessor (getter) and mutator (setter) methods.
d) By making the class abstract.
Ans: C) By declaring fields as private and providing public accessor (getter) and mutator (setter) methods.

174. Which of the following is an example of Compile-time Polymorphism?
a) Method Overriding
b) Method Overloading
c) Dynamic Method Dispatch
d) Abstract Methods
Ans: B) Method Overloading

175. What is the "Is-A" relationship primarily associated with in OOP?
a) Aggregation
b) Composition
c) Inheritance
d) Association
Ans: C) Inheritance

176. In Java, the extends keyword is used to achieve which OOP principle?
a) Abstraction
b) Encapsulation
c) Inheritance
d) Polymorphism
Ans: C) Inheritance

177. What is the core idea behind Polymorphism?
a) An object can take on "many forms" or behave differently based on the context.
b) Hiding the implementation details of a class.
c) Creating new classes from existing ones.
d) Bundling data and methods into a single unit.
Ans: A) An object can take on "many forms" or behave differently based on the context.

178. Which of the following types of inheritance is NOT directly supported by Java classes (but can be achieved through interfaces)?
a) Single Inheritance
b) Multilevel Inheritance
c) Hierarchical Inheritance
d) Multiple Inheritance
Ans: D) Multiple Inheritance

179. A Java class serves as a "template" or blueprint for what?
a) Other classes.
b) Methods.
c) Objects.
d) Variables.
Ans: C) Objects.

180. Which of the following is a key characteristic of an abstract class in Java?
a) It can be instantiated directly.
b) It can contain abstract methods, which must be implemented by concrete subclasses.
c) It cannot have any constructors.
d) All its methods must be abstract.
Ans: B) It can contain abstract methods, which must be implemented by concrete subclasses.

181. When a method in a subclass has the same signature (name and parameters) as a method in its superclass, what is this called?
a) Method Overloading
b) Method Hiding
c) Method Overriding
d) Method Dispatch
Ans: C) Method Overriding

182. What is the default access modifier for members (fields and methods) within a class if no modifier is explicitly specified?
a) public
b) protected
c) private
d) default (package-private)
Ans: D) default (package-private)

183. Which OOP concept refers to the ability of an object to respond to a method call based on the actual object type at runtime?
a) Static Binding
b) Early Binding
c) Dynamic Method Dispatch
d) Method Overloading
Ans: C) Dynamic Method Dispatch

184. What is the main advantage of using inheritance?
a) To restrict class access.
b) To achieve data integrity.
c) To promote code reusability and establish an "Is-A" hierarchy.
d) To remove the need for constructors.
Ans: C) To promote code reusability and establish an "Is-A" hierarchy.

185. If a class Dog extends Animal, and Animal has a public method eat(), how can Dog provide its own specific implementation of eat()?
a) By using method overloading.
b) By declaring eat() as final in Dog.
c) By overriding the eat() method in the Dog class.
d) By making eat() private in Dog.
Ans: C) By overriding the eat() method in the Dog class.

186. You are building a Car class. You want to model its speed and fuelLevel attributes. You need to ensure that these attributes can only be changed by the accelerate(), brake(), and refuel() methods. Which OOP principle and implementation strategy would you use?

a) Polymorphism; make speed and fuelLevel public.

b) Abstraction; declare speed and fuelLevel as abstract.

c) Encapsulation; declare speed and fuelLevel as private and provide public methods for interaction.

d) Inheritance; extend Car from a Vehicle class.

Ans: C) Encapsulation; declare speed and fuelLevel as private and provide public methods for interaction.

187. You are building a Car class. You want to model its speed and fuelLevel attributes. You need to ensure that these attributes can only be changed by the accelerate(), brake(), and refuel() methods. Which OOP principle and implementation strategy would you use?

a) Polymorphism; make speed and fuelLevel public.

b) Abstraction; declare speed and fuelLevel as abstract.

c) Encapsulation; declare speed and fuelLevel as private and provide public methods for interaction.

d) Inheritance; extend Car from a Vehicle class.

Ans: C) Encapsulation; declare speed and fuelLevel as private and provide public methods for interaction.

188. Consider the following Java code snippet:

Java

```
class Shape {
    void draw() {
        System.out.println("Drawing a generic shape.");
    }
}

class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a circle.");
    }
}

class Rectangle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a rectangle.");
    }
}

public class PolymorphismDemo {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Rectangle();
        Shape s3 = new Shape();

        s1.draw();
        s2.draw();
        s3.draw();
    }
}
```

What will be the output of this program, and which OOP principle is primarily demonstrated?
a) "Drawing a generic shape." (3 times); Principle: Inheritance.
b) "Drawing a circle." "Drawing a rectangle." "Drawing a generic shape."; Principle: Runtime Polymorphism (Dynamic Method Dispatch).
c) "Drawing a circle." "Drawing a rectangle." "Drawing a generic shape."; Principle: Compile-time Polymorphism.
d) "Drawing a generic shape." (3 times); Principle: Encapsulation.
Ans: b) "Drawing a circle." "Drawing a rectangle." "Drawing a generic shape."; Principle: Runtime Polymorphism (Dynamic Method Dispatch).

189. Why does Java not support multiple inheritance of classes directly (like C++)?
a) To reduce the number of keywords in the language.
b) To simplify compilation and avoid the "Diamond Problem" (ambiguity in method inheritance).
c) Because objects can only have one parent.
d) It makes the garbage collection process more complex.
Ans: B) To simplify compilation and avoid the "Diamond Problem" (ambiguity in method inheritance).

190. You have a class Calculator with a method add(int a, int b). You now need to add a method to add three numbers. You create another method add(int a, int b, int c) in the same class. What OOP concept have you applied?
a) Method Overriding
b) Method Hiding
c) Method Overloading
d) Dynamic Method Dispatch
Ans: C) Method Overloading

191. Which of the following scenarios is best suited for using an abstract class over an interface?
a) When you need to define a contract for unrelated classes.
b) When you want to provide a common base implementation (with state and concrete methods) that subclasses can inherit, along with some abstract methods that must be implemented.
c) When you want to achieve multiple inheritance of type.
d) When you only need to define constants.
Ans: B) When you want to provide a common base implementation (with state and concrete methods) that subclasses can inherit, along with some abstract methods that must be implemented.

192. You define a Vehicle class and then Car and Motorcycle classes. Car and Motorcycle are Vehicles. Later, you realize that all Vehicles should have a startEngine() method, but the implementation varies greatly. What is the most appropriate way to ensure all subclasses implement this, while still allowing the Vehicle class to provide common concrete methods like getTopSpeed()?
a) Make Vehicle a final class with a default startEngine() method.
b) Make Vehicle an interface with startEngine() and getTopSpeed().
c) Make Vehicle an abstract class with an abstract startEngine() method and a concrete getTopSpeed() method.
d) Use method overloading for startEngine() in each subclass.
Ans: C) Make Vehicle an abstract class with an abstract startEngine() method and a concrete getTopSpeed() method.

193. Consider a class Person with private String name; and a public String getName() method. If you modify the internal storage of name to private char[] nameChars;, but keep getName() working correctly, which principle ensures that external code using getName() is unaffected?
a) Polymorphism
b) Inheritance
c) Abstraction and Encapsulation
d) Method Overloading
Ans: C) Abstraction and Encapsulation

194. What is the result of applying the final keyword to a method in a superclass?
a) It makes the method static.
b) It makes the method abstract.
c) It prevents subclasses from overriding that method.
d) It ensures the method cannot be called from outside the class.
Ans: C) It prevents subclasses from overriding that method.

195. You have a PrintService class with a method print(String document). You want to extend its functionality to print(Image image) and print(PDFDocument pdf). How would you achieve this in PrintService?
a) By creating new subclasses for each document type.
b) By using method overriding.
c) By using method overloading within the PrintService class.
d) By declaring the print method as abstract.
Ans: C) By using method overloading within the PrintService class.

196. If ClassA has a private field int value;, and ClassB extends ClassA. Which of the following is true?
a) ClassB can directly access value using super.value.
b) ClassB can directly access value using this.value.
c) ClassB can only access value through public or protected methods provided by ClassA.
d) ClassB inherits value but cannot see it.
Ans: C) ClassB can only access value through public or protected methods provided by ClassA.

197. Which of the following is the best example of a "Has-A" relationship in Java?
a) Dog extends Animal
b) Car has an Engine
c) InterfaceA implements InterfaceB
d) MethodA calls MethodB
Ans: B) Car has an Engine

198. You have an interface Drivable with an abstract method drive(). A class Car implements Drivable. If you have Drivable vehicle = new Car();, and you call vehicle.drive(), what mechanism determines which drive() method is executed?
a) Compile-time Binding
b) Early Binding
c) Dynamic Method Dispatch
d) Static Polymorphism
Ans: C) Dynamic Method Dispatch

199. In Java, when designing a reusable component that needs to define a common behavior across different, potentially unrelated classes, but does not need to share implementation code or state, what OOP construct is typically used?
a) A concrete class
b) An abstract class
c) An interface
d) A final class
Ans: C) An interface

200. What is the purpose of the @Override annotation in Java?
a) It marks a method for garbage collection.
b) It indicates that a method is inherited from a superclass or interface.
c) It explicitly indicates that a method is intended to override a method in a superclass or implement an interface method, helping the compiler catch errors.
d) It makes a method final so it cannot be overridden further.
Ans: C) It explicitly indicates that a method is intended to override a method in a superclass or implement an interface method, helping the compiler catch errors.

201. Consider the scenario where a Shape class has a calculateArea() method. Circle and Rectangle are subclasses, each providing its own specific implementation of calculateArea(). When you store Circle and Rectangle objects in an ArrayList<Shape> and iterate through them, calling calculateArea() on each Shape reference, what allows the correct calculateArea() to be invoked for each specific object?
a) Method Overloading
b) Static binding
c) Runtime Polymorphism (Dynamic Method Dispatch)
d) Encapsulation
Ans: C) Runtime Polymorphism (Dynamic Method Dispatch)

202. If you have a class LivingBeing and subclasses Plant and Animal, this is an example of:
a) Single Inheritance
b) Multilevel Inheritance
c) Hierarchical Inheritance
d) Multiple Inheritance
Ans: C) Hierarchical Inheritance

203. A "template" in OOP generally refers to a generic blueprint. How does Java support generic programming which is analogous to templates in other languages like C++?
a) Through final classes.
b) Through abstract classes.
c) Through Generics (e.g., ArrayList<E>), allowing classes/methods to operate on objects of various types while maintaining type safety.
d) Java does not support templates.
Ans: C) Through Generics (e.g., ArrayList<E>), allowing classes/methods to operate on objects of various types while maintaining type safety.

204. You have a class Parent and a class Child (where Child extends Parent). If Parent has a public static method display(), and Child also defines a public static method display(), what happens?
a) Child overrides display().
b) Child overloads display().
c) Child hides display() (static method hiding).
d) This results in a compilation error.
Ans: C) Child hides display() (static method hiding).

205. Which of the following is true about polymorphism and the "Is-A" relationship?
a) Polymorphism can exist without an "Is-A" relationship.
b) An "Is-A" relationship is a prerequisite for runtime polymorphism.
c) They are completely unrelated concepts.
d) "Is-A" only applies to interfaces, not classes.
Ans: B) An "Is-A" relationship is a prerequisite for runtime polymorphism.

206. Why would a designer declare a class to be abstract rather than a concrete class?
a) To prevent it from being extended.
b) To force subclasses to provide implementations for certain methods, ensuring a common interface while allowing varied behavior.
c) To ensure all its fields are final.
d) To make it thread-safe.
Ans: B) To force subclasses to provide implementations for certain methods, ensuring a common interface while allowing varied behavior.

Session 7: Abstract Class, Interface (implementing multiple interfaces)

44. Suppose you have an interface Flyable with a method fly() and another interface Swimmable with a method swim(). If a class Duck implements both Flyable and Swimmable, what must be true about the Duck class?
a) It must declare fly() and swim() as abstract.
b) It must provide concrete implementations for both fly() and swim() methods.
c) It can only implement one of the methods and declare the other as abstract.
d) It cannot implement both interfaces due to Java's single inheritance rule.
Ans: B) It must provide concrete implementations for both fly() and swim() methods.

45. You are creating a utility library that needs to provide a contract for any object that can be "printed." The print() method will vary widely depending on the object. You also want to make sure that these "printable" objects can be part of a larger collection. Which approach is more flexible: an abstract class Printable or an interface Printable? Justify your answer.
a) An abstract class Printable because it can contain state.
b) An interface Printable because a class can implement multiple interfaces, allowing it to inherit printing behavior without affecting its primary class hierarchy.
c) Either would work equally well.
d) A concrete class Printable is best.
Ans: B) An interface Printable because a class can implement multiple interfaces, allowing it to inherit printing behavior without affecting its primary class hierarchy.

46. Consider an abstract class Shape with an abstract method calculateArea(). If Circle extends Shape and Square also extends Shape, what is mandatory for Circle and Square?
a) They must both be declared abstract.
b) They must both have a main method.
c) They must provide concrete implementations for the calculateArea() method.
d) They can choose whether or not to implement calculateArea().
Ans: C) They must provide concrete implementations for the calculateArea() method.

47. Can a Java class implement multiple interfaces?
a) No, a class can implement only one interface.
b) Yes, but only if the interfaces have no common methods.
c) Yes, a class can implement any number of interfaces.
d) Only if one of the interfaces is abstract.
Ans: C) Yes, a class can implement any number of interfaces.

48. Which keyword is used to declare a class that cannot be instantiated on its own and may contain abstract methods?

a) interface
b) final
c) abstract
d) static

Ans: C) Abstract

Session 8: Final Keyword, Functional Interfaces, Arrays, Enumerations

49. What happens if you declare a variable as static final int x; without initialization in a class?
a) It is automatically initialized to 0.
b) It causes a compilation error.
c) It is initialized to null.
d) It is initialized at runtime.
Answer: B) It causes a compilation error.

50. What is the effect of declaring a variable as final in Java?
a) Its value can be changed multiple times.
b) Its value must be initialized at declaration and cannot be changed thereafter.
c) It can only be accessed by static methods.
d) It marks the variable for garbage collection.
Ans: B) Its value must be initialized at declaration and cannot be changed thereafter.

51. If a method is declared as final in a superclass, what does it imply for its subclasses?
a) Subclasses must override this method.
b) Subclasses cannot override this method.
c) Subclasses can only access this method via super.
d) The method can only be called from within the superclass.
Ans: B) Subclasses cannot override this method.

52. What happens if a class is declared as final?
a) Its methods cannot be overridden.
b) It cannot have any constructors.
c) It cannot be extended (subclassed).
d) Its objects cannot be garbage collected.
Ans: C) It cannot be extended (subclassed).

53. Which of the following best describes a "functional interface" in Java?
a) An interface with exactly zero abstract methods.
b) An interface with exactly one abstract method.
c) An interface with multiple abstract methods.
d) An interface that uses the final keyword.
Ans: B) An interface with exactly one abstract method.

54. Which annotation is used to explicitly mark an interface as a functional interface (though optional if it meets the criteria)?
a) @Override
b) @FunctionalInterface
c) @Interface
d) @Singleton
Ans: B) @FunctionalInterface

55. What is the default value of elements in a newly created int array in Java?
a) null
b) 0
c) -1
d) Garbage values
Ans: B) 0

56. How do you declare and initialize an array of 5 String objects in Java?
a) String[] names = {"", "", "", "", ""};
b) String[] names = new String[5];
c) String names[5];
d) String names = new String[5];
Ans: B) String[] names = new String[5];

57. Which of the following is true about Java enumerations (enum)?
a) They are special types of classes.
b) Their instances are implicitly public static final.
c) They can have constructors, methods, and fields.
d) All of the above.
Ans: D) All of the above.

58. Which of the following is a new feature introduced in interfaces starting from Java 8?
a) Abstract methods
b) Static variables
c) Default methods
d) Private constructors
Ans: C) Default methods

59. Can we create an instance of Enum outside of Enum itself?

a) True
b) False

Ans: B) False

60. If we try to add Enum constants to a TreeSet, what sorting order will it use?

a) Sorted in the order of declaration of Enums
b) Sorted in alphabetical order of Enums
c) Sorted based on order() method
d) Sorted in descending order of names of Enums

Ans: A) Sorted in the order of declaration of Enums

61. Are enums are type-safe?

a) True
b) False
Ans: A) True

62. If a method is declared as final in a superclass, what does it imply for its subclasses?

a) Subclasses must override this method.

b) Subclasses cannot override this method.

c) Subclasses can only access this method via super.

d) The method can only be called from within the superclass.

Ans: B) Subclasses cannot override this method.

63. What is the default value of elements in a newly created int array in Java?

a) null

b) 0

c) -1

d) Garbage values

Ans: B) 0

64. Consider the following code snippet:

Java

```
public class MyClass {
    final int VALUE;
    public MyClass() {
        VALUE = 100;
    }
    public MyClass(int val) {
        VALUE = val;
    }
}
```

Is this code valid? If so, why? If not, why not?

a) Valid. final variables can be initialized in the constructor.

b) Invalid. final variables must be initialized at declaration.

c) Valid. final variables can be re-assigned multiple times.

d) Invalid. A class with final fields cannot have multiple constructors.

Ans: A) Valid. final variables can be initialized in the constructor.

65. You are designing a Configuration class that should store a set of predefined, unchangeable application states (e.g., STARTED, PAUSED, STOPPED). You want to ensure type safety and prevent invalid state values. Which Java feature is best suited for this purpose?

a) A set of final static int constants.

b) A final class with private constructor.

c) An enum type.

d) A String array.

Ans: C) An enum type.

66. You have an enumeration enum Day { MONDAY, TUESDAY, WEDNESDAY };. If you try to create a new instance of Day using Day newDay = new Day();, what will happen?

a) A new Day instance will be created.

b) A NullPointerException.

c) A compile-time error because enum constructors are implicitly private.

d) A runtime error indicating an invalid operation.

Ans: C) A compile-time error because enum constructors are implicitly private.

67. What is the primary benefit of using default methods in interfaces (Java 8+)?
a) They allow an interface to contain state (instance variables).
b) They facilitate adding new methods to existing interfaces without breaking backward compatibility for implementing classes.
c) They make interfaces directly instantiable.
d) They replace the need for abstract classes.
Ans: B) They facilitate adding new methods to existing interfaces without breaking backward compatibility for implementing classes.

68. Why would a developer choose to declare a class as final?
a) To enable multiple inheritance.
b) To allow extensive modification by subclasses.
c) To prevent other classes from extending it, often for security or design immutability.
d) To make all its methods abstract by default.
Ans: c) To prevent other classes from extending it, often for security or design immutability.

69. You have an interface Calculator with a single abstract method int operate(int a, int b);. Which of the following is a valid way to implement this interface using a Java 8 functional feature?
a) Calculator calc = new Calculator();
b) Calculator calc = (a, b) -> a + b;
c) Calculator calc = public int operate(int a, int b) { return a * b; };
d) Calculator calc = Calculator::new;
Ans: B) Calculator calc = (a, b) -> a + b;

70. Consider an array int[] numbers = {10, 20, 30};. If you try to access numbers[3], what kind of error will occur?
a) Compile-time error
b) Syntax error
c) ArrayIndexOutOfBoundsException at runtime
d) NullPointerException at runtime
Ans: C) ArrayIndexOutOfBoundsException at runtime

Session 9: Garbage Collection in Java

71. Which of the following makes an object eligible for garbage collection?
a) Setting a reference to null
b) Declaring a variable as final
c) Assigning a new value to a primitive variable
d) Calling System.gc()
Answer: A**) Setting a reference to null**

72. What is the primary purpose of Garbage Collection in Java?
a) To delete unused variables from the stack memory.
b) To automatically free up memory occupied by objects that are no longer referenced.
c) To optimize the performance of I/O operations.
d) To compile Java source code into bytecode.
Ans: B) To automatically free up memory occupied by objects that are no longer referenced.

73. When is an object considered "eligible for garbage collection" in Java?
a) When its constructor finishes execution.
b) When it is explicitly set to null by the programmer.
c) When it is no longer reachable from any active reference.
d) When the System.gc() method is explicitly called.
Ans: c) When it is no longer reachable from any active reference.

74. Which of the following methods can be called to request the JVM to run the garbage collector, but does not guarantee immediate execution?
a) System.runGarbageCollector()
b) Runtime.collectGarbage()
c) System.gc()
d) JVM.cleanup()
Ans: C) System.gc()

75. What is the finalize() method used for in Java?
a) To explicitly free memory of an object.
b) To perform cleanup activities before an object is garbage collected.
c) To mark an object as eligible for garbage collection.
d) To prevent an object from being garbage collected.
Ans: B) To perform cleanup activities before an object is garbage collected.

76. Consider the following code snippet:

Java

```
Object obj1 = new Object();
Object obj2 = new Object();
obj1 = null;
obj2 = obj1;
```

After these lines, how many objects initially created are eligible for garbage collection?

a) 0

b) 1

c) 2

d) It depends on JVM activity.

Ans: C) 2

77. Which of the following is a way to make an object eligible for garbage collection?

a) Reassigning a reference to another object

b) Declaring the object as static

c) Using the object in a method

d) Calling finalize() directly

Ans: A) Reassigning a reference to another object

78). An object `A` has a strong reference to object `B`. Object `B` has a strong reference back to object `A`. There are no other references to `A` or `B` from any active thread. What will happen during garbage collection?

a) The `finalize()` method of both objects will be called, but their memory will not be reclaimed.

b) Neither object will be garbage collected because they reference each other, creating a memory leak.

c) Both objects will be eligible for garbage collection.

d) A `StackOverflowError` will occur when the garbage collector tries to trace their references.

Ans: C) Both objects will be eligible for garbage collection.

79. When is an object considered "eligible for garbage collection" in Java?

a) When its constructor finishes execution.

b) When it is explicitly set to null by the programmer.

c) When it is no longer reachable from any active reference.

d) When the System.gc() method is explicitly called.

Ans: C) When it is no longer reachable from any active reference.

80. What is the finalize() method used for in Java?

a) To explicitly free memory of an object.

b) To perform cleanup activities before an object is garbage collected.

c) To mark an object as eligible for garbage collection.

d) To prevent an object from being garbage collected.

Ans: B) To perform cleanup activities before an object is garbage collected.

81. Why should you generally avoid relying heavily on the finalize() method for critical resource cleanup (like closing files or network connections)?
a) It's explicitly prohibited by the Java specification.
b) Its execution is not guaranteed, and its timing is unpredictable.
c) It makes the garbage collector run slower.
d) It can only be called once per application runtime.
Ans: B) Its execution is not guaranteed, and its timing is unpredictable.

82. Which of the following scenarios would make an object eligible for garbage collection?
a) I. Setting all reference variables pointing to the object to null.
b) II. Re-assigning all reference variables pointing to the object to another object.
c) III. An object becoming part of an "island of isolation" (objects referencing each other but none reachable from GC roots).
d) All of the above.
Ans: D) All of the above.

83. What is an "island of isolation" in the context of Java garbage collection?
a) A set of objects that are only reachable from other objects within the set, with no external references.
b) Objects that are created in a separate thread.
c) Objects that are declared as private.
d) Objects that reside on the stack memory.
Ans: A) A set of objects that are only reachable from other objects within the set, with no external references.

84. A junior developer explicitly calls System.gc() frequently in their code, believing it will improve performance. What is the likely outcome or best practice advice for this situation?
a) It will significantly boost performance by forcing immediate memory cleanup.
b) It might actually hinder performance due to the overhead of running the GC, and it doesn't guarantee immediate collection anyway.
c) It will only collect objects that implement finalize().
d) It is a mandatory step for releasing memory in Java.
Ans: b) It might actually hinder performance due to the overhead of running the GC, and it doesn't guarantee immediate collection anyway.

85. A Java application is experiencing OutOfMemoryError. The developer suspects memory leaks due to objects not being garbage collected. Which of the following is NOT a direct cause of objects not being garbage collected?
a) Long-lived references (e.g., objects stored in static collections that are never cleared).
b) Objects that are still reachable from a live thread.
c) Improper use of the new keyword.
d) finalize() methods taking too long to execute.
Ans: finalize() methods taking too long to execute.

Session 10: Methods, this keyword, Static Members, Scope, JDK Tools

86. If a Java application is running slow, which tool from the JDK would be most useful for profiling and identifying performance bottlenecks?
a) ‘javac’
b) ‘jvisualvm’ (or VisualVM)
c) ‘jar’
d) ‘javadoc’
Ans: B) ‘jvisualvm’ (or VisualVM)

87. What is "method overloading" in Java?
a) Defining a method in a subclass with the same signature as a superclass method.
b) Defining multiple methods in the same class with the same name but different parameter lists.
c) Defining a method with a very large number of parameters.
d) Calling a method multiple times from within the same class.

88. What does the this keyword refer to inside an instance method?
a) The current class definition.
b) The current object (instance) on which the method is invoked.
c) The superclass of the current object.
d) A static variable of the class.
Ans: B) The current object (instance) on which the method is invoked.

89. Which of the following statements about static methods is true?
a) They can access both static and instance (non-static) variables directly.
b) They can only access other static members directly.
c) They must be called on an object instance.
d) They can be overridden by non-static methods.
Ans: B) They can only access other static members directly.

90. What is the primary function of javadoc (a JDK tool)?
a) To compile Java source code.
b) To run Java applications.
c) To generate API documentation in HTML format from comments in source code.
d) To debug Java programs.
Ans: C) To generate API documentation in HTML format from comments in source code.

91. When an object is passed as a parameter to a method in Java, what is actually passed?
a) A copy of the object itself.
b) A copy of the object's memory address (a reference).
c) The values of all the object's instance variables.
d) Nothing; objects cannot be passed as parameters.
Ans: B) A copy of the object's memory address (a reference).

92. What is the "scope" of a local variable declared inside a method?
a) Throughout the entire class.
b) Only within the method where it is declared.
c) Only within the main method.
d) Throughout the entire package.
Ans: B) Only within the method where it is declared.

93. When an object is passed as a parameter to a method in Java, what is actually passed?
a) A copy of the object itself.
b) A copy of the object's memory address (a reference).
c) The values of all the object's instance variables.
d) Nothing; objects cannot be passed as parameters.
Ans: B) A copy of the object's memory address (a reference).

94. A Java program is frequently crashing with a StackOverflowError. Which of the following is a common cause related to method calls?
a) An infinite loop in a while statement.
b) Excessive recursive method calls without a proper base case.
c) Trying to access a private method from outside the class.
d) Insufficient heap memory allocated to the JVM.
Ans: B) Excessive recursive method calls without a proper base case.

95. Consider the following code:

Java

```
class Box {
    double width, height;
    Box(double w, double h) {
        this.width = w;
        this.height = h;
    }
    void setDimensions(double width, double height) {
        this.width = width;
        this.height = height;
    }
}
```

What is the purpose of this.width = w; and this.width = width; in the constructor and setDimensions method respectively?
a) It's optional; the code would work the same without this.
b) It explicitly distinguishes between instance variables and local/parameter variables with the same name.
c) It makes the variables static.
d) It refers to the superclass's variables.
Ans: b) It explicitly distinguishes between instance variables and local/parameter variables with the same name.

96. You have a class with a static method calculateInterest() and an instance method getCustomerName(). Which of the following calls is valid from an external class?

```
Java
MyClass obj = new MyClass();
```

a) obj.calculateInterest();
b) MyClass.calculateInterest();
c) obj.getCustomerName();
d) Both b and c are valid.
e) All a, b, and c are valid.
Ans: D) Both b and c are valid.

97. If you declare a variable inside a for loop
(e.g., for (int i = 0; i < 5; i++) { int temp = i * 2; ... }),
what is the lifetime of temp?
a) It exists throughout the entire method where the loop is.
b) It exists only for the duration of one iteration of the loop.
c) It exists until the garbage collector cleans it up.
d) Its lifetime is tied to the scope of the for loop, meaning it is created and destroyed with each iteration.
Ans: D) Its lifetime is tied to the scope of the for loop, meaning it is created and destroyed with each iteration.

98. You want to debug a Java application to find out why a specific variable is holding an unexpected value at a certain point in execution. Which JDK tool would be most helpful for this task?
a) javac
b) java
c) jdb
d) javadoc
Ans: C) jdb

Session 11: Packages, Access Control Rules, Types of Inheritance, IS-A Relationship

99. Why does Java support single inheritance for classes but achieve "multiple inheritance of type" through interfaces?
a) To simplify the language syntax.
b) To avoid the "Diamond Problem" of ambiguity with method implementations.
c) Because classes cannot have more than one parent by definition.
d) Interfaces are faster to process than classes.
Ans: B) To avoid the "Diamond Problem" of ambiguity with method implementations.

100. What is the primary purpose of Java packages?
a) To store primitive data types.
b) To manage memory allocation.
c) To organize classes and interfaces into logical groups and prevent naming conflicts.
d) To compile Java code into bytecode.
Ans: C) To organize classes and interfaces into logical groups and prevent naming conflicts.

101. Which access modifier makes a class, method, or field accessible only within the same package and by subclasses in any package?
a) private
b) protected
c) default (package-private)
d) public
Ans: B) protected

102. If Class B extends Class A, what type of relationship does this represent?
a) Has-A relationship
b) Uses-A relationship
c) Is-A relationship
d) Contains-A relationship
Ans: C) Is-A relationship

103. Which type of inheritance involves a single subclass extending a single superclass?
a) Multiple Inheritance
b) Hierarchical Inheritance
c) Multilevel Inheritance
d) Single Inheritance
Ans: D) Single Inheritance

104. Consider a class com.example.app.MyClass. What is the fully qualified name of this class?
a) MyClass
b) app.MyClass
c) com.example.app.MyClass
d) com.example.MyClass
Ans: C) com.example.app.MyClass

105. Which access modifier is the most restrictive?
a) public
b) protected
c) default
d) private
Ans: D) private

106. In Java, how can a class achieve functionality similar to "multiple inheritance of implementation"?
a) By extending multiple classes.
b) By implementing multiple interfaces.
c) By using nested classes.
d) By declaring all methods as static.
Ans: B) By implementing multiple interfaces.

107. You have two classes: PackageA.ClassX and PackageB.ClassY.
ClassX has a protected method doWork().
Can ClassY call ClassX.doWork()?
a) Yes, always.
b) No, never.
c) Only if ClassY extends ClassX.
d) Only if ClassY is in the same package as ClassX.
Ans: C) Only if ClassY extends ClassX.

108. What is the primary benefit of the IS-A relationship (inheritance) in the context of polymorphism?
a) It allows a subclass to have more private members.
b) It enables a superclass reference variable to hold an object of its subclass, allowing for dynamic method dispatch.
c) It guarantees that all methods are final.
d) It prevents method overloading.
Ans: B) It enables a superclass reference variable to hold an object of its subclass, allowing for dynamic method dispatch.

109. A project has a hierarchy: Animal -> Mammal -> Dog. This represents which type of inheritance?
a) Single Inheritance
b) Multiple Inheritance
c) Hierarchical Inheritance
d) Multilevel Inheritance
Ans: D) Multilevel Inheritance

110. You are importing classes. Which of the following import statements is equivalent to import java.util.ArrayList; and import java.util.LinkedList;?
a) import java.util.*;
b) import java.util.list.*;
c) import *;
d) import java.util.ArrayList, java.util.LinkedList;
Ans: A) import java.util.*;

111. Consider a class Parent in package com.example.core and a class Child in package com.example.sub, where Child extends Parent. If Parent has a default (package-private) method secretMethod(), can Child call secretMethod() directly?
a) Yes, because Child is a subclass.
b) No, because secretMethod() is package-private and Child is in a different package.
c) Yes, if secretMethod() is overridden in Child.
d) Only if secretMethod() is declared protected.
Ans: B) No, because secretMethod() is package-private and Child is in a different package.

Session 12: Polymorphism (Compile-time & Runtime), Abstract Classes, Final Keyword, Interfaces

112. Which form of polymorphism is demonstrated by method overloading?
a) Runtime Polymorphism
b) Dynamic Method Dispatch
c) Compile-time Polymorphism
d) Virtual Polymorphism
Ans: C) Compile-time Polymorphism

113. What is "Dynamic Method Dispatch" in Java?
a) Calling a method at compile time based on the reference type.
b) Calling a method at runtime based on the actual object type.
c) Directly calling a method using its memory address.
d) Overloading a method based on different return types.
Ans: B) Calling a method at runtime based on the actual object type.

114. Which of the following statements about abstract classes is true?
a) They can be instantiated directly using new.
b) They must contain at least one abstract method.
c) They can have constructors.
d) They cannot have concrete (non-abstract) methods.
Ans: C) They can have constructors.

115. If a variable is declared as final, it means:
a) It can be reassigned only once.
b) Its value cannot be changed after initialization.
c) It can only be accessed within the main method.
d) It is always a static variable.
Ans: B) Its value cannot be changed after initialization.

116. What is a key difference between an abstract class and an interface (prior to Java 8)?
a) An interface can have constructors, an abstract class cannot.
b) An abstract class can have instance variables and concrete methods, an interface (pre-Java 8) could not.
c) A class can only implement one interface but can extend multiple abstract classes.
d) Abstract classes support multiple inheritance; interfaces do not.
Ans: B) An abstract class can have instance variables and concrete methods, an interface (pre-Java 8) could not.

117. Which of the following is true regarding interfaces in Java 8 and above?

a) They can now have instance variables.

b) They can have default and static methods with implementations.

c) They can have private abstract methods.

d) They no longer require implementing classes to provide a body for abstract methods.

Ans: B) They can have default and static methods with implementations.

118. Consider the following code snippet:

Java

```
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound.");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks.");
    }
}

public class TestDispatch {
    public static void main(String[] args) {
        Animal myAnimal = new Dog();
        myAnimal.makeSound();
    }
}
```

What will be the output of this code, and which polymorphism concept does it illustrate?

a) Output: "Animal makes a sound."; Concept: Compile-time Polymorphism.

b) Output: "Dog barks."; Concept: Dynamic Method Dispatch (Runtime Polymorphism).

c) Output: "Dog barks."; Concept: Method Overloading.

d) Output: Compilation Error.

Ans: B) Output: "Dog barks."; Concept: Dynamic Method Dispatch (Runtime Polymorphism).

119. You need to design a system where various devices (e.g., Smartphone, Laptop, Tablet) can connect to a network. Each device has a connect() method, but the implementation differs. Additionally, some devices might also have a disconnect() method, while others might not. What is the most flexible and appropriate design using Java OOP principles to define the connect() behavior for all devices?

a) Create a concrete Device class with a connect() method.

b) Create a final Device class with a connect() method.

c) Define an interface Connectable with an abstract connect() method.

d) Use an abstract class Device with a concrete connect() method.

Ans: C) Define an interface Connectable with an abstract connect() method.

120. Why might a developer prefer an abstract class over an interface if they want to provide a common base implementation that subclasses can inherit, along with some abstract methods?
a) Because an abstract class allows for multiple inheritance.
b) Because an interface cannot have any concrete methods. (Incorrect for Java 8+)
c) Because an abstract class can have constructors, instance variables, and common method implementations that don't need to be reimplemented by every subclass.
d) Because interfaces are slower to execute.
Ans: C) Because an abstract class can have constructors, instance variables, and common method implementations that don't need to be reimplemented by every subclass.

121. If you have an interface Runnable (from java.lang) and a class MyTask implements Runnable, you can write

Runnable task = new MyTask();.
If Runnable has a method run(),
what mechanism determines which run() method is executed
when task.run() is called?

a) Static Binding
b) Compile-time Binding
c) Dynamic Method Dispatch
d) Constructor Overloading
Ans: C) Dynamic Method Dispatch

122. You have a method void process(Number num) and another method void process(Integer num). If you call process(new Integer(5)), which method will be invoked?
a) void process(Number num)
b) void process(Integer num)
c) Both will be called.
d) Compilation Error.
This scenario illustrates which type of polymorphism?
e) Runtime Polymorphism
f) Compile-time Polymorphism
Ans: B) void process(Integer num) then
F) Compile-time Polymorphism.

Session 13 & 14: Inner Classes & Exception Handling

123. Which exception is thrown when java is out off memory?
a) MemoryError
b) OutOfMemoryError
c) MemoryOutOfBoundsException
d) MemoryFullException
Ans : B) OutOfMemoryError

124. Which type of inner class can be declared static?
a) Member Inner Class
b) Local Inner Class
c) Anonymous Inner Class
d) Static Nested Class
Ans: D) Static Nested Class

125. Which type of inner class is defined inside a method or a block?
a) Member Inner Class
b) Local Inner Class
c) Static Nested Class
d) Anonymous Inner Class
Ans: B) Local Inner Class

126. An anonymous inner class typically is used when:
a) You need to create multiple instances of a class.
b) You need a class for single, specific use.
c) You want to hide the class definition completely.
d) You want to access static members of the outer class.
Ans: B) You need a class for single, specific use.

127. In Java, what is the superclass of all exceptions and errors?
a) Throwable
b) Exception
c) Error
d) RuntimeException
Ans: A) Throwable

128. Which keyword is used to explicitly throw an exception from a method?
a) catch
b) throws
c) throw
d) finally
Ans: C) throw

129. Which type of exception is typically checked at compile time (i.e., must be declared or handled)?

a) RuntimeException
b) Error
c) IOException
d) NullPointerException

Ans: C) IOException

130. What is the purpose of the finally block in a try-catch-finally construct?

a) It executes only if an exception occurs.
b) It executes only if no exception occurs.
c) It always executes, regardless of whether an exception occurs or not (unless the JVM exits).
d) It is used to declare custom exceptions.

Ans: C) It always executes, regardless of whether an exception occurs or not (unless the JVM exits).

131. Which of the following is true about Error in Java?

a) Errors are usually recoverable and should be caught by applications.
b) Errors indicate serious problems that applications should not try to catch.
c) Errors are subclass of RuntimeException.
d) Errors are typically handled using throws clause.

Ans: B) Errors indicate serious problems that applications should not try to catch.

132. What happens if an exception is thrown in a try block, and there is no matching catch block?

a) The program continues execution normally.
b) The finally block is skipped.
c) The exception is propagated up the call stack to the calling method.
d) A compile-time error occurs.

Ans: C) The exception is propagated up the call stack to the calling method.

133. Consider the following code structure:

Java

```
public class OuterClass {
    private int outerX = 10;
    class InnerClass {
        public void display() {
            System.out.println("Outer x: " + outerX);
        }
    }
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner = outer.new InnerClass();
        inner.display();
    }
}
```

133.1 Why is inner.display() able to access outerX even though outerX is private in OuterClass?
a) Because InnerClass is declared static.
b) Because inner classes have special access privileges to all members (including private) of their enclosing outer class.
c) Because outerX is a primitive type.
d) This code would result in a compilation error.
Ans: B) Because inner classes have special access privileges to all members (including private) of their enclosing outer class.

134. You need to implement a callback listener for a GUI button click, where the listener logic is very specific to that single button and won't be reused elsewhere. Which type of inner class is generally the most concise and appropriate for this scenario?
a) Member Inner Class
b) Static Nested Class
c) Local Inner Class
d) Anonymous Inner Class
Ans: D) Anonymous Inner Class

135. What happens if a constructor throws an exception?
a) The object is partially created and usable.
b) The object is not created, and the exception propagates.
c) The program terminates immediately.
d) The constructor is retried automatically.
Ans: B) The object is not created, and the exception propagates.

136. Consider the following code:

Java

```
public class TestException {
    public static void main(String[] args) {
        try {
            System.out.println("A");
            int result = 10 / 0;
            System.out.println("B");
        } catch (ArithmeticException e) {
            System.out.println("C");
        } finally {
            System.out.println("D");
        }
        System.out.println("E");
    }
}
```

What will be the output of this program?
a) A B C D E
b) A C D E
c) A B D E
d) A D E
Ans: B) A C D E

137. You are writing a method that reads data from a file. This method might encounter a FileNotFoundException (a checked exception). What are your two main options for handling this exception?
a) Catch the exception using a try-catch block, or declare it using throws in the method signature.
b) Ignore the exception; it will be handled by the JVM.
c) Convert it into a RuntimeException using throw.
d) Both a and c are valid.
Ans: A) Catch the exception using a try-catch block, or declare it using throws in the method signature.

138. When creating a custom exception class, which class should it typically extend if it represents an error from which the application *might* be able to recover (e.g., invalid user input)?
a) java.lang.Error
b) java.lang.RuntimeException
c) java.lang.Exception
d) java.lang.Throwable
Ans: C) java.lang.Exception

139. What is the key difference in behavior between throw new MyException(); and throws MyException?
a) throw is used to declare that a method might throw an exception, while throws is used to explicitly create and propagate an exception.
b) throw is used to explicitly create and propagate an exception, while throws is used to declare that a method might throw an exception.
c) throw is for checked exceptions, throws is for unchecked exceptions.
d) throw is used in a try block, throws is used in a catch block.
Ans: B) throw is used to explicitly create and propagate an exception, while throws is used to declare that a method *might* throw an exception.

140. Consider a method readFile() that declares throws IOException. If another method, processFile(), calls readFile(), what must processFile() do?
a) processFile() must also declare throws IOException.
b) processFile() must catch IOException.
c) processFile() must either catch IOException or declare throws IOException.
d) processFile() can ignore IOException because readFile() already handles it.
Ans: C) processFile() must either catch IOException or declare throws IOException.

141. Which type of exception typically indicates a programming bug (e.g., NullPointerException, ArrayIndexOutOfBoundsException) and does not need to be explicitly declared or caught?
a) Checked Exception
b) Compile-time Exception
c) Unchecked Exception (Runtime Exception)
d) System Exception
Ans: C) Unchecked Exception (Runtime Exception)

142. You have nested try-catch blocks. If an exception is thrown in the inner try block, and there is no matching catch in the inner block, where does the JVM look for a handler next?
a) In the finally block of the inner try.
b) In the catch blocks of the outer try block.
c) The program terminates immediately.
d) It searches for a main method.
Ans: B) In the catch blocks of the outer try block.

143. What is the purpose of the printStackTrace() method of an exception object?
a) To return the exception message.
b) To print the name of the exception.
c) To print the stack trace (call stack) from where the exception occurred to the console.
d) To log the exception to a file.
Ans: C) To print the stack trace (call stack) from where the exception occurred to the console.

144. Is it possible for a finally block to NOT execute after a try block? If so, when?
a) No, a finally block always executes.
b) Yes, if System.exit() is called from within the try or catch block.
c) Yes, if the try block contains an infinite loop.
d) Both b and c.
Ans: d) Both b and c.

Session 15 & 16: Java Core API & Collections Framework

145.) You need a collection that stores elements in the order they were inserted and also prevents duplicate elements. Which `Collection` implementation should you choose?
a) LinkedHashSet
b) TreeSet
c) ArrayList
d) HashSet
Ans: A) LinkedHashSet

146. Which of the following is the root class of all classes in Java's inheritance hierarchy?
a) Class
b) Object
c) System
d) Root
Ans: B) Object

147. What does the hashCode() method (from Object class) typically return?
a) A unique identifier for every object.
b) An integer representation of the object's memory address.
c) An integer hash code value for the object.
d) The name of the object's class.
Ans: C) An integer hash code value for the object.

148. Which java.lang class provides static methods for mathematical operations like sqrt(), abs(), and random()?
a) System
b) Number
c) Math
d) Runtime
Ans: C) Math

149. What is the process of automatically converting a primitive type (e.g., int) into its corresponding wrapper class object (e.g., Integer) called?
a) Unboxing
b) Casting
c) Autoboxing
d) Boxing
Ans: C) Autoboxing

150. Which of the following is true about String objects in Java?
a) They are mutable.
b) They are stored in the stack memory.
c) They are immutable.
d) They can be modified using append() method.
Ans: C) They are immutable.

151. Where are String literals typically stored in Java for optimization?
a) Heap memory
b) Stack memory
c) String Pool (part of Heap in recent JVMs)
d) Permanent Generation
Ans: C) String Pool (part of Heap in recent JVMs)

152. What is the main difference between StringBuffer and StringBuilder?
a) StringBuffer is faster, StringBuilder is slower.
b) StringBuffer is thread-safe (synchronized), StringBuilder is not.
c) StringBuilder is thread-safe (synchronized), StringBuffer is not.
d) StringBuffer uses less memory than StringBuilder.
Ans: B) StringBuffer is thread-safe (synchronized), StringBuilder is not.

153. Which method of the System class is commonly used to print output to the console?
a) System.in.read()
b) System.out.println()
c) System.err.print()
d) System.exit()
Ans: B) System.out.println()

154. Which of the following interfaces is the root of the Java Collections Framework hierarchy?
a) List
b) Set
c) Collection
d) Map
Ans: C) Collection

155. Which List implementation is generally preferred when frequent element insertions or deletions occur in the middle of the list?
a) ArrayList
b) LinkedList
c) Vector
d) Stack
Ans: B) LinkedList

156. Which Set implementation maintains the insertion order of elements?
a) HashSet
b) TreeSet
c) LinkedHashSet
d) EnumSet
Ans: C) LinkedHashSet

157. Which Map implementation stores key-value pairs in a sorted order based on the natural ordering of keys (or a provided Comparator)?

a) HashMap
b) LinkedHashMap
c) TreeMap
d) Hashtable

Ans: C) TreeMap

158. Which of the following collections ensures uniqueness of elements and does not maintain any specific order?

a) ArrayList
b) LinkedList
c) HashSet
d) Vector

Ans: C) HashSet

159. What is the primary difference between HashMap and Hashtable?

a) HashMap allows null keys/values, Hashtable does not.
b) Hashtable is synchronized, HashMap is not.
c) HashMap is older than Hashtable.
d) Both a and b.

Ans: D) Both a and b.

160. Which List implementation is synchronized by default, making it thread-safe but potentially slower for single-threaded operations?

a) ArrayList
b) LinkedList
c) Vector
d) CopyOnWriteArrayList

Ans: C) Vector

161. Consider the following code:

```
Java
String s1 = "Hello";
String s2 = "Hello";
String s3 = new String("Hello");
```

Which of the following comparisons will evaluate to true?

a) s1 == s3
b) s1.equals(s3)
c) s1 == s2
d) Both b and c.

Ans: D) Both b and c.

162. You need to store a list of customer orders. Orders are added sequentially, and sometimes an order needs to be removed from the beginning or end. You also need fast iteration through all orders. Which List implementation would be most appropriate?
a) ArrayList
b) LinkedList
c) Vector
d) Stack
Ans: B) LinkedList

163. Why is it generally recommended to use StringBuilder over StringBuffer in single-threaded environments?
a) StringBuilder consumes less memory.
b) StringBuilder provides better performance because it doesn't incur the overhead of synchronization.
c) StringBuffer is deprecated.
d) StringBuilder allows for immutable string operations.
Ans: B) StringBuilder provides better performance because it doesn't incur the overhead of synchronization.

164. You are storing a collection of unique email addresses. You want to iterate through them in the order they were added, and fast lookups are important. Which collection should you choose?
a) HashSet
b) TreeSet
c) LinkedHashSet
d) ArrayList
Ans: C) LinkedHashSet

165. When comparing two objects obj1 and obj2 for equality, why is it usually better to override the equals() method (from Object class) rather than using ==?
a) == only works for primitive types.
b) equals() compares memory addresses, while == compares content.
c) == compares memory addresses (for objects), while equals() can be customized to compare object content based on business logic.
d) equals() is a static method, so it's safer.
Ans: C) == compares memory addresses (for objects), while equals() can be customized to compare object content based on business logic.

166. What is Autoboxing and Unboxing primarily used for?
a) To convert String to int and vice versa.
b) To enable seamless conversion between primitive types and their corresponding wrapper objects, especially when interacting with collections.
c) To perform arithmetic operations on objects.
d) To manage memory automatically.
Ans: B) To enable seamless conversion between primitive types and their corresponding wrapper objects, especially when interacting with collections.

167. You need to store employee records (Employee ID to Employee Object). You need fast retrieval by ID, and the records should be sorted by Employee ID. Which collection should you use?

a) LinkedHashMap
b) HashMap
c) TreeMap
d) ArrayList

Ans: C) TreeMap

168. Consider the code:

Integer i1 = 100; Integer i2 = 100; Integer i3 = 200; Integer i4 = 200;
Which of the following will be true, considering Java's Integer caching behavior (for values -128 to 127)?

a) i1 == i2 is false, i3 == i4 is false
b) i1 == i2 is true, i3 == i4 is true
c) i1 == i2 is true, i3 == i4 is false
d) i1 == i2 is false, i3 == i4 is true

Ans: C) i1 == i2 is true, i3 == i4 is false

169. When should you prefer using ArrayList over LinkedList?

a) When you perform frequent insertions/deletions in the middle of the list.
b) When you mostly perform random access (getting elements by index) and iterate through the list.
c) When you need synchronized access to the list.
d) When memory usage is a critical concern.

Ans: B) When you mostly perform random access (getting elements by index) and iterate through the list.

170. You are building an application where multiple threads will concurrently modify a shared List. Which of the following is a good choice for thread safety in this scenario?

a) ArrayList (without external synchronization)
b) LinkedList (without external synchronization)
c) Vector
d) TreeMap

Ans: C) Vector

Session 17, 18 & 19 : -

1. Which of the following is not an advantage of Generics in Java ?
1. Compile time safety.
2. Type casting is not needed.
3. **Can store more than one type of object.**
4. Type safety is ensured.

---

5. Question

```
class Generic {

public static void main ( String args [ ] )

{

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

list.add(100);

list.add(200);

list.add(300);

String a = list.get(2);

System.out.println(a);

}

}
```

1. 100
2. **300**
3. 200
4. Compilation error.

---

5. Question

```
 class HashMap {
public static void main ( String args [ ] )
{
Map <Integer,String> map = new HashMap<Integer,String> ();
map.add(1,"one");
map.add(2."two");
map.add(3."three");
Set<Map.Entry<Integer,String>> set=map.entrySet();
Iterator<Map.Entry<Integer,String>> itr=set.iterator();
while(itr.hasNext()){
Map.Entry e=itr.next();
System.out.println(e.getKey()+" "+e.getValue());
}
}
```

1. **1 one 2 two 3 three**
2. 1 2 3
3. one two three

4. Compilation error.

---

5. Which of the following classes use Generic Classes ?

1. Interface

2. HashSet
3. Abstract
4. Constructor

---

5. Question

```
class Generic < G >
{
G obj;
Generic  ( G obj )
{
this.obj = obj;
}
public G getObject ( ) {
return this.obj;
}
}
class Main {
public static void main ( String args [ ] )
{
Generic < Integer> obj1 = new Generic <Integer> (10);
System.out.println(obj1.getObject());

Generic <String> obj2 = new Generic <String> (“DataFlair”);
System.out.println(obj2.getObject());
}
}
```

- 10
- DataFlair.
- **Both a and b**
- Compilation error.

---

6. Question

```
class StringSample <S> {
S obj;
StringSample( S obj )
{
this.obj = obj;
}
void print( ) {
return this.obj;
}
}
class Main {
public static void main ( String args [ ] )
{
Generic <Integer> object = new Generic <Integer> (“Ten”);
System.out.println(object.print());
}
}
```

1. 10
2. Ten
3. object.print()
4. **Compilation error.**

---

5. Which of the following methods are used to restrict the number of types of data for a class ?

1. Type casting.
2. Binding
3. **Bounded parameters.**
4. Marshalling.

---

5. Question

```
class Demo < O , T >
O = object1;
T = object2;
Demo ( object1 , object2 )
{
this.object1 = object1;
this.object2 = object2;
 }
public void print ( object1 , object2 )
{
System.out.println( object1 + object2 );
}
}
class Main {
public static void main ( String args [ ] )
{
Demo < Integer , String > obj = new Demo < Integer , String > ( “Two” , 2 );
obj.print();
}
}
```

1. Error in object creation.
2. **Parameters are changed and it raises an exception.**
3. Object should be passed while invoking the method.
4. No errors in the program.

---

5. Question

```
class GenericSample {
void method ( E element  )
{
System.out.println(element.getclass().getName());
}
}
public static void main ( String args [ ] )
{
method(“DataFlair”);
}
}
```

1. DataFlair
2. element
3. java.lang.String
4. java.lang.Integer.

---

5. Which of the following is possible in Generic classes ?
1. Can have two objects
2. Can pass two different parameters.
3. Can have a null object.
4. None of these.

---

5. Question

```
class Error {
Error( obj ) {
this.obj = obj;
}
void display( obj )
{
System.out.println(obj);
}
}
class Main {
public static void main ( String args [ ] )
{
Error < String > object = new Error < String > (“10”);
object.display();
}
}
```

1. Display method doesn't return any value.
2. Error in object declaration.
3. 10 cannot be passed as a string value.
4. No errors in the program 10 are displayed as output.

---

5. Question

```
class DemoClass {
public <T> void genericsMethod(T data) {
      System.out.println(data);
    }
}
class Main {
public static void main(String args [ ] )
DemoClass demo = new DemoClass();
demo.<String>genericsMethod(“DataFlair”);
}
}
```

1. DataFlair
2. demo
3. data
4. Compilation error.

---

5. Which of the following is used to display the package in generic classes ?
1. element.getClass();
2. element.getPackage();
3. element.getImportFile();

4. element.get();

5. Question

```
class Arraylist {
public static void main ( String args [ ] )
{
ArrayList <> a = new ArrayList<>  ();
a.add("String");
a.add("Integer");
a.add(2);
System.out.println(a.get(2));
}
}
```

1. String
2. Integer
3. **Compilation error**
4. Compiled but no output.

6. Question

```
class Sample < S , I >
S = obj1;
I = obj2;
Sample ( obj1 , obj2 )
{
this.obj1 = obj1;
this.obj2 = obj2;
}
public void text ( obj1 , obj2 )
{
System.out.println(obj1);
System.out.println(obj2);
}
}
class Main {
public static void main ( String args [ ] )
```

```
{
Sample < String , Integer > s = new Sample < String , Integer > ( “DataFlair” , 1 );
s.text();
}
}
```

5. DataFlair
6. 1
7. DataFlair 1
8. **Compilation error.**

---

7. Which of these Exception handlers cannot be type parameterized?
   a) catch
   b) throw
   c) throws
   d) **all of the mentioned**
8. Which of the following cannot be Type parameterized?
   a) **Overloaded Methods**
   b) Generic methods
   c) Class methods
   d) Overriding methods

---

9. **What is the primary benefit of using generics in Java?**
   A. Code readability
   B. Runtime type checking
   C**. Compile-time type safety**
   D. Simplified syntax

---

10. **Which of the following is a valid generic method declaration?**
    A. public void method(T t)
    B. public <T> void method(T t)
    C. public void <T> method(T t)
    D. public void method<T>(T t)

---

11. **What does the wildcard ? extends Number signify in generics?**
    A. Accepts any type
    B. Accepts Number and its superclasses
    C. Accepts Number and its subclasses
    D. Accepts only Number

---

12. **Can you create an instance of a generic type parameter T using new T()?**
    A. Yes, always
    B. No, because of type erasure
    C. Only if T has a no-argument constructor
    D. Only in Java 8 and above

---

13. **Which interface must a class implement to define a natural ordering?**
    A. Comparator
    B. Comparable
    C. Iterable
    D. Serializable

---

14. Which of the following is a correct way to declare a generic class in Java?
A. class MyClass<T> {}
B. class <T> MyClass {}
C. generic class MyClass<T> {}
D. class MyClass generic<T> {}

---

15. **What is the result of using raw types (non-generic usage) in generic classes?**
A. Improves type safety
B. Causes compilation error
C. Results in unchecked warnings
D. Prevents method overloading

---

16. **Which generic type declaration is used when any type is acceptable?**
A. <?>
B. <T>
C. <Any>
D. <Object>

---

17. **What does List<? super Integer> mean?**
A. List of Integer only
B. List of subclasses of Integer
C. List of superclasses of Integer
D. List of any type

---

18. **In which of the following can we NOT use wildcards?**
A. Method arguments
B. Method return types
C. Generic class declarations
D. Variable declarations

---

19. **What does the wildcard <? extends T> allow you to do?**
A. Add objects of type T and its subtypes to the collection
B. Read objects of type T and its subtypes from the collection
C. Add and read all object types
D. Prevent all operations

20. **Which wildcard allows writing into a generic collection?**
A. <? extends T>
B. <? super T>
C. <?>
D. <T>

21. **What does the wildcard <?> mean in Java Generics?**
A. Accepts only Object
B. Accepts a fixed known type
C. Accepts any type (unknown type)
D. Accepts primitive types only

22. **In inheritance, which of the following is TRUE about generics?**
A. List<Object> is a supertype of List<String>
B. List<Object> can store any data, including List<String>
C. List<String> can be assigned to List<Object>
D. Generics are invariant

23. **Why can't you add elements to a List<? extends Number>?**
A. Because it's immutable
B. Because the compiler doesn't know the exact subtype of Number
C. Because Number is an abstract class
D. Because wildcard syntax only allows read

24. **What is the correct way to ensure a method works with a list of any subclass of Shape?**
A. void draw(List<Shape> shapes)
B. void draw(List<?> shapes)
C. void draw(List<? extends Shape> shapes)
D. void draw(List<Object> shapes)

25. **Which of the following is ordered and allows duplicate elements?**
A. HashSet
B. TreeSet
C. **ArrayList**
D. HashMap

26. **Which of these classes maintains insertion order and disallows duplicates?**
A. HashSet
B. TreeSet
C. **LinkedHashSet**
D. ArrayList

27. **Which class implements a Map that sorts its keys using natural ordering?**
A. HashMap
B. LinkedHashMap
C. **TreeMap**
D. Hashtable

28. **What is the default sorting mechanism in Java Collections?**
A. Based on hashcodes
B. **Natural ordering (Comparable)**
C. Reverse ordering
D. Based on memory address

29. **Which interface must be implemented to define custom sorting logic?**
A. Serializable
B. **Comparator**
C. Iterable
D. Comparable

30. **What happens if compareTo() returns a positive number?**
A. Current object is less than the argument
B. Current object is equal to the argument
C. Current object is greater than the argument
D. An exception is thrown

31. **Which collection class allows storing key-value pairs and maintains insertion order?**
A. TreeMap

B. HashMap
C. **LinkedHashMap**
D. Hashtable

32. **What is returned by Map.get(key) if the key is not found?**
A. Throws NullPointerException
B. 0
C. **null**
D. Compilation error

33. **Which of the following is true about TreeSet?**
A. Allows null values
B. Is not sorted
C. Is thread-safe
D. Maintains elements in natural order

34. **To sort a list of custom objects by name in descending order, which is correct?**
A. Use Collections.reverseOrder()
B. Implement Comparator and override compare()
C. Use Comparable and sort manually
D. Sorting is not possible

Session 20 & 21 : -

* Multithreaded Programming :-

1. Which of the following is true about threads in Java?

A) Threads share the same memory space
B) Threads are independent processes
C) Threads have different memory allocations
D) Java does not support multithreading
✅ **Answer:** A

---

2. Which method is used to start a thread execution?

A) run()
B) start()
C) init()
D) execute()
✅ **Answer:** B

---

3. Which of the following is not a valid thread state in Java?

A) New
B) Runnable
C) Waiting
D) Finished
✅ **Answer:** D

---

4. What is the default priority of a thread in Java?

A) 0
B) 1
C) 5
D) 10
✅ **Answer:** C

---

5. Which interface must be implemented to create a thread using Runnable?

A) Thread
B) Runnable
C) Callable
D) Executor
✅ **Answer:** B

---

6. What happens if you call run() instead of start()?

A) It throws an exception
B) It starts a new thread

C) It runs in the current thread
D) Nothing happens
✅ **Answer:** C

---

7. Which method can be used to stop the execution of a thread for some time?

A) wait()
B) sleep()
C) pause()
D) yield()
✅ **Answer:** B

---

8. Which method is used to force the current thread to wait for another thread to finish?

A) join()
B) wait()
C) notify()
D) suspend()
✅ **Answer:** A

---

9. Which class is used to create a thread in Java apart from Runnable interface?

A) ThreadGroup
B) Process
C) Thread
D) ExecutorService
✅ **Answer:** C

---

10. What is the result of calling thread.setDaemon(true)?

A) It makes the thread high priority
B) It kills the thread
C) It marks the thread as a daemon thread
D) It starts the thread
✅ **Answer:** C

---

11. Which method is called automatically when a thread starts executing?

A) init()
B) run()
C) execute()
D) main()
✅ **Answer:** B

---

12. Which of the following is true about Daemon threads?

A) They are low priority threads
B) They stop when the main thread ends
C) They are created using ThreadGroup
D) They handle database operations
✅ **Answer:** B

---

13. What will happen if an exception is thrown inside a thread's run method and not handled?

A) JVM crashes
B) The thread stops
C) The process stops
D) All threads stop
✅ **Answer:** B

---

14. Which keyword is used to prevent thread interference?

A) static
B) final
C) synchronized
D) volatile
✅ **Answer:** C

---

15. Which class provides thread pool management in Java?

A) Runnable
B) ExecutorService
C) ThreadGroup
D) ThreadLocal
✅ **Answer:** B

---

16. Which method is used to suggest the thread scheduler to switch context?

A) yield()
B) sleep()
C) wait()
D) notify()
✅ **Answer:** A

---

17. Which interface allows you to return a result from a thread?

A) Runnable
B) Thread
C) Callable
D) FutureTask
✅ **Answer:** C

18. The thread scheduler uses which algorithm in Java by default?

A) FIFO
B) Round Robin
C) Preemptive
D) Platform-dependent
✅ **Answer:** D

---

19. What happens when a thread completes execution?

A) It goes into blocked state
B) It dies
C) It is suspended
D) It is paused
✅ **Answer:** B

---

20. Which of the following is not recommended for stopping a thread in Java?

A) interrupt()
B) flag checking
C) stop()
D) wait()
✅ **Answer:** C

---

21. What is the output when two threads update the same data concurrently without synchronization?

A) Data is always consistent
B) Data might become corrupt
C) Threads block each other
D) Compiler error
✅ **Answer:** B

---

*Multitasking: Process-Based vs Thread-Based

1. What is multitasking in an operating system?

A) Executing multiple tasks one after another
B) Running multiple tasks simultaneously
C) Executing a single task repeatedly
D) Running one process in multiple computers
✅ **Answer:** B

---

2. Which of the following best describes process-based multitasking?

A) Each process runs in the same memory space
B) Threads share memory with each other
C) Each process runs independently with its own memory
D) Threads are used for faster execution
✅ **Answer:** C

---

3. Thread-based multitasking is also known as:

A) Heavyweight multitasking
B) Lightweight multitasking
C) Monotasking
D) Parallel computing
✅ **Answer:** B

---

4. In Java, which class supports thread-based multitasking?

A) Runtime
B) Thread
C) Process
D) Object
✅ **Answer:** B

---

5. Which multitasking approach consumes more memory?

A) Thread-based
B) Process-based
C) Both consume same
D) None of the above
✅ **Answer:** B

---

6. Which of the following is not true about threads?

A) Threads share process memory
B) Threads have their own stack
C) Threads execute sequentially
D) Threads are lightweight
✅ **Answer:** C

---

7. Which is faster: process-based multitasking or thread-based multitasking?

A) Process-based
B) Thread-based
C) Both are equal
D) Depends on the processor
✅ **Answer:** B

---

8. Which multitasking type provides better isolation and security?

A) Thread-based
B) Process-based
C) Hybrid-based
D) Task-based
✅ **Answer:** B

---

9. Which of the following is an advantage of thread-based multitasking?

A) Better memory isolation
B) Independent failure tolerance
C) Less overhead and faster context switching
D) Needs more CPU
✅ **Answer:** C

---

10. In process-based multitasking, each process:

A) Shares memory with all other processes
B) Has its own memory and resources
C) Runs within a thread
D) Must execute in sequence
✅ **Answer:** B

---

11. What is the overhead of creating new processes compared to threads?

A) Much lower
B) Slightly higher
C) Much higher
D) Same
✅ **Answer:** C

---

12. Which Java class is used to execute external processes?

A) Process
B) Thread
C) ExecutorService

D) ThreadGroup
✅ **Answer:** A

---

13. In thread-based multitasking, what is typically shared among threads?

A) Code and data segments
B) Stack
C) Registers
D) Cache
✅ **Answer:** A

---

14. If one process crashes, others are:

A) Always terminated
B) Not affected
C) Automatically restarted
D) Suspended
✅ **Answer:** B

---

15. In which multitasking is context switching more expensive?

A) Thread-based
B) Process-based
C) Both are equal
D) Depends on RAM
✅ **Answer:** B

---

16. Which of these is more suitable for isolated applications (e.g., browsers, databases)?

A) Thread-based
B) Process-based
C) Object-based
D) Module-based
✅ **Answer:** B

---

17. In thread-based multitasking, what leads to potential data corruption?

A) Stack overflow
B) CPU scheduling
C) Shared memory without synchronization
D) Priority scheduling
✅ **Answer:** C

---

18. Which provides better fault tolerance in case of errors?

A) Threads
B) Processes
C) Both
D) None
✅ **Answer:** B

---

19. Multithreaded programs in Java improve performance primarily through:

A) Increased memory
B) Synchronization
C) Parallel execution and shared resources
D) Network buffering
✅ **Answer:** C

---

20. Which of the following is a disadvantage of process-based multitasking?

A) High speed
B) High memory usage
C) Thread safety
D) Simplified debugging
✅ **Answer:** B

---

21. Process-based multitasking is more commonly used in:

A) Java applications
B) Web browsers
C) Operating system-level programs
D) Game development
✅ **Answer:** C

---

* Thread State Transitions

1. How many thread states are defined in Java?

A) 3
B) 4
C) 5
D) 6
✅ **Answer:** D
(*NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED*)

---

2. Which method is used to move a thread from RUNNABLE to TIMED_WAITING?

A) wait()
B) sleep()
C) notify()
D) start()
✅ **Answer:** B

---

3. Which thread state indicates that a thread has been created but not yet started?

A) RUNNABLE
B) NEW
C) BLOCKED
D) WAITING
✅ **Answer:** B

---

4. A thread enters the TERMINATED state when:

A) It is blocked
B) It sleeps
C) It completes execution
D) It is notified
✅ **Answer:** C

---

5. Calling start() on a thread moves it from:

A) NEW → RUNNABLE
B) NEW → BLOCKED
C) NEW → TERMINATED
D) WAITING → RUNNABLE
✅ **Answer:** A

---

6. Which method causes a thread to go from RUNNABLE to WAITING?

A) wait()
B) join(long timeout)
C) sleep()
D) notify()
✅ **Answer:** A

---

7. Which state means the thread is eligible to run but is not currently running?

A) NEW
B) RUNNABLE
C) WAITING
D) TERMINATED
✅ **Answer:** B

---

8. What happens when join() is called on a thread?

A) It is killed
B) The thread waits for another thread to finish
C) It starts execution
D) It goes to sleep
✅ **Answer:** B

---

9. Which thread state occurs when a thread is trying to enter a synchronized block that is already held by another thread?

A) WAITING
B) BLOCKED
C) RUNNABLE
D) TERMINATED
✅ **Answer:** B

---

10. A thread moves to TIMED_WAITING when which method is called?

A) wait()
B) notify()
C) join(1000)
D) yield()
✅ **Answer:** C

---

11. Which state is entered when the thread's run() method completes?

A) NEW
B) RUNNABLE
C) TERMINATED
D) BLOCKED
✅ **Answer:** C

---

12. When a thread is waiting for another thread to notify it, it is in:

A) RUNNABLE
B) WAITING
C) BLOCKED
D) TIMED_WAITING
✅ **Answer:** B

---

13. The thread transitions to BLOCKED when:

A) It calls join()
B) It tries to enter a locked synchronized block

C) It finishes execution
D) It sleeps
✅ **Answer:** B

---

14. Which of the following will move a thread from WAITING to RUNNABLE?

A) notify()
B) sleep()
C) yield()
D) start()
✅ **Answer:** A

---

15. Which thread state can be transitioned to RUNNABLE after a time period?

A) WAITING
B) BLOCKED
C) TIMED_WAITING
D) TERMINATED
✅ **Answer:** C

---

16. Which state transition is not possible directly?

A) NEW → TERMINATED
B) BLOCKED → RUNNABLE
C) TIMED_WAITING → RUNNABLE
D) NEW → RUNNABLE
✅ **Answer:** A

---

17. Thread.sleep() moves the thread to which state?

A) RUNNABLE
B) WAITING
C) TIMED_WAITING
D) TERMINATED
✅ **Answer:** C

---

18. What is the difference between WAITING and TIMED_WAITING?

A) WAITING is for infinite time; TIMED_WAITING is for fixed time
B) TIMED_WAITING is a blocking state
C) WAITING is not a real state
D) Both are the same
✅ **Answer:** A

---

19. What causes a thread in BLOCKED state to resume RUNNABLE?

A) Time expires
B) It completes execution
C) Lock is released
D) notify() is called
✅ **Answer:** C

---

20. Which method does not affect thread state transitions?

A) start()
B) run()
C) sleep()
D) join()
✅ **Answer:** B
(*Calling run() directly does not create a new thread or change its state.*)

---

21. Which is the correct transition when thread.start() is called?

A) NEW → RUNNABLE → TERMINATED
B) RUNNABLE → BLOCKED → NEW
C) NEW → WAITING
D) RUNNABLE → NEW
✅ **Answer:** A

---

*The Thread class & its API

1. Which package contains the Thread class in Java?

A) java.util
B) java.lang
C) java.io
D) java.thread
✅ **Answer:** B

---

2. Which of these is the correct way to create a thread using the Thread class?

A) Thread t = new Runnable();
B) Runnable r = new Thread();
C) Thread t = new Thread();
D) Thread t = new Thread(runnableInstance);
✅ **Answer:** D

---

3. Which method in the Thread class is used to start a thread?

A) run()
B) begin()
C) start()

D) init()
✅ **Answer:** C

---

4. What does the run() method of the Thread class contain?

A) Code for thread creation
B) Initialization code
C) Code to be executed by the thread
D) JVM entry point
✅ **Answer:** C

---

5. What will happen if you call run() instead of start()?

A) New thread starts
B) Nothing happens
C) It throws an exception
D) Code runs in the current thread, not a new one
✅ **Answer:** D

---

6. Which method can you override when extending the Thread class?

A) start()
B) stop()
C) run()
D) wait()
✅ **Answer:** C

---

7. How do you set the priority of a thread?

A) setLevel(int)
B) setPriority(int)
C) definePriority(int)
D) priority(int)
✅ **Answer:** B

---

8. What is the default priority of a thread?

A) 0
B) 1
C) 5
D) 10
✅ **Answer:** C

---

9. Which constants are defined in Thread class for priority?

A) MIN_PRIORITY, MID_PRIORITY, MAX_PRIORITY
B) LOW_PRIORITY, NORMAL_PRIORITY, HIGH_PRIORITY
C) MIN_PRIORITY, NORM_PRIORITY, MAX_PRIORITY
D) NONE
✅ **Answer:** C

---

10. Which method can be used to pause a thread temporarily?

A) wait()
B) yield()
C) stop()
D) suspend()
✅ **Answer:** B

---

11. Which method is used to stop the thread for a specific period?

A) pause()
B) wait()
C) sleep()
D) delay()
✅ **Answer:** C

---

12. What does the isAlive() method return?

A) true if the thread has started but not yet dead
B) false always
C) true only before start()
D) false after join()
✅ **Answer:** A

---

13. Which method is used to wait for a thread to finish execution?

A) sleep()
B) yield()
C) join()
D) notify()
✅ **Answer:** C

---

14. How can you check the current executing thread?

A) getThread()
B) currentThread()
C) getRunningThread()
D) getThreadName()
✅ **Answer:** B
(Thread.currentThread())

---

15. What will Thread.sleep(1000) do?

A) Puts the thread to sleep forever
B) Sleeps the thread for 1000 milliseconds
C) Sleeps the thread for 1000 nanoseconds
D) Terminates the thread
✅ **Answer:** B

---

16. What exception must be handled when using sleep()?

A) RuntimeException
B) IOException
C) InterruptedException
D) SleepException
✅ **Answer:** C

---

17. Which method can be used to interrupt a sleeping thread?

A) stop()
B) kill()
C) notify()
D) interrupt()
✅ **Answer:** D

---

18. What does the setDaemon(true) method do?

A) Starts the thread
B) Kills the thread
C) Marks it as background thread
D) Makes thread high priority
✅ **Answer:** C

---

19. What is the significance of daemon threads?

A) They are long-running
B) They prevent JVM from shutting down
C) JVM exits when only daemon threads remain
D) Used only in networking
✅ **Answer:** C

---

20. What does getName() return for a thread?

A) Thread ID
B) Thread's class name
C) Name of the thread

D) Thread state
✅ **Answer:** C

---

21. Which method sets the name of a thread?

A) name()
B) setName()
C) threadName()
D) assignName()
✅ **Answer:** B

---

*The Runnable interface

1. The Runnable interface is defined in which package?

A) java.io
B) java.util
C) java.lang
D) java.thread
✅ **Answer:** C

---

2. How many methods are there in the Runnable interface?

A) 2
B) 1
C) 3
D) 0
✅ **Answer:** B
(public void run())

---

3. Which method must be implemented when using Runnable?

A) start()
B) execute()
C) run()
D) launch()
✅ **Answer:** C

---

4. What is the main advantage of implementing Runnable over extending Thread?

A) Uses more memory
B) Allows multiple inheritance
C) Thread executes faster

D) No method overriding needed
✅ **Answer:** B

---

5. What is the correct way to use a class that implements Runnable?

A) Runnable r = new Thread();
B) Thread t = new Runnable();
C) Thread t = new Thread(new MyRunnable());
D) Thread t = MyRunnable.run();
✅ **Answer:** C

---

6. If you implement Runnable, how do you start the thread?

A) run()
B) start()
C) execute()
D) call()
✅ **Answer:** B
(*Call start() on a Thread object, not on the Runnable itself*)

---

7. What happens when you call run() directly on a Runnable object?

A) New thread is created
B) Exception is thrown
C) Code runs in the current thread
D) Compilation error
✅ **Answer:** C

---

8. Which of the following is TRUE about Runnable?

A) It is an abstract class
B) It is a functional interface
C) It extends Thread
D) It contains a static run method
✅ **Answer:** B

---

9. Which Java 8 feature makes Runnable easier to use?

A) Anonymous classes
B) Method references
C) Lambda expressions
D) Thread pools
✅ **Answer:** C

---

10. What is the correct lambda expression for a Runnable task?

A) Runnable r = () -> System.out.println("Run");
B) Runnable r = () => {}
C) Runnable r = new Runnable();
D) Runnable r = run() -> {}
✅ **Answer:** A

---

11. Which method of Thread accepts a Runnable object?

A) run(Runnable r)
B) execute(Runnable r)
C) Thread(Runnable r)
D) pass(Runnable r)
✅ **Answer:** C

---

12. What will happen if you pass null to Thread constructor as Runnable?

A) Compilation error
B) NullPointerException
C) Thread executes empty
D) JVM crash
✅ **Answer:** C

---

13. What is the default return type of the run() method in Runnable?

A) int
B) boolean
C) void
D) String
✅ **Answer:** C

---

14. Which of the following is NOT a valid way to create a thread using Runnable?

A) new Thread(new RunnableImpl()).start();
B) Thread t = new Thread(() -> System.out.println("Runnable")); t.start();
C) Runnable r = () -> {}; Thread t = r; t.start();
D) Runnable r = new MyRunnable(); new Thread(r).start();
✅ **Answer:** C

---

15. Why might you choose Runnable instead of Thread class?

A) It performs better
B) Java recommends it
C) You want to inherit from another class
D) It starts automatically
✅ **Answer:** C

---

16. Which is the correct class signature to implement Runnable?

A) class MyThread extends Runnable
B) class MyThread implements Thread
C) class MyThread implements Runnable
D) class MyThread implements Threadable
✅ **Answer:** C

---

17. Which interface does Runnable extend?

A) Cloneable
B) Serializable
C) None
D) Threadable
✅ **Answer:** C

---

18. How is Runnable different from Callable?

A) Callable returns a value
B) Runnable is used in parallelism
C) Runnable has more methods
D) Runnable is slower
✅ **Answer:** A

---

19. Which Executor method accepts a Runnable?

A) execute(Runnable)
B) invoke(Runnable)
C) call(Runnable)
D) run(Runnable)
✅ **Answer:** A

---

20. In a class implementing Runnable, which method signature is correct?

A) public int run()
B) private void run()
C) public void run()
D) void run(int x)
✅ **Answer:** C

---

21. Runnable is mainly used when:

A) You want to return a value from thread
B) You want multiple thread objects with same task
C) You want to execute one-time task

D) You want daemon threads

✅ **Answer:** B

---

*Thread synchronization technique

1. What is thread synchronization in Java used for?

A) Increasing thread speed
B) Reducing memory usage
C) Preventing race conditions
D) Making thread sleep

✅ **Answer:** C

---

2. Which keyword is used for synchronization in Java?

A) sync
B) synchronized
C) threadsafe
D) atomic

✅ **Answer:** B

---

3. What does the synchronized keyword do?

A) Pauses a thread
B) Stops other threads permanently
C) Allows only one thread to access the block/method
D) Creates new threads

✅ **Answer:** C

---

4. Which of the following can be synchronized?

A) Methods
B) Code blocks
C) Static methods
D) All of the above

✅ **Answer:** D

---

5. What happens when a thread cannot acquire a lock on a synchronized block?

A) It crashes
B) It waits until the lock is released
C) It skips the block
D) It throws an exception

✅ **Answer:** B

---

6. Which type of lock is used internally by Java synchronization?

A) File lock
B) Semaphore
C) Monitor lock
D) Thread lock
✅ **Answer:** C

---

7. Which object is used as the monitor when a non-static synchronized method is called?

A) The thread object
B) The class object
C) The object on which method is called
D) System.out
✅ **Answer:** C

---

8. Which is true for a static synchronized method?

A) It locks the class object
B) It locks the instance
C) It can’t be synchronized
D) It throws an error
✅ **Answer:** A

---

9. Which method releases the lock and waits for notification?

A) sleep()
B) notify()
C) wait()
D) yield()
✅ **Answer:** C

---

10. Which methods are used to communicate between threads in synchronized blocks?

A) start(), run()
B) notify(), wait(), notifyAll()
C) sleep(), join()
D) init(), destroy()
✅ **Answer:** B

---

11. Which method wakes up a single thread waiting on the object?

A) sleep()
B) yield()
C) notify()
D) start()
✅ **Answer:** C

---

12. Which method wakes up all waiting threads?

A) resumeAll()
B) notifyAll()
C) runAll()
D) yieldAll()
✅ **Answer:** B

---

13. What exception is thrown when wait() is used outside a synchronized block?

A) IllegalThreadStateException
B) InterruptedException
C) IllegalMonitorStateException
D) NoSuchElementException
✅ **Answer:** C

---

14. Which synchronization mechanism allows threads to coordinate without blocking?

A) busy wait
B) sleep()
C) atomic variables
D) Lock-free queues
✅ **Answer:** D

---

15. Which Java class provides advanced locking mechanism?

A) ThreadGroup
B) LockSupport
C) ReentrantLock
D) ThreadLock
✅ **Answer:** C

---

16. ReentrantLock belongs to which package?

A) java.util
B) java.lang
C) java.util.concurrent.locks
D) java.locking
✅ **Answer:** C

---

17. Which method acquires the lock in ReentrantLock?

A) open()
B) enter()
C) lock()

D) getLock()
✅ **Answer:** C

---

18. Which method releases the lock in ReentrantLock?

A) release()
B) unlock()
C) exit()
D) free()
✅ **Answer:** B

---

19. What happens if unlock() is called without lock()?

A) Lock is released
B) Nothing happens
C) Throws IllegalMonitorStateException
D) JVM halts
✅ **Answer:** C

---

20. Which of these is NOT a thread synchronization tool in Java?

A) Semaphore
B) CyclicBarrier
C) CountDownLatch
D) ThreadGroup
✅ **Answer:** D

---

21. Which method is used to forcefully release the CPU and allow other threads?

A) notify()
B) sleep()
C) yield()
D) resume()
✅ **Answer:** C

---

22. Which synchronization tool allows one thread to wait until a set of operations in other threads complete?

A) CountDownLatch
B) Semaphore
C) Lock
D) Runnable
✅ **Answer:** A

---

*Applying thread safety to Collection framework classes.

1. Are Java's core collection classes thread-safe by default?

A) Yes
B) No
C) Only ArrayList
D) Only HashMap
✅ **Answer:** B

---

2. Which collection class is synchronized by default?

A) ArrayList
B) HashSet
C) Hashtable
D) TreeMap
✅ **Answer:** C

---

3. How can you make a List thread-safe using Java’s utility class?

A) new ThreadSafeList()
B) Collections.threadSafeList()
C) Collections.synchronizedList(list)
D) Thread.synchronize(list)
✅ **Answer:** C

---

4. Which method wraps a map for thread safety?

A) Collections.makeSafe(map)
B) Collections.synchronize(map)
C) Collections.synchronizedMap(map)
D) Thread.safe(map)
✅ **Answer:** C

---

5. What is the major disadvantage of Collections.synchronizedList()?

A) It does not synchronize access
B) It throws exception often
C) It provides poor iteration safety
D) It changes data structure
✅ **Answer:** C
(*Iteration must be manually synchronized*)

---

6. Which concurrent collection is best suited for high-concurrency maps?

A) Hashtable
B) HashMap
C) TreeMap

D) ConcurrentHashMap
✅ **Answer:** D

---

7. Which package provides concurrent collection classes?

A) java.util
B) java.lang.concurrent
C) java.util.concurrent
D) java.concurrent
✅ **Answer:** C

---

8. Which collection allows thread-safe operations without locking the entire collection?

A) Vector
B) Hashtable
C) ConcurrentHashMap
D) ArrayList
✅ **Answer:** C

---

9. Which of the following is NOT a concurrent collection class?

A) CopyOnWriteArrayList
B) ConcurrentLinkedQueue
C) HashSet
D) ConcurrentSkipListMap
✅ **Answer:** C

---

10. What does CopyOnWriteArrayList do on every mutation (add/remove)?

A) Nothing
B) Copies the whole list
C) Locks the thread
D) Pauses the thread
✅ **Answer:** B

---

11. What is a good use case for CopyOnWriteArrayList?

A) High write frequency
B) Large data sizes
C) Many reads, few writes
D) Binary search
✅ **Answer:** C

---

12. Which method ensures a thread-safe Set?

A) Collections.synchronizedSet()
B) Thread.safeSet()
C) Set.makeThreadSafe()
D) Synchronized.set()
✅ **Answer:** A

---

13. Which of the following is true about Vector?

A) It is thread-safe
B) It is faster than ArrayList
C) It is a concurrent collection
D) It is deprecated
✅ **Answer:** A

---

14. How does ConcurrentHashMap improve concurrency?

A) It blocks entire map
B) It uses fine-grained locking
C) It runs only one thread
D) It duplicates data
✅ **Answer:** B

---

15. Which class uses internal segments for concurrency?

A) Vector
B) Hashtable
C) ConcurrentHashMap (Java 7)
D) TreeMap
✅ **Answer:** C

---

16. In Java 8+, ConcurrentHashMap uses:

A) Copy-on-write
B) Segment-based locking
C) Lock-free buckets and synchronized bins
D) Hash chains
✅ **Answer:** C

---

17. What does Collections.synchronizedCollection() do?

A) Makes a collection immutable
B) Makes a collection synchronized
C) Deletes a collection
D) Locks only reads
✅ **Answer:** B

---

18. Is iteration over a synchronized collection thread-safe by default?

A) Yes
B) No
C) Only on HashMap
D) Only in Java 11+
✅ **Answer:** B
(*Must manually synchronize during iteration*)

---

19. Which of these is true about ConcurrentLinkedQueue?

A) It is blocking
B) It uses locks
C) It is non-blocking and thread-safe
D) It does not allow null
✅ **Answer:** C

---

20. What is the thread-safe alternative to ArrayDeque?

A) Stack
B) ConcurrentLinkedDeque
C) Vector
D) Deque
✅ **Answer:** B

---

21. Which of these ensures safe iteration in concurrent environments?

A) Enumeration
B) Iterator
C) Fail-safe iterator
D) Fail-fast iterator
✅ **Answer:** C

---

22. What happens if you modify a HashMap from two threads without synchronization?

A) Works fine
B) Throws an error
C) May lead to data corruption or infinite loop
D) Gets auto-synchronized
✅ **Answer:** C

Session 24 & 25

*Database Access Methods, JDBC, driver & architecture

1. What does JDBC stand for?

A) Java DataBase Connection
B) Java DataBase Communication
C) Java Database Connectivity
D) Java Direct Base Connection
✅ **Answer:** C

---

2. Which package contains the JDBC classes and interfaces?

A) java.sql
B) java.jdbc
C) java.db
D) java.io
✅ **Answer:** A

---

3. Which method is used to load a JDBC driver class?

A) Driver.load()
B) Class.forName()
C) DriverManager.getConnection()
D) System.loadDriver()
✅ **Answer:** B

---

4. What interface provides a connection to a database in JDBC?

A) Statement
B) ResultSet
C) DriverManager
D) Connection
✅ **Answer:** D

---

5. Which method is used to execute a SELECT SQL query?

A) executeQuery()
B) executeUpdate()
C) execute()
D) runQuery()
✅ **Answer:** A

---

6. Which method is used to execute INSERT, UPDATE, or DELETE SQL commands?

A) executeQuery()
B) runQuery()

C) executeUpdate()
D) executeCommand()
✅ **Answer:** C

---

7. Which of the following is NOT a JDBC driver type?

A) Type 1
B) Type 2
C) Type 3
D) Type 5
✅ **Answer:** D

---

8. Which JDBC driver type is known as the "thin driver"?

A) Type 1
B) Type 2
C) Type 3
D) Type 4
✅ **Answer:** D

---

9. Which driver uses native OS libraries to connect to the database?

A) Type 1
B) Type 2
C) Type 3
D) Type 4
✅ **Answer:** B

---

10. Which driver communicates with a middleware server?

A) Type 1
B) Type 2
C) Type 3
D) Type 4
✅ **Answer:** C

---

11. What does DriverManager.getConnection() return?

A) Statement
B) Connection
C) ResultSet
D) URL
✅ **Answer:** B

---

12. What does the ResultSet object hold?

A) SQL queries
B) Connection data
C) Output of SQL SELECT query
D) Metadata
✅ **Answer:** C

---

13. What method moves the cursor to the next row in a ResultSet?

A) next()
B) move()
C) fetch()
D) step()
✅ **Answer:** A

---

14. Which of these methods is used to prevent SQL Injection?

A) executeQuery()
B) Statement
C) executeUpdate()
D) PreparedStatement
✅ **Answer:** D

---

15. Which method of PreparedStatement is used to set a string value?

A) setString()
B) putString()
C) insertString()
D) updateString()
✅ **Answer:** A

---

16. Which interface allows scrollable and updatable result sets?

A) Statement
B) PreparedStatement
C) CallableStatement
D) ResultSet
✅ **Answer:** D

---

17. Which of these is used to call stored procedures in JDBC?

A) CallableStatement
B) Statement
C) PreparedStatement
D) ProcedureCaller
✅ **Answer:** A

18. What is the default ResultSet type?

A) TYPE_SCROLL_SENSITIVE
B) TYPE_SCROLL_INSENSITIVE
C) TYPE_FORWARD_ONLY
D) TYPE_UPDATABLE
✅ **Answer:** C

---

19. Which driver is platform independent and requires no native library?

A) Type 1
B) Type 2
C) Type 3
D) Type 4
✅ **Answer:** D

---

20. Which object closes the connection to the database?

A) ResultSet.close()
B) Statement.close()
C) Connection.close()
D) Driver.close()
✅ **Answer:** C

---

21. Which of the following can improve performance by pre-compiling SQL?

A) Statement
B) ResultSet
C) PreparedStatement
D) Connection
✅ **Answer:** C

---

22. Which JDBC feature lets you update rows directly in ResultSet?

A) ResultSet.TYPE_FORWARD_ONLY
B) ResultSet.TYPE_SCROLL_INSENSITIVE
C) ResultSet.CONCUR_READ_ONLY
D) ResultSet.CONCUR_UPDATABLE
✅ **Answer:** D

---

23. Which method is used to execute a general SQL statement that may return multiple results?

A) executeQuery()
B) executeUpdate()
C) execute()

D) runQuery()
✅ **Answer:** C

---

24. What does the method wasNull() in ResultSet check?

A) If the result is null
B) If the column value in the last read was SQL NULL
C) If the query failed
D) If the table is empty
✅ **Answer:** B

---

25. Which driver translates JDBC calls into ODBC calls?

A) Type 1
B) Type 2
C) Type 3
D) Type 4
✅ **Answer:** A
(*Also known as JDBC-ODBC Bridge*)

---

26. Which method in Connection interface disables auto-commit mode?

A) setAutoCommit(false)
B) disableAutoCommit()
C) commitOff()
D) startTransaction()
✅ **Answer:** A

---

27. What does commit() do in JDBC?

A) Starts a transaction
B) Cancels the current query
C) Makes all changes made in the transaction permanent
D) Rolls back all changes
✅ **Answer:** C

---

28. Which JDBC component is responsible for managing a list of database drivers?

A) Connection
B) Statement
C) DriverManager
D) PreparedStatement
✅ **Answer:** C

---

29. Which interface is NOT part of the core JDBC API?

A) Connection
B) Statement
C) ResultSet
D) EntityManager
✅ **Answer:** D
(*EntityManager is part of JPA, not JDBC*)

---

30. Which class in JDBC is used to retrieve database metadata?

A) ResultSetMetaData
B) DatabaseMetaData
C) TableMetaData
D) DBInfo
✅ **Answer:** B

---

31. Which object helps to get column information of a ResultSet?

A) Statement
B) Connection
C) ResultSetMetaData
D) RowSet
✅ **Answer:** C

---

32. In JDBC, what will happen if you forget to close a Connection?

A) Nothing
B) Connection pool will auto-close
C) Memory leak or exhaustion of database connections
D) JVM will close it automatically
✅ **Answer:** C

---

*The java.sql package

1. Which package must be imported for using JDBC?

A) java.db
B) java.jdbc
C) java.sql
D) javax.jdbc
✅ **Answer:** C

---

2. The java.sql.Connection interface represents:

A) A SQL query
B) A network connection
C) A session with a specific database

D) A browser session
✅ **Answer:** C

---

3. Which interface is used to execute static SQL statements?

A) Statement
B) ResultSet
C) Connection
D) Driver
✅ **Answer:** A

---

4. Which interface represents the result set of a query?

A) Connection
B) ResultSet
C) Statement
D) MetaData
✅ **Answer:** B

---

5. What does PreparedStatement help avoid?

A) Runtime errors
B) SQL injection attacks
C) Compile-time errors
D) None of the above
✅ **Answer:** B

---

6. Which method of Statement is used to retrieve data from a table?

A) executeUpdate()
B) runQuery()
C) executeQuery()
D) fetch()
✅ **Answer:** C

---

7. Which interface is used to execute stored procedures?

A) Statement
B) PreparedStatement
C) CallableStatement
D) ProcedureStatement
✅ **Answer:** C

---

8. The method ResultSet.next() returns:

A) true if the next row exists
B) false if the next row exists
C) The row number
D) Nothing
✅ **Answer:** A

---

9. Which object is used to set SQL parameters dynamically?

A) Statement
B) PreparedStatement
C) Connection
D) DriverManager
✅ **Answer:** B

---

10. Which interface provides metadata about a database?

A) ResultSetMetaData
B) DatabaseMetaData
C) ConnectionMetaData
D) QueryMetaData
✅ **Answer:** B

---

11. Which method is used to retrieve a DatabaseMetaData object?

A) getMetaData()
B) Connection.getDatabaseMetaData()
C) DriverManager.getMetaData()
D) Statement.getMeta()
✅ **Answer:** B

---

12. The method setAutoCommit(false) is used to:

A) Start a background thread
B) Turn off commit feature
C) Disable auto commit mode
D) End a transaction
✅ **Answer:** C

---

13. What type of exception do most JDBC methods throw?

A) IOException
B) SQLException
C) ClassNotFoundException
D) RuntimeException
✅ **Answer:** B

---

14. ResultSet is part of which package?

A) java.sql
B) java.util
C) javax.sql
D) java.db
✅ **Answer:** A

---

15. Which interface provides information about column types and properties in a ResultSet?

A) ResultSet
B) ResultSetMetaData
C) ColumnMetaData
D) TableInfo
✅ **Answer:** B

---

16. What is the purpose of DriverManager class?

A) Create new SQL drivers
B) Register and manage JDBC drivers
C) Send data packets
D) Perform data encryption
✅ **Answer:** B

---

17. Which method is used to close the ResultSet?

A) ResultSet.destroy()
B) ResultSet.flush()
C) ResultSet.close()
D) ResultSet.terminate()
✅ **Answer:** C

---

18. Which of the following types does ResultSet.getString(columnIndex) return?

A) Integer
B) String
C) Boolean
D) Double
✅ **Answer:** B

---

19. Can a ResultSet be scrollable and updatable?

A) No
B) Only scrollable
C) Yes, if created with specific options

D) Only updatable

✅ **Answer:** C

---

20. What will happen if you call close() on a Connection before reading the entire ResultSet?

A) Nothing
B) The program continues
C) An exception may be thrown
D) All data is auto-read

✅ **Answer:** C

---

21. Which method is used to roll back changes in a transaction?

A) cancel()
B) rollback()
C) reset()
D) terminate()

✅ **Answer:** B

---

22. Which method of Statement executes an SQL statement that may return multiple results?

A) executeQuery()
B) executeUpdate()
C) execute()
D) run()

✅ **Answer:** C

---

23. The java.sql.Date class extends which Java class?

A) java.util.Date
B) java.lang.Date
C) java.sql.Timestamp
D) java.sql.Time

✅ **Answer:** A

---

24. To retrieve a numeric value from ResultSet, which method is used?

A) getInt()
B) getNumber()
C) getLong()
D) getValue()

✅ **Answer:** A

---

25. Which exception must be handled or declared when using JDBC classes?

A) IOException
B) SQLException

C) FileNotFoundException
D) NullPointerException
✅ **Answer:** B

---

26. Which interface provides methods to batch multiple SQL commands for execution?

A) Statement
B) Batchable
C) PreparedStatement
D) BatchUpdateException
✅ **Answer:** C

---

27. Which method adds a SQL command to a batch?

A) addBatch()
B) appendBatch()
C) batchAdd()
D) addToBatch()
✅ **Answer:** A

---

28. What does the executeBatch() method return?

A) Total number of successful commands
B) An array of update counts for each command
C) Boolean indicating success
D) Nothing
✅ **Answer:** B

---

29. Which interface allows reading and writing of tabular data from a database?

A) RowSet
B) ResultSet
C) Statement
D) TableSet
✅ **Answer:** A

---

30. The java.sql.Time class is used to represent:

A) Date only
B) Time only (hour, minute, second)
C) Timestamp
D) Time zone data
✅ **Answer:** B

---

31. Which of these interfaces supports scroll-insensitive ResultSets?

A) ResultSet.TYPE_FORWARD_ONLY
B) ResultSet.TYPE_SCROLL_SENSITIVE
C) ResultSet.TYPE_SCROLL_INSENSITIVE
D) ResultSet.CONCUR_READ_ONLY
✅ **Answer:** C

---

*Driver Manager, Connection, Statement, PreparedStatement, Result Set

1. What is the primary role of DriverManager in JDBC?
   A) Manage connections
   B) Manage registered drivers and establish connections
   C) Execute SQL queries
   D) Manage transactions
   Answer: B
2. How do you register a JDBC driver with DriverManager?
   A) DriverManager.registerDriver()
   B) Class.forName()
   C) DriverManager.connect()
   D) Driver.register()
   Answer: B (typically)
3. Which method establishes a database connection?
   A) getConnection()
   B) connect()
   C) openConnection()
   D) DriverManager.getConnection()
   Answer: D
4. What type of object does DriverManager.getConnection() return?
   A) Driver
   B) Connection
   C) Statement
   D) ResultSet
   Answer: B
5. Can multiple drivers be registered with DriverManager?
   A) Yes
   B) No
   Answer: A
6. What happens if no suitable driver is found for the URL?
   A) Returns null
   B) Throws SQLException
   C) Returns a default driver
   D) Throws ClassNotFoundException
   Answer: B
7. Which of these is a valid JDBC URL prefix?
   A) jdbc:mysql://
   B) http://mysql/
   C) sql://localhost/
   D) db:mysql:
   Answer: A

8. What is the first step in making a JDBC connection?
   A) Create Statement
   B) Register the driver
   C) Execute SQL
   D) Close Connection
   Answer: B
9. Which class method loads a driver class dynamically?
   A) DriverManager.register()
   B) Class.forName()
   C) DriverManager.loadDriver()
   D) Driver.load()
   Answer: B
10. DriverManager is part of which package?
   A) java.jdbc
   B) java.sql
   C) javax.sql
   D) java.db
   Answer: B
11. What does the DriverManager do internally with registered drivers?
   A) Loads drivers on demand
   B) Maintains a list and selects driver for URL
   C) Closes unused drivers
   D) Manages driver pools
   Answer: B
12. Which exception is typically thrown by DriverManager methods?
   A) IOException
   B) SQLException
   C) ClassNotFoundException
   D) NullPointerException
   Answer: B
13. Which is the correct syntax to get a connection?
   A) DriverManager.getConnection(url, user, password)
   B) DriverManager.connect(url, user, password)
   C) Connection.getConnection(url, user, password)
   D) Driver.getConnection(url)
   Answer: A
14. What happens when you call DriverManager.getConnection() multiple times?
   A) Returns same Connection object
   B) Creates new Connection each time
   C) Throws exception
   D) Returns null
   Answer: B
15. Can DriverManager be used to connect to multiple types of databases?
   A) Yes
   B) No
   Answer: A
16. Is DriverManager thread-safe?
   A) Yes
   B) No
   Answer: A

17. How to deregister a driver?
A) DriverManager.deregisterDriver(driver)
B) Driver.deregister()
C) DriverManager.removeDriver(driver)
D) Not possible
Answer: A

18. Which version of JDBC introduced DriverManager?
A) JDBC 1.0
B) JDBC 2.0
C) JDBC 3.0
D) JDBC 4.0
Answer: A

19. Is it mandatory to call Class.forName() in JDBC 4.0 and above?
A) Yes
B) No, automatic driver loading is supported
Answer: B

20. How does DriverManager select the driver?
A) First registered driver
B) Based on URL prefix match
C) Alphabetically
D) Randomly
Answer: B

21. Which interface represents a session with a database?
A) Driver
B) Connection
C) Statement
D) ResultSet
Answer: B

22. Which method is used to commit a transaction?
A) commit()
B) save()
C) finalize()
D) submit()
Answer: A

23. How to disable auto-commit mode?
A) setAutoCommit(false)
B) disableAutoCommit()
C) autoCommit(false)
D) setCommit(false)
Answer: A

24. What happens if you don’t close a Connection?
A) No effect
B) Possible memory leaks and connection exhaustion
C) JVM auto-closes it
D) Connection closes automatically after query
Answer: B

25. Which method closes a Connection?
A) close()
B) terminate()
C) end()
D) stop()
Answer: A

26. Which method retrieves metadata about the database?
A) getMetaData()
B) getInfo()
C) getDatabase()
D) getConnectionInfo()
Answer: A

27. Can you set transaction isolation level via Connection?
A) Yes, using setTransactionIsolation()
B) No
Answer: A

28. Which isolation level allows dirty reads?
A) TRANSACTION_READ_UNCOMMITTED
B) TRANSACTION_READ_COMMITTED
C) TRANSACTION_REPEATABLE_READ
D) TRANSACTION_SERIALIZABLE
Answer: A

29. Is Connection thread-safe?
A) Yes
B) No
Answer: B

30. Which method rolls back a transaction?
A) rollback()
B) undo()
C) cancel()
D) revert()
Answer: A

31. Can a Connection create Statement objects?
A) Yes
B) No
Answer: A

32. Which method returns whether the connection is closed?
A) isClosed()
B) checkClosed()
C) isConnectionClosed()
D) closed()
Answer: A

33. What happens if you call commit() on an auto-commit connection?
A) No effect
B) Commit transaction
C) Throws SQLException
D) Rolls back transaction
Answer: A

34. Which of the following methods can set the connection read-only?
A) setReadOnly(true)
B) makeReadOnly()
C) enableReadOnly()
D) setReadOnly()
Answer: A

35. Which method changes the catalog name for a connection?
A) setCatalog(String catalog)
B) changeCatalog(String name)
C) switchCatalog(String name)
D) setDatabase(String name)
Answer: A

36. How to set network timeout for a connection?
A) setNetworkTimeout(Executor executor, int milliseconds)
B) setTimeout(int ms)
C) configureTimeout()
D) setQueryTimeout()
Answer: A

37. Which method checks if a Connection is valid?
A) isValid(int timeout)
B) validate()
C) checkValidity()
D) isConnectionAlive()
Answer: A

38. Can Connection interface be implemented by custom classes?
A) Yes
B) No
Answer: A

39. Which exceptions are thrown by Connection methods?
A) IOException
B) SQLException
C) NullPointerException
D) IllegalArgumentException
Answer: B

40. Which method sets the schema for a connection?
A) setSchema(String schema)
B) changeSchema(String schema)
C) useSchema(String schema)
D) setDatabaseSchema(String schema)
Answer: A

41. • Which interface is used to execute simple SQL statements?
A) Statement
B) PreparedStatement
C) CallableStatement
D) ResultSet
Answer: A

42. • Which method executes a SELECT SQL command?
A) executeQuery()
B) executeUpdate()
C) execute()
D) runQuery()
Answer: A

43. • Which method executes INSERT, UPDATE, or DELETE?
A) executeUpdate()
B) executeQuery()
C) execute()
D) runUpdate()
Answer: A

44. • Which method can execute any SQL command?
A) execute()
B) executeQuery()
C) executeUpdate()
D) run()
Answer: A

45. • Which method closes the Statement?
A) close()

B) terminate()
C) stop()
D) end()
Answer: A

46. • Can a Statement object be reused for multiple SQL commands?
A) Yes
B) No
Answer: A

47. • What is the return type of executeQuery()?
A) ResultSet
B) int
C) boolean
D) void
Answer: A

48. • What does executeUpdate() return?
A) Number of affected rows
B) ResultSet
C) Boolean
D) None
Answer: A

49. • Does Statement support batch processing?
A) Yes
B) No
Answer: A

50. • Which method adds SQL commands to batch?
A) addBatch()
B) batchAdd()
C) addCommand()
D) insertBatch()
Answer: A

51. • Which method executes batch commands?
A) executeBatch()
B) runBatch()
C) batchExecute()
D) runBatchCommands()
Answer: A

52. • Can Statement execute multiple queries in one execute()?
A) Yes
B) No
Answer: A

53. • What is the default ResultSet concurrency for Statement?
A) CONCUR_READ_ONLY
B) CONCUR_UPDATABLE
C) CONCUR_WRITEABLE
D) CONCUR_NONE
Answer: A

54. • Can you scroll ResultSet from a Statement?
A) No, ResultSet is forward-only by default
B) Yes
Answer: A

55. • Which method clears batch commands?
A) clearBatch()
B) resetBatch()
C) clearCommands()

D) resetCommands()
Answer: A

56. • What is the default ResultSet type for Statement?
A) TYPE_FORWARD_ONLY
B) TYPE_SCROLL_INSENSITIVE
C) TYPE_SCROLL_SENSITIVE
D) TYPE_UPDATABLE
Answer: A

57. • Which package contains the Statement interface?
A) java.sql
B) java.jdbc
C) javax.sql
D) java.db
Answer: A

58. • Can Statements be closed automatically when Connection closes?
A) Yes
B) No
Answer: A

59. • Which exception is commonly thrown when executing a Statement?
A) SQLException
B) IOException
C) ClassNotFoundException
D) RuntimeException
Answer: A

60. • Which method executes SQL commands that return multiple results?
A) execute()
B) executeQuery()
C) executeUpdate()
D) executeMultiple()
Answer: A

61. Which interface extends Statement and allows precompiled SQL statements?
A) PreparedStatement
B) CallableStatement
C) Statement
D) ResultSet
Answer: A

62. What is the benefit of PreparedStatement over Statement?
A) Prevents SQL injection
B) Improves performance with precompiled queries
C) Both A and B
D) None
Answer: C

63. Which method sets an integer parameter?
A) setInt(int parameterIndex, int value)
B) setInteger()
C) setIntValue()
D) setParamInt()
Answer: A

64. How do you set a string parameter?
A) setString(int parameterIndex, String value)
B) setStr()
C) setParameterString()
D) setStringValue()
Answer: A

65. How to execute a PreparedStatement?
A) executeQuery() or executeUpdate()
B) execute() only
C) runQuery()
D) runUpdate()
Answer: A

66. Can PreparedStatement be used for SELECT queries?
A) Yes
B) No
Answer: A

67. How do you clear parameters in PreparedStatement?
A) clearParameters()
B) resetParameters()
C) clear()
D) reset()
Answer: A

68. Can PreparedStatements be reused with different parameters?
A) Yes
B) No
Answer: A

69. Which method returns the SQL query string?
A) toString()
B) getQueryString()
C) getSQL()
D) getQuery()
Answer: A (or driver dependent)

70. PreparedStatement is useful in preventing:
A) Runtime errors
B) SQL Injection
C) Compile time errors
D) Network errors
Answer: B

71. Which package contains PreparedStatement?
A) java.sql
B) java.jdbc
C) javax.sql
D) java.db
Answer: A

72. Which method sets a double parameter?
A) setDouble(int parameterIndex, double value)
B) setDecimal()
C) setFloat()
D) setReal()
Answer: A

73. Can you use PreparedStatement for batch updates?
A) Yes
B) No
Answer: A

74. Which method adds current parameters to batch?
A) addBatch()
B) batchAdd()
C) addParameters()
D) batchParameters()
Answer: A

75. What happens if you don’t set a parameter in PreparedStatement?
A) SQLException thrown
B) Default value used
C) Null inserted
D) Statement ignored
Answer: A

76. Which method executes batch commands on PreparedStatement?
A) executeBatch()
B) runBatch()
C) batchExecute()
D) execute()
Answer: A

77. What type of SQL statements can PreparedStatement execute?
A) Only SELECT
B) Only INSERT/UPDATE/DELETE
C) Both SELECT and DML
D) None
Answer: C

78. Which method sets a boolean parameter?
A) setBoolean(int parameterIndex, boolean value)
B) setBool()
C) setFlag()
D) setBit()
Answer: A

79. Which is a benefit of using PreparedStatement for repeated queries?
A) Reduced parsing overhead
B) Increased network traffic
C) Reduced security
D) Slower execution
Answer: A

80. What exception type is thrown by PreparedStatement methods?
A) IOException
B) SQLException
C) ClassNotFoundException
D) RuntimeException
Answer: B

81. • Which interface represents data returned from a database query?
A) Connection
B) Statement
C) ResultSet
D) Driver
Answer: C

82. • Which method moves the cursor forward one row?
A) next()
B) move()
C) forward()
D) getNext()
Answer: A

83. • How to retrieve a String from a ResultSet?
A) getString(columnIndex)
B) getText()
C) getStringValue()
D) getChar()
Answer: A

84. • Can ResultSet be scrollable?
A) Yes, with specific settings
B) No
Answer: A

85. • Which ResultSet type is forward-only?
A) TYPE_FORWARD_ONLY
B) TYPE_SCROLL_INSENSITIVE
C) TYPE_SCROLL_SENSITIVE
D) TYPE_UPDATABLE
Answer: A

86. • How to update data in a ResultSet?
A) Using updateXXX() methods and calling updateRow()
B) Direct SQL commands
C) Cannot update ResultSet
D) Using refresh() method
Answer: A

87. • Which method closes the ResultSet?
A) close()
B) terminate()
C) end()
D) stop()
Answer: A

88. • What does `getMetaData()` on ResultSet return?
A) ResultSetMetaData object
B) DatabaseMetaData
C) Column info string
D) None
Answer: A

89. • Can you retrieve data by column name?
A) Yes
B) No
Answer: A

90. • Which method checks if the last read column was SQL NULL?
A) wasNull()
B) isNull()
C) getNull()
D) checkNull()
Answer: A

91. • Which method moves cursor to the first row?
A) first()
B) moveFirst()
C) start()
D) getFirst()
Answer: A

92. • Which method moves cursor to the last row?
A) last()
B) end()
C) moveLast()
D) getLast()
Answer: A

93. • Which method moves cursor to the previous row?
A) previous()
B) back()
C) moveBack()

D) prior()
Answer: A

94. • How to insert a new row via ResultSet?
A) moveToInsertRow(), updateXXX(), insertRow()
B) insertRow() only
C) Cannot insert via ResultSet
D) addRow()
Answer: A

95. • Can ResultSet be updatable?
A) Yes, with CONCUR_UPDATABLE
B) No
Answer: A

96. • Which method refreshes a row's data from the database?
A) refreshRow()
B) reload()
C) updateRow()
D) syncRow()
Answer: A

97. • What does getConcurrency() return?
A) Concurrency mode of ResultSet
B) Number of rows
C) Lock mode
D) None
Answer: A

98. • What exception is thrown on invalid ResultSet operation?
A) SQLException
B) IOException
C) RuntimeException
D) NullPointerException
Answer: A

99. • Which package contains ResultSet?
A) java.sql
B) java.db
C) javax.sql
D) java.jdbc
Answer: A

100. • How do you check if ResultSet is empty?
A) Call next(), if false, empty
B) isEmpty() method
C) size() method
D) No way
Answer: A

Session 26 & 27

*CallableStatement, Stored procedure & functions

CallableStatement –

1. Which JDBC interface is used to execute stored procedures?
   A) Statement
   B) PreparedStatement
   C) CallableStatement
   D) ResultSet
   Answer: C

2. How do you prepare a CallableStatement?
   A) Connection.prepareCall()
   B) Connection.prepareStatement()
   C) Connection.createCallable()
   D) Connection.createStatement()
   Answer: A

3. What syntax is used in CallableStatement SQL string?
   A) {call procedureName(?, ?)}
   B) call procedureName(?, ?)
   C) exec procedureName
   D) execute procedureName
   Answer: A

4. How to register an OUT parameter in CallableStatement?
   A) registerOutParameter(int parameterIndex, int sqlType)
   B) setOutParameter()
   C) registerParameter()
   D) setOutput()
   Answer: A

5. Which method executes the CallableStatement?
   A) execute()
   B) executeQuery()
   C) executeUpdate()
   D) run()
   Answer: A

6. Can CallableStatement return ResultSet objects?
   A) Yes
   B) No
   Answer: A

7. How do you retrieve an OUT parameter value?
   A) getXXX(int parameterIndex)
   B) getOutParameter()
   C) getParameterValue()
   D) getResult()
   Answer: A

8. Which exception is thrown on CallableStatement errors?
   A) SQLException
   B) IOException

C) RuntimeException
D) ClassNotFoundException
Answer: A

9. Can CallableStatement support IN, OUT, and INOUT parameters?
A) Yes
B) No
Answer: A

10. Which package contains CallableStatement?
A) java.sql
B) javax.sql
C) java.db
D) java.jdbc
Answer: A

11. What is the difference between execute() and executeQuery() in CallableStatement?
A) execute() can run any SQL, executeQuery() only SELECT
B) executeQuery() runs any SQL
C) Both are same
D) None
Answer: A

12. How to close a CallableStatement?
A) close()
B) terminate()
C) stop()
D) end()
Answer: A

13. Can you reuse a CallableStatement with different parameters?
A) Yes
B) No
Answer: A

14. What does {? = call functionName(?)} represent?
A) Calling a stored function with return value
B) Calling a stored procedure
C) Syntax error
D) None
Answer: A

15. Which method registers the return value for a stored function?
A) registerOutParameter(1, sqlType)
B) setReturnParameter()
C) registerReturnValue()
D) registerFunctionReturn()
Answer: A

16. Can CallableStatement be used to execute dynamic SQL?
A) No
B) Yes
Answer: B (if dynamic SQL is a stored procedure)

17. Is CallableStatement thread-safe?
A) No
B) Yes
Answer: A

18. Which JDBC driver type supports CallableStatement?
A) All driver types
B) Only Type 4
C) Only Type 1
D) Only Type 2
Answer: A (mostly)

19. How are multiple ResultSets handled in CallableStatement?
A) Using getMoreResults() method
B) Using nextResultSet()
C) Multiple execute() calls
D) Not supported
Answer: A

20. What is the default ResultSet concurrency for CallableStatement?
A) CONCUR_READ_ONLY
B) CONCUR_UPDATABLE
C) CONCUR_WRITEABLE
D) CONCUR_NONE
Answer: A

---

2. Stored Procedures –

I. What is a stored procedure?
A) A precompiled set of SQL statements stored in the database
B) A Java method
C) A database table
D) A user interface form
Answer: A

II. Stored procedures can improve:
A) Performance
B) Security
C) Code reusability
D) All of the above
Answer: D

III. Which SQL command is used to create a stored procedure?
A) CREATE PROCEDURE
B) CREATE FUNCTION
C) CREATE PROCEDURE CALL
D) CREATE EXEC
Answer: A

IV. Stored procedures can accept which parameter types?
A) IN
B) OUT

C) INOUT
D) All of the above
Answer: D

V. Stored procedures are stored in:
A) Database server
B) Client machine
C) Application server
D) Web server
Answer: A

VI. Can stored procedures return ResultSets?
A) Yes
B) No
Answer: A

VII. Which language is mostly used to write stored procedures?
A) SQL or procedural SQL (PL/SQL, T-SQL)
B) Java
C) Python
D) C++
Answer: A

VIII. Which is true about stored procedure transactions?
A) Can contain commit and rollback
B) Cannot control transactions
C) Only reads data
D) None
Answer: A

IX. How do stored procedures help prevent SQL injection?
A) Parameters are bound and validated
B) Queries are dynamic
C) Queries are concatenated
D) They don’t help
Answer: A

X. How is a stored procedure executed in SQL?
A) EXEC procedureName or CALL procedureName()
B) RUN procedureName
C) START procedureName
D) EXECUTE procedureName()
Answer: A

XI. Can stored procedures call other stored procedures?
A) Yes
B) No
Answer: A

XII. Stored procedures help in:
A) Centralized business logic
B) Reducing network traffic
C) Both A and B

D) None
Answer: C

XIII. What does the keyword BEGIN and END signify in stored procedures?
A) Block of procedural statements
B) Comments
C) No meaning
D) Transaction boundaries only
Answer: A

XIV. Which database supports stored procedures?
A) MySQL
B) Oracle
C) SQL Server
D) All of the above
Answer: D

XV. Which of the following can be a disadvantage of stored procedures?
A) Increased complexity
B) Vendor lock-in
C) Hard to debug
D) All of the above
Answer: D

XVI. How do stored procedures handle errors?
A) Using exception handlers or error handling syntax
B) Not possible
C) Automatically fixed
D) None
Answer: A

XVII. Which statement modifies data inside a stored procedure?
A) INSERT, UPDATE, DELETE
B) SELECT only
C) CREATE only
D) DROP only
Answer: A

XVIII. Are stored procedure parameters optional?
A) Some databases support default values
B) No
Answer: A

XIX. Can stored procedures return values?
A) No
B) Yes, via OUT parameters or return codes
Answer: B

XX. Which tool is used to debug stored procedures?
A) Database IDE or tools like SQL Developer, SSMS
B) Command prompt only
C) Web browser
D) None
Answer: A

5. Stored Functions –

XXI. What is a stored function?
A) A database routine that returns a single value
B) A Java method
C) A database table
D) A report generator
Answer: A

XXII. How is a stored function different from a stored procedure?
A) Function returns a value, procedure may not
B) Function cannot modify data, procedure can
C) Function can be used in SQL expressions
D) All of the above
Answer: D

XXIII. Which SQL command creates a function?
A) CREATE FUNCTION
B) CREATE PROCEDURE
C) CREATE FUNC
D) CREATE ROUTINE
Answer: A

XXIV. Can stored functions have parameters?
A) Yes
B) No
Answer: A

XXV. How do stored functions return a value?
A) Using RETURN statement
B) Using OUT parameter
C) Using PRINT
D) Using SELECT
Answer: A

XXVI. Can stored functions contain SQL statements?
A) Yes
B) No
Answer: A

XXVII. Stored functions can be used in:
A) SQL queries
B) Stored procedures
C) Views
D) All of the above
Answer: D

XXVIII. Which of the following is NOT true about stored functions?
A) Can modify database data
B) Should be deterministic ideally
C) Return single value

D) Can be used in expressions
Answer: A

XXIX. Can stored functions perform transactions?
A) Typically no, they should be side-effect free
B) Yes
Answer: A

XXX. How are stored functions called?
A) In SQL statements like SELECT functionName(params)
B) Using EXEC
C) Using RUN
D) Using CALL only
Answer: A

XXXI. Which language is commonly used for stored functions?
A) SQL, PL/SQL, T-SQL
B) Java
C) Python
D) C++
Answer: A

XXXII. Can stored functions return ResultSets?
A) No, only single values
B) Yes
Answer: A

XXXIII. Are stored functions useful for computed columns?
A) Yes
B) No
Answer: A

XXXIV. Can stored functions call stored procedures?
A) No
B) Yes (depends on DB)
Answer: A (mostly no)

XXXV. What is a deterministic function?
A) Always returns the same output for same input
B) Has side effects
C) Modifies data
D) None
Answer: A

XXXVI. Can stored functions be recursive?
A) Yes (in some DBs)
B) No
Answer: A

XXXVII. Which of the following is an advantage of stored functions?
A) Code reuse
B) Easier debugging
C) Better performance

D) All of the above
Answer: D

XXXVIII. How do you drop a stored function?
A) DROP FUNCTION functionName
B) DELETE FUNCTION
C) REMOVE FUNCTION
D) DROP PROC
Answer: A

XXXIX. Which of these can be used in stored function parameters?
A) IN only
B) OUT only
C) INOUT only
D) IN, OUT, INOUT
Answer: A

XL. Which exception type is common in stored functions?
A) SQLException
B) IOException
C) RuntimeException
D) NullPointerException
Answer: A

---

*Scrollable ResultSet

1. What does a Scrollable ResultSet allow?
A) Moving cursor forward only
B) Moving cursor both forward and backward
C) No movement of cursor
D) Only moving to the last row
Answer: B

2. Which ResultSet type supports scrolling?
A) TYPE_FORWARD_ONLY
B) TYPE_SCROLL_INSENSITIVE
C) TYPE_SCROLL_SENSITIVE
D) Both B and C
Answer: D

3. How do you create a scrollable ResultSet?
A) Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
B) conn.createStatement()
C) conn.prepareStatement()
D) None of these
Answer: A

4. What is the difference between TYPE_SCROLL_INSENSITIVE and TYPE_SCROLL_SENSITIVE?
A) INSENSITIVE does not reflect DB changes, SENSITIVE does
B) Both reflect DB changes
C) Both do not reflect DB changes
D) None
Answer: A

5. Which method moves the cursor to the last row in ResultSet?
   A) last()
   B) end()
   C) moveToLast()
   D) final()
   Answer: A
6. Which method moves the cursor to the first row?
   A) first()
   B) begin()
   C) start()
   D) moveToFirst()
   Answer: A
7. What does beforeFirst() do in ResultSet?
   A) Moves cursor before the first row
   B) Moves cursor after the last row
   C) Moves cursor to the first row
   D) Moves cursor to the last row
   Answer: A
8. What is the effect of afterLast() method?
   A) Moves cursor after the last row
   B) Moves cursor before the first row
   C) Moves cursor to last row
   D) Moves cursor to first row
   Answer: A
9. How do you move the cursor to an absolute row number in ResultSet?
   A) absolute(int row)
   B) move(int row)
   C) gotoRow(int row)
   D) setPosition(int row)
   Answer: A
10. If you call absolute(-1), where does the cursor move?
    A) Last row
    B) First row
    C) Throws exception
    D) Before first row
    Answer: A
11. What does relative(int rows) method do?
    A) Moves cursor relative to current position by rows
    B) Moves to absolute position
    C) Moves cursor to first row
    D) Moves cursor to last row
    Answer: A
12. Which method retrieves the current row number of the cursor?
    A) getRow()
    B) rowNumber()
    C) currentRow()
    D) row()
    Answer: A
13. Is Scrollable ResultSet supported by all JDBC drivers?
    A) No, driver dependent
    B) Yes, all support

C) No drivers support it
D) Only some databases support
Answer: A

14. What concurrency modes are supported with scrollable ResultSet?
A) CONCUR_READ_ONLY and CONCUR_UPDATABLE
B) CONCUR_READ_ONLY only
C) CONCUR_UPDATABLE only
D) None
Answer: A

15. Can you update rows in a scrollable and updatable ResultSet?
A) Yes
B) No
Answer: A

16. What happens if you try to scroll in a TYPE_FORWARD_ONLY ResultSet?
A) Throws SQLException
B) Scrolls normally
C) No effect
D) Cursor moves to first row only
Answer: A

17. Which method moves the cursor to the next row?
A) next()
B) advance()
C) moveNext()
D) goNext()
Answer: A

18. How to check if cursor is before the first row?
A) isBeforeFirst()
B) isBefore()
C) isFirst()
D) isCursorBefore()
Answer: A

19. Can you create a scrollable ResultSet using PreparedStatement?
A) Yes, using prepareStatement(sql, type, concurrency)
B) No
C) Only with Statement
D) Only with CallableStatement
Answer: A

20. Which method moves the cursor to the previous row?
A) previous()
B) back()
C) goBack()
D) last()
Answer: A

---

*Writing multi-tiered DB applications (Use DAO & DTO layers)

1. What does DAO stand for?
A) Data Access Object
B) Data Application Object
C) Data Arithmetic Operator

D) Database Access Operator
Answer: A

2. What is the primary responsibility of the DAO layer?
A) Encapsulating database access logic
B) Handling UI rendering
C) Managing business rules
D) Managing network connections
Answer: A

3. What does DTO stand for?
A) Data Transfer Object
B) Data Type Object
C) Data Transaction Operation
D) Database Transfer Operator
Answer: A

4. What is the main purpose of a DTO?
A) Carry data between processes or layers
B) Execute database queries
C) Handle business logic
D) Manage user interface
Answer: A

5. Which layer typically uses DTO objects?
A) Service or Business layer
B) DAO layer
C) UI layer
D) Database layer
Answer: A

6. In a multi-tiered application, which layer directly interacts with the database?
A) DAO layer
B) DTO layer
C) Business layer
D) Presentation layer
Answer: A

7. Why is it recommended to separate DAO and DTO?
A) To decouple data persistence and data representation
B) To combine UI and DB logic
C) To increase code duplication
D) None of the above
Answer: A

8. Which of these is NOT a typical DAO method?
A) findById()
B) save()
C) renderUI()
D) delete()
Answer: C

9. How does DAO improve application maintainability?
A) By isolating database logic in one place
B) By mixing SQL in UI code
C) By hardcoding SQL queries everywhere
D) By embedding database code in business logic
Answer: A

10. Which design pattern does DAO represent?
A) Structural design pattern
B) Behavioral design pattern
C) Creational design pattern
D) Architectural design pattern
Answer: A (some say Architectural)
11. How does DTO help in reducing network load?
A) By bundling multiple data fields in one object
B) Sending multiple separate data packets
C) Using JSON instead of XML
D) By compressing images
Answer: A
12. What kind of data does DTO typically contain?
A) Only data, no behavior (methods)
B) Complex business logic
C) UI rendering code
D) Database connection details
Answer: A
13. Can DTO objects be mutable?
A) Usually yes, but can be immutable for thread safety
B) Never mutable
C) Only in UI layer
D) None of these
Answer: A
14. Which Java interface is typically implemented by DAO classes?
A) Custom DAO interface specific to entity
B) Runnable
C) Serializable
D) Cloneable
Answer: A
15. In DAO pattern, how is connection to the database typically managed?
A) Using connection pooling or passed connection object
B) Creating new connection in every method
C) Using static connection only
D) Directly opening raw sockets
Answer: A
16. Which layer converts database entities to DTO objects?
A) DAO or Service layer
B) Presentation layer
C) UI layer
D) Network layer
Answer: A
17. What is a key advantage of using DAO pattern?
A) Database vendor independence
B) Tightly coupled code
C) Hardcoded SQL statements everywhere
D) No testability
Answer: A
18. How do DTO and DAO differ?
A) DAO accesses data, DTO carries data
B) DAO carries data, DTO accesses data

C) Both are the same
D) None
Answer: A

19. Which Java technology often uses DAO & DTO layers?
A) Java EE / Jakarta EE
B) Java ME
C) Android only
D) JavaFX only
Answer: A

20. How does multi-tier architecture benefit from DAO & DTO?
A) Clear separation of concerns
B) Slower development
C) Less maintainable code
D) Increased tight coupling
Answer: A

Session 28 & 29

*Java8 Interfaces, default methods

1. Which Java version introduced default methods in interfaces?
A) Java 7
B) Java 8
C) Java 9
D) Java 6
Answer: B

2. What is the purpose of default methods in Java interfaces?
A) To provide method implementation inside interfaces
B) To make interfaces abstract
C) To prevent method overriding
D) To allow interfaces to have constructors
Answer: A

3. Which keyword is used to declare a default method in an interface?
A) default
B) static
C) final
D) abstract
Answer: A

4. Can an interface in Java 8 have static methods?
A) Yes
B) No
Answer: A

5. Which of the following is a valid default method declaration in an interface?
A) default void show() { System.out.println("Hello"); }
B) void default show();
C) final void show() {}
D) abstract default void show();
Answer: A

6. If a class implements two interfaces having the same default method, what happens?
A) The class must override the default method
B) The compiler throws an error
C) The method from the first interface is chosen automatically
D) The method from the second interface is chosen automatically
Answer: A

7. Can default methods call other methods within the same interface?
A) Yes
B) No
Answer: A

8. Are default methods inherited by subclasses implementing the interface?
A) Yes
B) No
Answer: A

9. Can default methods access instance variables?
A) No, interfaces can't have instance variables
B) Yes, always
C) Only static variables
D) Only final variables
Answer: A

10. How do you call a default method from a specific interface if there is a conflict?
A) InterfaceName.super.methodName();
B) super.methodName();
C) this.methodName();
D) default.methodName();
Answer: A

11. Can interfaces with default methods be instantiated?
A) No, interfaces cannot be instantiated
B) Yes
Answer: A

12. Which modifier is NOT allowed in default method declaration?
A) private
B) public
C) default
D) abstract
Answer: D

13. Can a default method be overridden by implementing classes?
A) Yes
B) No
Answer: A

14. Which of the following statements is true about Java 8 interfaces?
A) Interfaces can have private methods
B) Interfaces can have constructors
C) Interfaces can have default and static methods
D) Interfaces can extend classes
Answer: C

15. Which of these is a limitation of default methods?
A) They cannot access instance fields
B) They cannot have a body
C) They cannot be overridden
D) They cannot be used with lambda expressions
Answer: A

16. How does default method improve interface evolution?
A) By allowing addition of new methods without breaking existing implementations
B) By removing old methods
C) By forcing all classes to implement new methods
D) None of these
Answer: A

17. Can a default method throw a checked exception?
A) Yes
B) No
Answer: A

18. Can an interface extend multiple interfaces having default methods?
A) Yes
B) No
Answer: A

19. How are abstract methods declared in Java 8 interfaces?
A) Without body and without default keyword
B) With default keyword and body
C) With static keyword
D) With final keyword
Answer: A

20. Which statement about static methods in interfaces is correct?
A) Static methods belong to the interface, not to instances
B) Static methods can be overridden by implementing classes
C) Static methods can be called through object instances
D) Static methods cannot have a body
Answer: A

21. Can a default method in an interface be marked as final?
A) Yes
B) No
Answer: B

22. Which of the following is true about the accessibility of default methods?
A) They are implicitly public
B) They can be private
C) They are package-private by default
D) They can be protected
Answer: A

23. If an implementing class provides an implementation for a default method, which version is called at runtime?
A) The class's overridden method
B) The interface's default method
C) Both methods
D) Compiler error
Answer: A

24. Can you declare a default method as synchronized?
A) Yes
B) No
Answer: A

25. How do you resolve a conflict when a class implements two interfaces with the same default method signature?
A) Override the method in the class and explicitly call one of the interface's default methods
B) No need to do anything, the compiler resolves automatically
C) Use super keyword only
D) Remove one interface
Answer: A

26. Can default methods be used with lambda expressions?
A) Yes, interfaces with default methods are still functional interfaces if they have only one abstract method
B) No

27. Which of these can interfaces NOT have in Java 8?
A) Instance variables
B) Static methods
C) Default methods
D) Abstract methods
Answer: A

28. What keyword must you use to declare a static method inside an interface?
A) static
B) default
C) abstract
D) final
Answer: A

29. Can default methods call static methods inside the same interface?
A) Yes
B) No
Answer: A

30. What happens if an interface extends two interfaces with conflicting default methods?
A) The child interface must override the conflicting method or the implementing class must

do so
B) The first interface's method is used automatically
C) The second interface's method is used automatically
D) Compilation fails without any override
Answer: A

---

*Lambda expressions

1. Lambda expressions were introduced in which Java version?
   A) Java 7
   B) Java 8
   C) Java 9
   D) Java 6
   Answer: B
2. What is a lambda expression in Java?
   A) An anonymous method (function) implementation
   B) A named method
   C) A new class type
   D) A static method
   Answer: A
3. What functional interface does a lambda expression implement?
   A) Any interface with exactly one abstract method
   B) Any interface
   C) Only Runnable interface
   D) Only Comparator interface
   Answer: A
4. Which package contains predefined functional interfaces?
   A) java.util.function
   B) java.lang
   C) java.io
   D) java.util.concurrent
   Answer: A
5. Which of the following is the correct syntax of a lambda expression?
   A) (parameters) -> expression
   B) -> (parameters) expression
   C) (parameters) <- expression
   D) function(parameters) {}
   Answer: A
6. Can lambda expressions access local variables of the enclosing scope?
   A) Yes, but only if they are final or effectively final
   B) Yes, without restrictions
   C) No
   D) Only static variables
   Answer: A
7. Which keyword is used to represent the lambda body if it contains multiple statements?
   A) Curly braces {}
   B) Parentheses ()
   C) Square brackets []
   D) Angle brackets <>
   Answer: A

8. Can a lambda expression throw checked exceptions?
   A) Yes, if declared in the functional interface method signature
   B) No
   C) Only runtime exceptions
   D) Only errors
   Answer: A
9. Which of the following is NOT a functional interface?
   A) Runnable
   B) Comparator
   C) List
   D) Callable
   Answer: C
10. How many abstract methods should a functional interface have?
   A) Exactly one
   B) None
   C) More than one
   D) Any number
   Answer: A
11. What does the following lambda expression do?
   (x, y) -> x + y
   A) Returns the sum of x and y
   B) Prints x and y
   C) Declares variables x and y
   D) None of the above
   Answer: A
12. How can you explicitly specify the data types of parameters in a lambda?
   A) (int x, int y) -> x + y
   B) (x: int, y: int) -> x + y
   C) int (x, y) -> x + y
   D) (x, y) int -> x + y
   Answer: A
13. Which of these statements about lambda expressions is true?
   A) Lambda expressions can be used to instantiate functional interfaces
   B) Lambda expressions can only be used in anonymous classes
   C) Lambda expressions cannot access instance variables
   D) Lambda expressions do not support method references
   Answer: A
14. What is the return type of a lambda expression?
   A) Same as the return type of the functional interface's abstract method
   B) Always void
   C) Always int
   D) Always Object
   Answer: A
15. Can lambda expressions have their own local variables inside the body?
   A) Yes
   B) No
   Answer: A
16. How do you represent a lambda expression with no parameters?
   A) () -> System.out.println("Hello")
   B) (void) -> System.out.println("Hello")
   C) ( ) -> System.out.println("Hello")

D) null -> System.out.println("Hello")
Answer: A

17. Which of the following is NOT a benefit of lambda expressions?
A) Reduced boilerplate code
B) Improved readability
C) Increased verbosity
D) Better support for functional programming
Answer: C

18. What is a method reference in relation to lambda expressions?
A) A shorthand notation for lambda expressions that invoke a method
B) A way to declare methods inside an interface
C) A pointer to a class
D) A static variable reference
Answer: A

19. Which keyword is used to define a functional interface explicitly?
A) @FunctionalInterface annotation
B) @Lambda annotation
C) @Interface annotation
D) No keyword is required
Answer: A

20. Which of the following is an example of a valid lambda expression assigned to a variable?
A) Runnable r = () -> System.out.println("Run");
B) Runnable r = () => System.out.println("Run");
C) Runnable r = -> System.out.println("Run");
D) Runnable r = ( ) => { System.out.println("Run"); };
Answer: A

21. Can lambda expressions capture and modify local variables of the enclosing method?
A) No, they can only capture final or effectively final variables
B) Yes, any variable can be modified
C) Only static variables can be modified
D) Yes, but only if synchronized
Answer: A

22. Which of the following is true about the target type of a lambda expression?
A) It must be a functional interface
B) It can be any class or interface
C) It must be an abstract class
D) It must be a concrete class
Answer: A

23. What happens if you use lambda expression with a non-functional interface?
A) Compilation error
B) Runtime error
C) Code runs successfully
D) Warning but runs
Answer: A

24. Which of the following interfaces is a functional interface?
A) `java.util.function.Predicate<T>`
B) `java.util.List<T>`
C) `java.lang.Runnable`
D) Both A and C
Answer: D

25. How can you pass a lambda expression as an argument?
A) By passing it to a method expecting a functional interface

B) Directly as a method parameter without any interface
C) Only by wrapping in an anonymous inner class
D) You cannot pass lambda expressions as arguments
Answer: A

---

*Functional interfaces, Built-in functional interfaces

1. What is a functional interface in Java?
   A) An interface with exactly one abstract method
   B) An interface with multiple abstract methods
   C) An interface with no methods
   D) An abstract class
   Answer: A
2. Which annotation is used to declare a functional interface?
   A) @FunctionalInterface
   B) @Interface
   C) @Override
   D) @AbstractInterface
   Answer: A
3. Can a functional interface have multiple default methods?
   A) Yes
   B) No
   Answer: A
4. What happens if you annotate an interface with @FunctionalInterface but it has more than one abstract method?
   A) Compilation error
   B) Runtime error
   C) Warning only
   D) No effect
   Answer: A
5. Which package contains built-in functional interfaces?
   A) java.util.function
   B) java.lang
   C) java.io
   D) java.util.concurrent
   Answer: A
6. Which of the following is NOT a built-in functional interface?
   A) Predicate
   B) Consumer
   C) Runnable
   D) Supplier
   Answer: C
7. What does the Predicate<T> functional interface represent?
   A) A boolean-valued function of one argument
   B) A function that returns a value
   C) A consumer of a value
   D) A supplier of values
   Answer: A
8. Which method is declared in the Predicate<T> interface?
   A) boolean test(T t)
   B) void accept(T t)

C) T get()
D) R apply(T t)
Answer: A

9. What does the Consumer<T> interface represent?
A) An operation that accepts a single input argument and returns no result
B) A function that supplies a result
C) A predicate function
D) A function that takes two arguments
Answer: A

10. Which method is declared in the Consumer<T> interface?
A) void accept(T t)
B) boolean test(T t)
C) T get()
D) R apply(T t)
Answer: A

11. What does the Supplier<T> interface do?
A) Provides a result of type T without input arguments
B) Accepts input and produces a boolean
C) Consumes a value without returning anything
D) Applies a function on two inputs
Answer: A

12. Which method is declared in Supplier<T>?
A) T get()
B) void accept(T t)
C) boolean test(T t)
D) R apply(T t)
Answer: A

13. What does the Function<T,R> interface represent?
A) A function that accepts an argument of type T and returns a result of type R
B) A boolean function
C) A supplier of values
D) A consumer of values
Answer: A

14. Which method is declared in Function<T,R>?
A) R apply(T t)
B) void accept(T t)
C) T get()
D) boolean test(T t)
Answer: A

15. Can a functional interface have static methods?
A) Yes
B) No
Answer: A

16. Which built-in functional interface represents a function with two arguments?
A) BiFunction<T,U,R>
B) Function<T,R>
C) Consumer<T>
D) Predicate<T>
Answer: A

17. What method does BiFunction<T,U,R> declare?
A) R apply(T t, U u)

B) boolean test(T t, U u)
C) void accept(T t, U u)
D) T get()
Answer: A

18. Which built-in interface is used as the target for lambda expressions that do not return a value and take no input parameters?
A) Runnable
B) Supplier
C) Consumer
D) Function
Answer: A

19. What is the functional method of Runnable interface?
A) void run()
B) void accept()
C) T get()
D) boolean test()
Answer: A

20. Which of the following functional interfaces returns no result?
A) Consumer
B) Function
C) Supplier
D) Predicate
Answer: A

21. Can a functional interface extend another functional interface?
A) Yes
B) No
Answer: A

22. What is the purpose of the default method in a functional interface?
A) To provide an implementation without affecting the functional method
B) To declare an abstract method
C) To prevent interface inheritance
D) To make the interface non-functional
Answer: A

23. Which functional interface is best suited for representing a function that returns a primitive int value?
A) IntSupplier
B) Supplier<Integer>
C) Function<T, Integer>
D) Consumer<Integer>
Answer: A

24. What is the return type of the test method in the Predicate<T> interface?
A) boolean
B) void
C) T
D) R
Answer: A

25. Which functional interface would you use if you want to represent a function taking two inputs and returning a boolean?
A) BiPredicate<T,U>
B) BiConsumer<T,U>
C) BiFunction<T,U,R>

D) Predicate<T>
Answer: A

26. The method andThen is commonly found in which functional interface?
A) Function<T,R>
B) Predicate<T>
C) Consumer<T>
D) Supplier<T>
Answer: A

27. Which interface represents an operation that takes two input arguments and returns no result?
A) BiConsumer<T,U>
B) BiFunction<T,U,R>
C) BiPredicate<T,U>
D) Consumer<T>
Answer: A

28. In the context of functional interfaces, what is “composition”?
A) Combining two functions to form a new function
B) Declaring multiple interfaces at once
C) Inheriting methods from two interfaces
D) Writing default methods
Answer: A

29. Which of these functional interfaces supports primitive specialization for int values?
A) IntConsumer
B) Consumer<Integer>
C) Function<Integer, Integer>
D) Supplier<Integer>
Answer: A

30. What is the main benefit of using built-in functional interfaces?
A) Reduces the need to declare custom functional interfaces
B) Increases code verbosity
C) Forces use of anonymous classes
D) None of the above
Answer: A

---

*Method and Constructor references.

1. What is a method reference in Java?
A) A shorthand notation of a lambda expression to call a method
B) A pointer to a variable
C) A type of class inheritance
D) A new interface type
Answer: A

2. Which symbol is used for method references in Java?
A) ::
B) ->
C) .
D) :
Answer: A

3. How many types of method references are there?
A) Four
B) Two
C) Three

D) One

Answer: A

4. Which of the following is NOT a valid type of method reference?
   A) Static method reference
   B) Instance method reference of a particular object
   C) Instance method reference of an arbitrary object of a particular type
   D) Abstract method reference
   Answer: D
5. Which syntax is used to refer to a static method of a class?
   A) ClassName::staticMethodName
   B) objectName::staticMethodName
   C) ClassName.staticMethodName()
   D) ClassName->staticMethodName
   Answer: A
6. How do you refer to an instance method of a particular object?
   A) objectName::instanceMethodName
   B) ClassName::instanceMethodName
   C) objectName.instanceMethodName()
   D) ClassName.instanceMethodName()
   Answer: A
7. How do you refer to an instance method of an arbitrary object of a particular type?
   A) ClassName::instanceMethodName
   B) objectName::instanceMethodName
   C) ClassName.instanceMethodName()
   D) objectName.instanceMethodName()
   Answer: A
8. What does constructor reference syntax look like?
   A) ClassName::new
   B) new ClassName()
   C) ClassName.new()
   D) new::ClassName
   Answer: A
9. Which functional interface is typically used with constructor references?
   A) Supplier<T>
   B) Consumer<T>
   C) Function<T,R>
   D) Predicate<T>
   Answer: A
10. Can constructor references take arguments?
    A) Yes, if the functional interface's abstract method takes arguments
    B) No
    Answer: A
11. Which method reference matches the following lambda? (s) -> s.toLowerCase()
    A) String::toLowerCase
    B) String::toUpperCase
    C) s::toLowerCase
    D) Object::toString
    Answer: A
12. Which method reference matches (x, y) -> x.equals(y)?
    A) String::equals
    B) Object::equals

C) equals::String
D) x::equals
Answer: A

13. Which method reference can replace this lambda? (s) -> System.out.println(s)?
A) System.out::println
B) System::println
C) System.out.println
D) System::out
Answer: A

14. What kind of method reference is List::size?
A) Instance method of an arbitrary object of a particular type
B) Static method reference
C) Instance method of a particular object
D) Constructor reference
Answer: A

15. Can method references be used wherever a lambda expression is expected?
A) Yes, if the method signature matches the functional interface method
B) No
Answer: A

16. What will this method reference mean? ArrayList::new?
A) Reference to the constructor of ArrayList
B) Reference to a static method in ArrayList
C) Reference to an instance method
D) Invalid syntax
Answer: A

17. Which of these can NOT be used with method references?
A) Private methods
B) Public methods
C) Static methods
D) Instance methods
Answer: A (because private methods can't be referenced outside the class)

18. Which of the following functional interfaces is suitable for a constructor reference that takes one argument and returns an object?
A) Function<T,R>
B) Consumer<T>
C) Supplier<T>
D) Predicate<T>
Answer: A

19. What is the advantage of method references over lambda expressions?
A) More concise and readable code
B) More powerful than lambdas
C) They allow private method access
D) No advantage
Answer: A

20. Which of the following is a valid method reference to a static method?
A) Math::max
B) max::Math
C) Math.max()
D) Math->max
Answer: A

21. Which functional interface would best match this constructor reference: Person::new where Person has a constructor with two parameters (String name, int age)?
A) BiFunction<String, Integer, Person>
B) Function<String, Person>
C) Supplier<Person>
D) Consumer<Person>
Answer: A

22. What kind of method reference is used in System.out::println?
A) Instance method reference of a particular object
B) Static method reference
C) Constructor reference
D) Instance method reference of an arbitrary object
Answer: A

23. Can method references be chained or composed directly like lambda expressions?
A) No, but you can compose the functional interfaces that use them
B) Yes, directly with :: operator
C) Only static method references can be chained
D) Only constructor references can be chained
Answer: A

24. Which method reference type does the syntax String::valueOf represent?
A) Static method reference
B) Instance method reference
C) Constructor reference
D) Arbitrary object instance method reference
Answer: A

25. What will the method reference this::toString refer to inside a class?
A) Instance method reference of a particular object (this)
B) Static method reference
C) Constructor reference
D) Arbitrary instance method reference
Answer: A

26. Which of these is NOT true about constructor references?
A) They always invoke a constructor
B) They can match functional interfaces with matching parameters
C) They can be assigned to functional interfaces with any method signature
D) They improve code readability
Answer: C

27. What is the functional interface type for a constructor reference that takes no arguments?
A) Supplier<T>
B) Function<T,R>
C) Consumer<T>
D) Predicate<T>
Answer: A

28. Which method reference can replace the lambda expression (str) -> str.length()?
A) String::length
B) str::length
C) Object::toString
D) String::size
Answer: A

29. Which statement is true about using method references with private methods?
A) They cannot be used outside their class

B) They can be used anywhere like public methods
C) They can be used only inside anonymous classes
D) They cannot be used at all
Answer: A

30. Which of the following is an example of an instance method reference of an arbitrary object of a particular type?
A) String::toUpperCase
B) myObject::toString
C) ClassName::new
D) System.out::println
Answer: A

---

*Java.util.Stream, stream operations & executions

1. Which package contains the Stream API in Java?
A) java.util.stream
B) java.util
C) java.io
D) java.stream
Answer: A

2. What is a stream in Java?
A) A sequence of elements supporting sequential and parallel aggregate operations
B) A file handling mechanism
C) A method to read bytes
D) A type of thread
Answer: A

3. Which method is used to create a stream from a Collection?
A) collection.stream()
B) Stream.of(collection)
C) Stream.create(collection)
D) collection.createStream()
Answer: A

4. What type of operations are intermediate operations in streams?
A) Lazy operations that return another stream
B) Terminal operations that produce results or side-effects
C) Blocking operations
D) Parallel operations only
Answer: A

5. Which of the following is a terminal operation in streams?
A) forEach()
B) filter()
C) map()
D) sorted()
Answer: A

6. What does the filter() method do in a stream pipeline?
A) Selects elements that match a given predicate
B) Transforms elements
C) Sorts elements
D) Collects elements into a collection
Answer: A

7. Which method is used to convert a stream back into a collection?
   A) collect(Collectors.toList())
   B) toList()
   C) asList()
   D) toCollection()
   Answer: A
8. What is the return type of map() operation on a stream?
   A) Another stream with mapped elements
   B) A collection
   C) A list
   D) Void
   Answer: A
9. What does the reduce() method do?
   A) Aggregates elements of a stream into a single result
   B) Filters elements
   C) Sorts elements
   D) Converts stream to array
   Answer: A
10. Which of these is a lazy operation?
    A) map()
    B) forEach()
    C) count()
    D) reduce()
    Answer: A
11. What happens when a terminal operation is called on a stream?
    A) The entire stream pipeline is executed
    B) Nothing, it remains lazy
    C) Only the first element is processed
    D) Stream is closed without processing
    Answer: A
12. Can a stream be reused after a terminal operation?
    A) No, streams cannot be reused after a terminal operation
    B) Yes, streams can be reused multiple times
    C) Only if they are parallel streams
    D) Only if explicitly reset
    Answer: A
13. Which method creates an infinite stream?
    A) Stream.iterate()
    B) Stream.of()
    C) Stream.generate()
    D) Both A and C
    Answer: D
14. What does flatMap() do in a stream?
    A) Flattens nested streams into a single stream
    B) Maps elements without flattening
    C) Filters elements
    D) Collects elements
    Answer: A
15. What is the main difference between findFirst() and findAny()?
    A) findFirst() returns the first element, findAny() may return any element, useful in parallel streams

B) Both are exactly the same
C) findAny() always returns the last element
D) findFirst() works only on sorted streams
Answer: A

16. Which stream operation can be used to sort elements?
A) sorted()
B) filter()
C) map()
D) reduce()
Answer: A

17. How do you create a parallel stream from a collection?
A) collection.parallelStream()
B) Stream.parallel(collection)
C) Stream.ofParallel(collection)
D) collection.stream().parallel()
Answer: A

18. Which of the following is NOT a feature of streams?
A) Streams do not store elements
B) Streams are immutable
C) Streams support internal iteration
D) Streams can be reused multiple times
Answer: D

19. Which method terminally triggers a stream to count its elements?
A) count()
B) size()
C) length()
D) countElements()
Answer: A

20. Which functional interface is used with filter() method?
A) Predicate<T>
B) Consumer<T>
C) Supplier<T>
D) Function<T,R>
Answer: A

21. What does the distinct() operation do in a stream pipeline?
A) Removes duplicate elements
B) Filters elements by predicate
C) Sorts the elements
D) Converts stream to a list
Answer: A

22. Which terminal operation collects elements into a Map?
A) collect(Collectors.toMap())
B) toMap()
C) map()
D) groupBy()
Answer: A

23. What is the difference between map() and flatMap()?
A) map() applies a function and returns a stream of results; flatMap() flattens streams into a single stream
B) map() filters elements; flatMap() sorts elements
C) flatMap() returns a list; map() returns a stream

D) No difference
Answer: A

24. Which of these operations can be parallelized easily?
A) Stateless intermediate operations like map(), filter()
B) Stateful intermediate operations like sorted()
C) Terminal operations only
D) None of the above
Answer: A

25. What is the return type of the peek() method in a stream?
A) Stream — it allows performing an action on each element without modifying the stream
B) Void
C) List
D) Array
Answer: A

26. Which stream operation is used for debugging purposes?
A) peek()
B) forEach()
C) filter()
D) map()
Answer: A

27. How can you create a stream from an array?
A) Arrays.stream(array)
B) Stream.of(array)
C) Both A and B
D) None of the above
Answer: C

28. What is the characteristic of a short-circuiting terminal operation?
A) It may not process all elements
B) It processes all elements always
C) It modifies the source
D) It returns void
Answer: A

29. Which of the following is a short-circuiting operation?
A) anyMatch()
B) allMatch()
C) noneMatch()
D) All of the above
Answer: D

30. Which of these is a reduction operation?
A) reduce()
B) filter()
C) map()
D) peek()
Answer: A

31. What is the output of Stream.of(1, 2, 3).map(x -> x * 2).collect(Collectors.toList())?
A) [2, 4, 6]
B) [1, 2, 3]
C) [1, 4, 9]
D) []
Answer: A

32. Which collector groups elements according to a classifier function?
A) Collectors.groupingBy()
B) Collectors.toList()
C) Collectors.toSet()
D) Collectors.partitioningBy()
Answer: A

33. What does the sorted(Comparator) method return?
A) A sorted stream according to the comparator
B) A sorted list
C) A new array
D) Nothing
Answer: A

34. What is the default ordering of the sorted() method without parameters?
A) Natural ordering of elements (must implement Comparable)
B) Reverse ordering
C) No ordering
D) Random ordering
Answer: A

35. What does Collectors.partitioningBy() return?
A) A Map<Boolean, List<T>> partitioning elements into two groups based on a predicate
B) A list of elements
C) A map of grouped elements
D) A stream
Answer: A

36. What is the difference between collect() and reduce()?
A) collect() is mutable reduction (e.g., collecting into collections), reduce() is immutable reduction into a single value
B) No difference
C) reduce() collects elements into a collection
D) collect() aggregates into a single value
Answer: A

37. How do you create an empty stream?
A) Stream.empty()
B) Stream.of()
C) new Stream()
D) null
Answer: A

38. Which method is used to convert a stream into an array?
A) toArray()
B) collect()
C) array()
D) asArray()
Answer: A

39. What is the purpose of the limit() method in a stream?
A) To truncate the stream to a maximum size
B) To filter elements by size
C) To sort elements
D) To collect elements into a list
Answer: A

40. What is a characteristic of an unordered stream?
A) Operations may return results in any order

B) The stream maintains insertion order
C) Stream elements are sorted
D) Stream only supports sequential execution
Answer: A

Session 30:

*Advanced java.util.Stream operations (e.g. reduce(), flatMap(), etc)

1. What does the reduce() method in Java Streams do?
   A) Combines stream elements into a single result using an associative accumulation function
   B) Filters elements based on a condition
   C) Transforms each element in the stream
   D) Sorts the stream elements
   Answer: A
2. Which of these is a valid signature for the reduce() method?
   A) Optional<T> reduce(BinaryOperator<T> accumulator)
   B) T reduce(T identity, BinaryOperator<T> accumulator)
   C) Both A and B
   D) None of the above
   Answer: C
3. What does the flatMap() operation do on a Stream?
   A) Transforms each element into a Stream and flattens the result into a single Stream
   B) Filters elements based on a predicate
   C) Reduces elements into a single result
   D) Sorts elements in natural order
   Answer: A
4. Which functional interface is used as a parameter to reduce()?
   A) BinaryOperator<T>
   B) Predicate<T>
   C) Consumer<T>
   D) Supplier<T>
   Answer: A
5. How is the reduce() method different from collect()?
   A) reduce() is for immutable reduction to a single value; collect() is for mutable reduction to a container
   B) Both do the same thing
   C) collect() reduces to a single value only
   D) reduce() collects into a collection
   Answer: A
6. What is the result of Stream.of(1, 2, 3, 4).reduce(0, (a, b) -> a + b)?
   A) 10
   B) 24
   C) 0
   D) null
   Answer: A

7. Which method is best to convert a Stream<Stream<T>> to Stream<T>?
   A) flatMap()
   B) map()
   C) reduce()
   D) filter()
   Answer: A
8. What does the following expression return?
   Stream.of("a", "bb", "ccc").map(String::length).reduce(0, Integer::sum)
   A) Sum of the lengths: 6
   B) Concatenated string "abbccc"
   C) List of lengths [1, 2, 3]
   D) An error
   Answer: A
9. Which of the following is true about reduce() without an identity value?
   A) Returns an Optional<T> because stream might be empty
   B) Always returns a value
   C) Throws an exception for empty streams
   D) Returns null for empty streams
   Answer: A
10. Can flatMap() be used to flatten a List<List<String>> into a Stream<String>?
    A) Yes
    B) No
    C) Only with map()
    D) Only with reduce()
    Answer: A
11. What is the purpose of the identity value in reduce(identity, accumulator)?
    A) Initial value and neutral element of the accumulator
    B) Starting point for filtering
    C) Starting point for mapping
    D) None of the above
    Answer: A
12. What is the difference between map() and flatMap()?
    A) map() transforms each element to one output element; flatMap() transforms each element into a stream and then flattens
    B) No difference
    C) map() flattens streams; flatMap() does not
    D) map() filters elements; flatMap() maps elements
    Answer: A
13. Which of the following is NOT a characteristic of the accumulator function in reduce()?
    A) Must be associative
    B) Should be stateless
    C) Must be commutative
    D) Must return void
    Answer: D
14. Which operation returns a stream of elements without duplicates?
    A) distinct()
    B) reduce()
    C) flatMap()
    D) filter()
    Answer: A

15. What is the output of the following?
Stream.of(1, 2, 3).flatMap(i -> Stream.of(i, i * 10)).collect(Collectors.toList())
A) [1, 10, 2, 20, 3, 30]
B) [1, 2, 3]
C) [10, 20, 30]
D) []
Answer: A

16. What happens if you pass a non-associative accumulator function to reduce() in a parallel stream?
A) Results can be unpredictable or incorrect
B) It throws an exception
C) It runs sequentially instead
D) It automatically fixes the function
Answer: A

17. Which stream operation is useful to transform and flatten a nested structure?
A) flatMap()
B) map()
C) filter()
D) reduce()
Answer: A

18. What is the result type of reduce() with identity and accumulator?
A) The same type as the stream element
B) Always Optional
C) List
D) Void
Answer: A

19. Which of the following is a valid way to sum all elements of an IntStream?
A) intStream.reduce(0, Integer::sum)
B) intStream.sum()
C) Both A and B
D) None of the above
Answer: C

20. What does this code do?
Stream.of("Java", "Stream").flatMap(s ->
Arrays.stream(s.split(""))).distinct().sorted().forEach(System.out::print);
A) Prints unique letters in alphabetical order: JSartemv
B) Prints original strings
C) Prints sorted strings
D) Prints the first letters only
Answer: A

21. Which method is commonly used to combine elements in a stream into a single string with a delimiter?
A) collect(Collectors.joining(","))
B) reduce()
C) map()
D) flatMap()
Answer: A

22. How does reduce() behave in parallel streams?
A) The accumulator function must be associative and stateless for correct parallel execution
B) It always runs sequentially
C) The order of execution is guaranteed

D) It cannot be used with parallel streams
Answer: A

23. What is the output type of reduce(BinaryOperator<T> accumulator)?
A) Optional<T>
B) T
C) List<T>
D) Stream<T>
Answer: A

24. Which operation can be used to transform each element into zero or more elements?
A) flatMap()
B) map()
C) filter()
D) reduce()
Answer: A

25. What will happen if you use reduce() without providing an identity on an empty stream?
A) Returns Optional.empty()
B) Returns null
C) Throws NoSuchElementException
D) Returns default zero value
Answer: A

26. Which of the following operations triggers execution of the stream pipeline?
A) Terminal operations like collect(), reduce(), forEach()
B) Intermediate operations like map(), filter()
C) None, streams are always eager
D) All operations trigger execution immediately
Answer: A

27. Can flatMap() be used to flatten a nested List structure?
A) Yes, it can flatten List<List<T>> into Stream<T>
B) No, only map() can do this
C) Only with reduce()
D) Only with filter()
Answer: A

28. Which method would you use to process elements lazily and return a stream of results?
A) Intermediate operations like map(), filter(), flatMap()
B) Terminal operations like forEach()
C) Collectors
D) None of the above
Answer: A

29. What is the purpose of Optional in reduce() without an identity?
A) To safely handle the possibility of an empty stream
B) To force an exception on empty streams
C) To always provide a default value
D) To allow parallel processing only
Answer: A

30. Which method on Stream interface supports sequential and parallel execution?
A) reduce()
B) count()
C) filter()
D) All of the above
Answer: D

31. In a stream pipeline, where should you place filter() for optimal performance?
A) Before mapping or flatMapping operations to reduce elements early
B) At the end of the pipeline
C) It doesn't matter
D) After terminal operations
Answer: A

32. What does the collect() method require as an argument?
A) A Collector which implements how to accumulate elements
B) A Function
C) A Predicate
D) A Consumer
Answer: A

33. How does reduce() differ from collect() when used with parallel streams?
A) reduce() requires associative and stateless accumulator, collect() can be used with mutable containers
B) collect() requires associative accumulator
C) They behave exactly the same
D) Neither works with parallel streams
Answer: A

34. What does this code return?
Stream.of(1, 2, 3, 4).reduce((a, b) -> a > b ? a : b)
A) Maximum value wrapped in Optional: Optional[4]
B) Minimum value
C) Sum of all values
D) Null
Answer: A

35. What is the expected behavior of this snippet?
Stream.of("a", "bb", "ccc").flatMap(s -> Stream.of(s.split(""))).count()
A) Returns total count of characters: 6
B) Returns number of strings: 3
C) Throws an error
D) Returns zero
Answer: A

*Reflection concepts

- What is Java Reflection used for?
A) To inspect and manipulate classes, methods, fields at runtime
B) To perform file input/output
C) To manage threads
D) To compile code
Answer: A
- Which package contains the core reflection classes?
A) java.lang.reflect
B) java.lang.io
C) java.util.reflect
D) java.io
Answer: A
- Which class is used to get metadata about a class at runtime?
A) Class
B) Object

C) Method
D) Field
Answer: A

- How do you obtain the Class object for a class named MyClass?
  A) MyClass.class
  B) new MyClass().getClass()
  C) Class.forName("MyClass")
  D) All of the above
  Answer: D
- Which method allows you to invoke a method on an object via reflection?
  A) Method.invoke(Object obj, Object... args)
  B) Class.invoke()
  C) Object.invoke()
  D) Field.invoke()
  Answer: A
- What does the getDeclaredMethods() method return?
  A) All methods declared in the class, including private ones
  B) Only public methods
  C) Only inherited methods
  D) Only static methods
  Answer: A
- How can you access a private field using reflection?
  A) Call setAccessible(true) on the Field object
  B) You cannot access private fields
  C) Use getField() instead of getDeclaredField()
  D) Use Class.getFields()
  Answer: A
- What exception might be thrown if you try to get a class by name and it does not exist?
  A) ClassNotFoundException
  B) IOException
  C) NoSuchFieldException
  D) IllegalAccessException
  Answer: A
- What is the purpose of Field.set(Object obj, Object value)?
  A) To set the value of a field on the specified object
  B) To get the field value
  C) To declare a new field
  D) To delete a field
  Answer: A
- Which reflection class represents a method?
  A) java.lang.reflect.Method
  B) java.lang.reflect.Field
  C) java.lang.reflect.Constructor
  D) java.lang.Class
  Answer: A
- What does Class.forName("java.lang.String") return?
  A) Class object for java.lang.String
  B) An instance of String
  C) A new String object
  D) null
  Answer: A

- How to get all constructors of a class including private ones?
  A) Class.getDeclaredConstructors()
  B) Class.getConstructors()
  C) Class.getConstructor()
  D) Class.getDeclaredMethods()
  Answer: A
- Which method is used to create a new instance of a class via reflection?
  A) Constructor.newInstance()
  B) Class.newInstance()
  C) Both A and B (with Class.newInstance() deprecated since Java 9)
  D) Object.newInstance()
  Answer: C
- How to access the modifiers (like public, private) of a method or field?
  A) Using getModifiers() method
  B) Using getName() method
  C) Using getType() method
  D) Using toString() method
  Answer: A
- Which exception is thrown if you try to invoke a method on a null object reference?
  A) NullPointerException
  B) InvocationTargetException
  C) IllegalAccessException
  D) ClassNotFoundException
  Answer: A
- What does the isAccessible() method check in reflection?
  A) Whether the reflected object (field, method) is accessible
  B) Whether the class is public
  C) Whether the method is static
  D) Whether the field is final
  Answer: A
- What is the difference between getMethod() and getDeclaredMethod()?
  A) getMethod() returns only public methods including inherited ones, getDeclaredMethod() returns all declared methods regardless of access modifier
  B) No difference
  C) getMethod() returns private methods only
  D) getDeclaredMethod() returns only inherited methods
  Answer: A
- Can you change the value of a final field via reflection?
  A) Generally no, but with setAccessible(true) and some JVM tricks, it's possible
  B) Yes, always
  C) No, never
  D) Only if the field is static
  Answer: A
- Which exception indicates an error while invoking a method using reflection?
  A) InvocationTargetException
  B) ClassNotFoundException
  C) IllegalAccessException
  D) NoSuchMethodException
  Answer: A
- How can you find out if a class implements a particular interface at runtime?
  A) Using Class.isAssignableFrom() method

B) Using instanceof operator
C) Using getInterfaces() and checking the array
D) Both A and C
Answer: D

- Which method returns all interfaces implemented by a class?
  A) Class.getInterfaces()
  B) Class.getSuperClass()
  C) Class.getDeclaredMethods()
  D) Class.getFields()
  Answer: A
- What does the getSuperclass() method return?
  A) The superclass of the class or null if none exists
  B) All parent classes in hierarchy
  C) Interfaces implemented by the class
  D) The Object class always
  Answer: A
- How can you find out the parameter types of a method using reflection?
  A) Using Method.getParameterTypes()
  B) Using Class.getParameters()
  C) Using Field.getType()
  D) Using Method.getReturnType()
  Answer: A
- What exception will be thrown if you try to access a non-existent method using getMethod()?
  A) NoSuchMethodException
  B) ClassNotFoundException
  C) IllegalAccessException
  D) InvocationTargetException
  Answer: A
- How can you get the return type of a method in reflection?
  A) Method.getReturnType()
  B) Method.getReturn()
  C) Method.getType()
  D) Class.getReturnType()
  Answer: A
- Which reflection method is used to check if a method or field is static?
  A) Check if (modifiers & Modifier.STATIC) != 0 using getModifiers()
  B) Method.isStatic()
  C) Field.isStatic()
  D) Class.isStatic()
  Answer: A
- What is InvocationTargetException used for?
  A) It wraps exceptions thrown by an invoked method or constructor
  B) Indicates a failure to find the target method
  C) Indicates illegal access
  D) Used for class loading issues
  Answer: A
- Can constructors be accessed and invoked using reflection?
  A) Yes, via Constructor class methods
  B) No, constructors are not accessible via reflection
  C) Only default constructors

D) Only public constructors
Answer: A

- What is the use of getDeclaredFields() method?
  A) Returns all fields declared by the class regardless of access modifier
  B) Returns only public fields
  C) Returns inherited fields
  D) Returns static fields only
  Answer: A
- Is it possible to change method behavior at runtime using reflection?
  A) No, reflection allows inspection and invocation but not modification of method code
  B) Yes, by modifying bytecode
  C) Yes, directly via Method class
  D) Yes, by changing method signature
  Answer: A

---

*Invoking methods dynamically, Assigning values to private field

1. Which reflection method allows you to invoke a method dynamically on an object?
   A) Method.invoke(Object obj, Object... args)
   B) Class.invoke()
   C) Object.call()
   D) Field.invoke()
   Answer: A
2. To invoke a private method using reflection, what must you do first?
   A) Call setAccessible(true) on the Method object
   B) Use getMethod() instead of getDeclaredMethod()
   C) Nothing special, private methods can be invoked directly
   D) Use a proxy class
   Answer: A
3. What exception must be handled or declared when invoking a method reflectively?
   A) InvocationTargetException
   B) IOException
   C) ClassNotFoundException
   D) NoSuchFieldException
   Answer: A
4. How do you get a Method object for a method named foo with parameter types (int, String) in class MyClass?
   A) MyClass.class.getDeclaredMethod("foo", int.class, String.class)
   B) MyClass.class.getMethod("foo")
   C) MyClass.getMethod("foo", int.class, String.class)
   D) MyClass.class.getMethod("foo", Object.class)
   Answer: A
5. What exception is thrown if the method invoked throws an exception internally?
   A) InvocationTargetException
   B) IllegalAccessException
   C) NoSuchMethodException
   D) ClassCastException
   Answer: A
6. Which exception occurs if you try to invoke a method on a null object reference?
   A) NullPointerException
   B) InvocationTargetException

C) IllegalAccessException
D) NoSuchMethodException
Answer: A

7. How to assign a value to a private field named value in an object obj using reflection?
A) Field f = obj.getClass().getDeclaredField("value"); f.setAccessible(true); f.set(obj, newValue);
B) obj.value = newValue;
C) Field.set(obj, "value", newValue);
D) obj.setField("value", newValue);
Answer: A

8. Which method is used to access a private field for modification?
A) Field.setAccessible(true)
B) Field.get()
C) Field.invoke()
D) Field.modify()
Answer: A

9. What exception is thrown when you try to set a field value but the field is final?
A) Usually IllegalAccessException or may silently fail depending on JVM
B) IllegalArgumentException
C) NoSuchFieldException
D) ClassCastException
Answer: A

10. Which method throws NoSuchFieldException?
A) Class.getDeclaredField(String name) if field not found
B) Field.set()
C) Class.getMethod()
D) Method.invoke()
Answer: A

11. Can you invoke static methods using reflection?
A) Yes, by passing null as the object parameter to Method.invoke()
B) No, only instance methods can be invoked
C) Yes, but only public ones
D) No, static methods need direct invocation
Answer: A

12. What happens if you call setAccessible(false) on a Field?
A) You lose the ability to access private/protected members via reflection
B) Field becomes permanently accessible
C) Throws exception
D) Nothing
Answer: A

13. Which of these will get you a private method named calculate with no parameters?
A) getDeclaredMethod("calculate")
B) getMethod("calculate")
C) getDeclaredMethods()
D) getMethods()
Answer: A

14. When invoking a method with varargs using reflection, how should the arguments be passed?
A) Pass an Object array matching the method signature
B) Pass individual arguments separately
C) Use a Collection

D) Only primitive types allowed
Answer: A

15. Which exception occurs if the method signature is incorrect when invoking?
A) IllegalArgumentException
B) NoSuchMethodException
C) ClassNotFoundException
D) IllegalAccessException
Answer: A

16. What is the default accessibility of fields and methods via reflection?
A) Only public members are accessible unless setAccessible(true) is used
B) All members accessible by default
C) Private members are accessible by default
D) None accessible
Answer: A

17. Which method returns all declared fields of a class including private fields?
A) Class.getDeclaredFields()
B) Class.getFields()
C) Class.getDeclaredMethods()
D) Class.getMethods()
Answer: A

18. How do you invoke a method named process with one integer argument?
A) method.invoke(obj, 5)
B) method.invoke(obj, new Integer(5))
C) Both A and B
D) method.invoke(5)
Answer: C

19. Which exception is thrown if you try to access a field without calling setAccessible(true) and the field is private?
A) IllegalAccessException
B) NoSuchFieldException
C) InvocationTargetException
D) NullPointerException
Answer: A

20. Can you access private fields of a superclass via reflection from subclass object?
A) Yes, but only if you use getDeclaredField() on superclass and setAccessible(true)
B) No, private fields are not accessible
C) Only public fields of superclass are accessible
D) Only with special JVM flags
Answer: A

21. How do you get a Field object for a private field count?
A) obj.getClass().getDeclaredField("count")
B) obj.getClass().getField("count")
C) Field.get("count")
D) obj.getField("count")
Answer: A

22. What exception is thrown if you try to invoke a method that does not exist?
A) NoSuchMethodException
B) ClassNotFoundException
C) IllegalAccessException
D) InvocationTargetException
Answer: A

23. How to invoke a method dynamically when the method has no parameters?
A) method.invoke(obj)
B) method.invoke(obj, null)
C) method.invoke()
D) Both A and B
Answer: D

24. Which method retrieves the value of a field reflectively?
A) Field.get(Object obj)
B) Field.set(Object obj, Object value)
C) Class.getField()
D) Object.getField()
Answer: A

25. What does setAccessible(true) do?
A) It suppresses Java language access checking for reflected objects
B) Makes a private method public permanently
C) Changes the source code of the class
D) Throws an exception if field is private
Answer: A

26. Can you modify a final field's value via reflection?
A) Usually no, but sometimes possible with setAccessible(true) and unsafe operations
B) Yes, always
C) No, never
D) Only for static fields
Answer: A

27. What will happen if you try to invoke a static method with a non-null object reference?
A) The object reference is ignored and method is invoked correctly
B) Throws IllegalArgumentException
C) Throws NullPointerException
D) Runtime error
Answer: A

28. How can you get the list of all methods including private ones of a class?
A) getDeclaredMethods()
B) getMethods()
C) getAllMethods()
D) getClassMethods()
Answer: A

29. What exception can be thrown if method invocation fails because the underlying method throws an exception?
A) InvocationTargetException
B) IllegalAccessException
C) NoSuchMethodException
D) SecurityException
Answer: A

30. Is it possible to invoke constructors dynamically using reflection?
A) Yes, using Constructor.newInstance()
B) No, constructors can’t be invoked dynamically
C) Only default constructors
D) Only public constructors
Answer: A

31. Which method can you use to retrieve a private constructor?
A) Class.getDeclaredConstructor()

B) Class.getConstructor()
C) Class.getConstructors()
D) Class.getDeclaredConstructors()
Answer: A

32. How do you invoke a constructor reflectively with parameters?
A) Constructor.newInstance(Object... initargs)
B) Class.newInstance()
C) Constructor.invoke()
D) Object.newInstance()
Answer: A

33. What exception is thrown if you try to instantiate an abstract class via reflection?
A) InstantiationException
B) IllegalAccessException
C) ClassNotFoundException
D) NoSuchMethodException
Answer: A

34. Which reflection method provides information about method parameter names (Java 8+)?
A) Method.getParameters()
B) Method.getParameterTypes()
C) Class.getParameters()
D) Method.getParameterNames()
Answer: A

35. What is required to successfully change a private field's value reflectively in a security manager environment?
A) Proper permissions granted to reflection operations
B) Nothing special
C) JVM flag only
D) Use deprecated APIs
Answer: A

---

*Annotation concept, retention policies

1. What is the purpose of annotations in Java?
A) To provide metadata information to the compiler and runtime
B) To write comments in code
C) To replace Java interfaces
D) To create classes
Answer: A

2. Which meta-annotation is used to specify how long an annotation is retained?
A) @Retention
B) @Target
C) @Documented
D) @Inherited
Answer: A

3. Which retention policy indicates the annotation is discarded by the compiler and not available at runtime?
A) RetentionPolicy.SOURCE
B) RetentionPolicy.CLASS
C) RetentionPolicy.RUNTIME

D) None of these
Answer: A

4. Which retention policy makes annotations available in the class file but not at runtime?
A) RetentionPolicy.CLASS
B) RetentionPolicy.SOURCE
C) RetentionPolicy.RUNTIME
D) RetentionPolicy.PERSISTENT
Answer: A

5. If an annotation has retention policy RUNTIME, where is it accessible?
A) Available to JVM during runtime via reflection
B) Only during compilation
C) Not available at all
D) Available only in source code
Answer: A

6. Which meta-annotation specifies where an annotation can be applied (e.g., method, field, type)?
A) @Target
B) @Retention
C) @Documented
D) @Inherited
Answer: A

7. What happens if an annotation does not specify any retention policy?
A) Default is RetentionPolicy.CLASS
B) Default is RetentionPolicy.RUNTIME
C) Default is RetentionPolicy.SOURCE
D) Compile error
Answer: A

8. Which retention policy should be used if you want to access annotation info via reflection at runtime?
A) RetentionPolicy.RUNTIME
B) RetentionPolicy.CLASS
C) RetentionPolicy.SOURCE
D) RetentionPolicy.PERSISTENT
Answer: A

9. Can annotations themselves be annotated?
A) Yes, annotations can be annotated with meta-annotations
B) No
C) Only custom annotations
D) Only JDK provided annotations
Answer: A

10. Which annotation indicates that an annotation should be documented by javadoc and similar tools?
A) @Documented
B) @Retention
C) @Target
D) @Inherited
Answer: A

11. What does the @Inherited annotation indicate?
A) An annotation on a superclass is inherited by subclasses
B) Annotation is available only in superclass
C) Annotation is deprecated
D) Annotation applies only to interfaces
Answer: A

12. Which retention policy is best suited for annotations used by code analysis tools only?
A) RetentionPolicy.SOURCE
B) RetentionPolicy.CLASS
C) RetentionPolicy.RUNTIME
D) None
Answer: A

13. How can you access an annotation at runtime?
A) Using reflection APIs such as Class.getAnnotation()
B) Using Class.getMethods()
C) Using Class.getFields()
D) Using Class.getDeclaredConstructors()
Answer: A

14. What is the default target if an annotation does not specify @Target?
A) All program elements (class, method, field, etc.)
B) Only classes
C) Only methods
D) Only fields
Answer: A

15. Which annotation would you use to restrict an annotation to only methods?
A) @Target(ElementType.METHOD)
B) @Retention(RetentionPolicy.METHOD)
C) @Inherited
D) @Documented
Answer: A

16. What is the purpose of the @Repeatable annotation?
A) Allows an annotation to be applied multiple times to the same element
B) Marks an annotation as deprecated
C) Indicates an annotation is only for runtime
D) Specifies annotation retention policy
Answer: A

17. Which annotation element type can hold other annotations?
A) Annotation type
B) String
C) int
D) Class
Answer: A

18. Which element is mandatory in a custom annotation?
A) None, all elements can have default values
B) value() if no default is provided
C) name() always

D) type() always
Answer: B

19. How do you specify a default value for an annotation element?
A) Using default keyword in annotation definition
B) Using @Default annotation
C) Using constructor
D) No default allowed
Answer: A

20. Which annotation in Java SE marks a method that overrides a superclass method?
A) @Override
B) @Deprecated
C) @SuppressWarnings
D) @Retention
Answer: A

21. The annotation @Deprecated has which retention policy by default?
A) RetentionPolicy.CLASS
B) RetentionPolicy.SOURCE
C) RetentionPolicy.RUNTIME
D) None
Answer: B

22. What does @SuppressWarnings do?
A) Instructs the compiler to ignore specific warnings
B) Suppresses runtime exceptions
C) Disables errors in code
D) Disables logging
Answer: A

23. Which annotation is used to generate documentation for annotations?
A) @Documented
B) @Retention
C) @Target
D) @Override
Answer: A

24. Which of the following is NOT a valid ElementType for @Target?
A) CONSTRUCTOR
B) PACKAGE
C) METHOD
D) VARIABLE
Answer: B (Package is valid, but spelled PACKAGE is valid, so trick question, all are valid except if mistyped)

25. *(Clarification: Actually, PACKAGE is valid. So no trick. Let's correct)*
26. Corrected 9:

27. Which of the following **is** a valid ElementType for @Target?
A) TYPE_PARAMETER
B) FILE
C) METHOD

D) PROJECT
Answer: C

28. How can you define an annotation that applies only to fields and methods?
A) @Target({ElementType.FIELD, ElementType.METHOD})
B) @Target(ElementType.ALL)
C) @Retention(RetentionPolicy.CLASS)
D) @Target(ElementType.PARAMETER)
Answer: A

29. Which of these is NOT a Java built-in annotation?
A) @FunctionalInterface
B) @Synchronized
C) @Deprecated
D) @Override
Answer: B

30. What is the retention policy of the built-in annotation @FunctionalInterface?
A) RetentionPolicy.CLASS
B) RetentionPolicy.RUNTIME
C) RetentionPolicy.SOURCE
D) None
Answer: C

31. Which of the following can be used to mark an annotation that can only be applied to interfaces?
A) @Target(ElementType.TYPE) and check interface at runtime
B) @Target(ElementType.INTERFACE)
C) @Inherited
D) @Retention(RetentionPolicy.RUNTIME)
Answer: A

32. What is the use of @Repeatable annotation?
A) To specify that the annotation type can be applied multiple times to the same declaration
B) To repeat a method call
C) To mark deprecated annotations
D) To suppress warnings
Answer: A

33. What meta-annotation is used to declare an annotation type?
A) @interface keyword, no meta-annotation needed
B) @Annotation
C) @DeclareAnnotation
D) @MetaAnnotation
Answer: A

Session 17, 18 & 19: Generics, Collection Framework (List, Set, Map), Sorting

171. What is the most significant consequence of the Java compiler performing type erasure on generic collections like `List<String>`?

A. The type information is available at runtime, allowing for dynamic type checking.

B. It enables backward compatibility, allowing generic code to interoperate with legacy code that does not use generics.

C. It enhances performance by reducing the memory footprint of collection objects.

D. It allows for the creation of arrays of generic types, such as `new List<String>[10]`.

Ans: B) It enables backward compatibility, allowing generic code to interoperate with legacy code that does not use generics.

Session 22 & 23: Multithreaded programming, Thread synchronization

172. A method in a multi-threaded application needs to increment a shared counter. Which approach provides thread safety for the increment operation but allows other threads to read the counter's value concurrently without blocking?

A. Using an AtomicInteger and its incrementAndGet() method.

B. Using a synchronized method to read and write the counter.

C. Declaring the integer counter as `volatile`.

D. Using a ReentrantLock to guard the counter.

Ans : A) Using an AtomicInteger and its incrementAndGet() method

173. Is SimpleDateFormat thread safe ?

a) True
b) False

Ans: B) False

Session 28 & 29: Java 8 Interfaces (default methods), Lambda expressions, Functional interfaces, Streams

174. A class implements two interfaces, `InterfaceA` and `InterfaceB`. Both interfaces declare a default method `void log()`. What must the implementing class do to compile successfully?

A. The class must be declared abstract.

B. The class does not need to do anything; the `log()` method from `InterfaceA` will be used by default.

C. The class must override the `log()` method and provide its own implementation.

D. The class can choose which default method to use with the `super` keyword, for instance, `InterfaceB.super.log()` inside another method.

Ans: C) The class must override the `log()` method and provide its own implementation.

175. What does LocalTime represent?

a) Date without time
b) Time without Date
c) Date and Time
d) Date and Time with timezone

Ans: B) Time without Date

Session 30: Advanced Streams, Reflection, Annotation

176. You have a `final` class `LegacySystem`. You cannot modify its source code, but you need to add new functionality to it before passing its objects to a modern framework. Which design pattern would be most appropriate?

A. Inheritance, by creating a new class that extends `LegacySystem`.

B. The Singleton Pattern, to ensure only one instance of `LegacySystem` exists.

C. Creating an abstract class that `LegacySystem` can implement.

D. The Decorator Pattern, by creating a wrapper that implements the same interface.

Ans: D) The Decorator Pattern, by creating a wrapper that implements the same interface.

177. Which method returns the elements of Enum class?

a) getEnums()
b) getEnumConstants()
c) getEnumList()
d) getEnum()

Ans: B) getEnumConstants()

13.) Why is it generally recommended to prefer interfaces over abstract classes when designing an API in Java?

A. Interfaces allow for better performance because they support default methods.

B. Abstract classes cannot contain concrete methods, making them less useful.

C. A class can implement multiple interfaces but can only extend one class, providing greater flexibility.

D. Interfaces are more secure because all their methods are implicitly public.

Ans: C) A class can implement multiple interfaces but can only extend one class, providing greater flexibility.

27. What will happen if a subclass does not override an abstract method of its superclass?
a) It will compile successfully
b) It will throw a runtime error
c) It will result in a compilation error
d) It will execute with a warning

Ans: C) It will result in a compilation error

44. A class can implement how many interfaces?
a) Only one
b) Zero
c) Any number of interfaces
d) A maximum of two
Ans: C) Any number of interfaces