Java Interview & Practice Questions (100)

Below are 100 multiple-choice questions covering all topics from Day1 to Day4 notes. Each question has 4 options. Answers and explanations are in Answers.md.

A) Base:5 Derived:5
B) Derived:5 Base:5
C) Compilation error
D) Runtime error

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

```java
class Base {
    public Base(int x) {
        System.out.println("Base:" + x);
    }
}
class Derived extends Base {
    public Derived(int x) {
        super(x);
        System.out.println("Derived:" + x);
    }
}
public class Main {
    public static void main(String[] args) {
        new Derived(5);
    }
}
```

A) Base:5 Derived:5
B) Derived:5 Base:5
C) Compilation error
D) Runtime error

2. Which statement about static initializer blocks is true?

A) They run every time an object is created
B) They run once when the class is loaded
C) They can only initialize instance variables
D) They cannot throw exceptions

3. What is the result of calling `super()` and `this()` in the same constructor?

A) Both are called

B) Only `super()` is called

C) Only `this()` is called

D) Compilation error

E) A static B static

F) B static A static

G) No output

H) Compilation error

4. What is the output?

```java
class A {
    static {
        System.out.println("A static");
    }
}
class B extends A {
    static {
        System.out.println("B static");
    }
}
public class Main {
    public static void main(String[] args) {
        new B();
    }
}
```

A) A static B static
B) B static A static
C) No output
D) Compilation error

5. Which is true about anonymous classes?

A) They can extend multiple classes

B) They can implement multiple interfaces

C) They are used for one-time use

D) They must be static

E) 42

F) 7

G) Compilation error

H) Runtime error

6. What is the output?

```java
class Api {
    Number value() { return 42; }
}
class Impl extends Api {
    Integer value() { return 7; }
}
public class Main {
    public static void main(String[] args) {
        Api api = new Impl();
        System.out.println(api.value());
    }
}
```

A) 42
B) 7
C) Compilation error
D) Runtime error

7. Which of the following is NOT a valid access modifier in Java?

A) public
B) private
C) protected
D) internal

8. What happens if a class does not override a default method from two interfaces with the same signature?

A) The method from the first interface is used

B) The method from the second interface is used

C) Compilation error

D) Runtime error

E) I

F) Super

G) Compilation error

H) Runtime error

9. What is the output?

```
interface I {
    default void ping() {
        System.out.println("I");
    }
}
class Super {
    public void ping() {
        System.out.println("Super");
    }
}
class Sub extends Super implements I {}
public class Main {
    public static void main(String[] args) {
        new Sub().ping();
    }
}
```

A) I
B) Super
C) Compilation error
D) Runtime error

10. Which of the following is true about method overloading?

A) Overloaded methods must have different names

B) Overloaded methods must have different parameter lists

C) Overloaded methods must have different return types only

D) Overloaded methods cannot be static

E) Base static Derived static

F) Derived static Base static

G) No output

H) Compilation error

11. What is the output?

```
class Base {
    static {
        System.out.println("Base static");
    }
}
class Derived extends Base {
    static {
        System.out.println("Derived static");
    }
}
public class Main {
    public static void main(String[] args) {
        new Derived();
    }
}
```

A) Base static Derived static
B) Derived static Base static
C) No output

D) Compilation error

12. Which of the following is true about instance initializer blocks?

A) They run once per class

B) They run every time an object is created

C) They can only initialize static variables

D) They must be static

E) A instance B instance

F) B instance A instance

G) No output

H) Compilation error

13. What is the output?

```
class A {
    {
        System.out.println("A instance");
    }
}
class B extends A {
    {
        System.out.println("B instance");
    }
}
public class Main {
    public static void main(String[] args) {
        new B();
    }
}
```

A) A instance B instance
B) B instance A instance
C) No output
D) Compilation error

14. Which is true about constructor chaining?

A) Only `super()` can be used

B) Only `this()` can be used

C) Both `super()` and `this()` can be used, but only one as the first statement

D) Both can be used anywhere in the constructor

E) Base Derived

F) Derived Base

G) No output

H) Compilation error

15. What is the output?

```
class Base {
    public Base() {
        System.out.println("Base");
    }
}
class Derived extends Base {
    public Derived() {
        System.out.println("Derived");
    }
}
public class Main {
    public static void main(String[] args) {
        new Derived();
    }
}
```

A) Base Derived
B) Derived Base
C) No output
D) Compilation error

16. Which of the following is true about static variables?

A) They are initialized every time an object is created

B) They are shared among all instances of a class

C) They cannot be accessed in static blocks

D) They must be final

E) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5

F) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5

G) Static block Instance block Default constructor

H) Compilation error

17. What is the output?

```
class Example {
    static {
        System.out.println("Static block");
    }
    {
        System.out.println("Instance block");
    }
    public Example() {
        System.out.println("Default constructor");
    }
    public Example(int x) {
        this();
        System.out.println("Constructor with x: " + x);
    }
}
public class Main {
    public static void main(String[] args) {
        Example e1 = new Example();
        Example e2 = new Example(5);
    }
}
```

A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5
B) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5
C) Static block Instance block Default constructor
D) Compilation error

18. Which of the following is NOT a valid use case for static blocks?

A) Loading drivers

B) Initializing static resources

C) Initializing instance variables

D) Setting up configuration

E) A static B static A instance A constructor B instance B constructor

F) B static A static B instance B constructor A instance A constructor

G) No output

H) Compilation error

19. What is the output?

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

A) A static B static A instance A constructor B instance B constructor
B) B static A static B instance B constructor A instance A constructor
C) No output
D) Compilation error

20. Which of the following is true about anonymous classes?

A) They can be used to override methods of a class or interface

B) They must be declared as static

C) They can have constructors

D) They cannot access final variables

E) Base:5 Derived:5,10

F) Base:5 Derived:5 Derived:5,10

G) Derived:5,10 Base:5

H) Compilation error

21. What is the output?

```
class Base {
    public Base(int x) {
        System.out.println("Base:" + x);
    }
}
```

```
class Derived extends Base {
    public Derived(int x, int y) {
        this(x);
        System.out.println("Derived:" + x + "," + y);
    }
    public Derived(int x) {
        super(x);
        System.out.println("Derived:" + x);
    }
}
public class Main {
    public static void main(String[] args) {
        new Derived(5, 10);
    }
}
```

A) Base:5 Derived:5,10
B) Base:5 Derived:5 Derived:5,10
C) Derived:5,10 Base:5
D) Compilation error

22. Which of the following is true about static blocks?

A) They can access static variables

B) They can access instance variables

C) They run after constructors

D) They cannot throw exceptions

E) Static block: a=10, b=0 Static block 2: b=20 10 20

F) 10 20 Static block: a=10, b=0 Static block 2: b=20

G) Static block: a=10, b=0 10 20 Static block 2: b=20

H) Compilation error

23. What is the output?

```
class A {
    static int a = 10;
    static int b;
    static {
        System.out.println("Static block: a=" + a + ", b=" + b);
        b = 20;
    }
    static {
        System.out.println("Static block 2: b=" + b);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        System.out.println(A.a);
        System.out.println(A.b);
    }
}
```

A) Static block: a=10, b=0 Static block 2: b=20 10 20
B) 10 20 Static block: a=10, b=0 Static block 2: b=20
C) Static block: a=10, b=0 10 20 Static block 2: b=20
D) Compilation error

24. Which of the following is true about final static variables?

A) They must be initialized at declaration

B) They can be initialized in static blocks

C) They cannot be initialized

D) They can only be initialized in constructors

E) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5

F) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5

G) Static block Instance block Default constructor

H) Compilation error

25. What is the output?

```
class Example {
    static {
        System.out.println("Static block");
    }
    {
        System.out.println("Instance block");
    }
    public Example() {
        System.out.println("Default constructor");
    }
    public Example(int x) {
        this();
        System.out.println("Constructor with x: " + x);
    }
}
public class Main {
    public static void main(String[] args) {
        Example e1 = new Example();
        Example e2 = new Example(5);
    }
}
```

A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5
B) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5
C) Static block Instance block Default constructor
D) Compilation error

26. Which of the following is true about static blocks?

A) They run only once per classloader

B) They run every time an object is created

C) They can only initialize static variables

D) They must be final

E) A static B static A instance A constructor B instance B constructor

F) B static A static B instance B constructor A instance A constructor

G) No output

H) Compilation error

27. What is the output?

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

A) A static B static A instance A constructor B instance B constructor
B) B static A static B instance B constructor A instance A constructor
C) No output
D) Compilation error

28. Which of the following is true about static variables?

A) They are initialized in the order they appear

B) They are initialized after constructors

C) They cannot be accessed in static blocks

D) They must be final

E) Static block: a=10, b=0 Static block 2: b=20 10 20

F) 10 20 Static block: a=10, b=0 Static block 2: b=20

G) Static block: a=10, b=0 10 20 Static block 2: b=20

H) Compilation error

29. What is the output?

```
class Base {
    static int a = 10;
    static int b;
    static {
        System.out.println("Static block: a=" + a + ", b=" + b);
        b = 20;
    }
    static {
        System.out.println("Static block 2: b=" + b);
    }
}
public class Main {
    public static void main(String[] args) {
        System.out.println(Base.a);
        System.out.println(Base.b);
    }
}
```

A) Static block: a=10, b=0 Static block 2: b=20 10 20
B) 10 20 Static block: a=10, b=0 Static block 2: b=20

C) Static block: a=10, b=0 10 20 Static block 2: b=20
D) Compilation error

30. Which of the following is true about static blocks?

A) They can access static variables

B) They can access instance variables

C) They run after constructors

D) They cannot throw exceptions

E) A static B static A instance A constructor B instance B constructor

F) B static A static B instance B constructor A instance A constructor

G) No output

H) Compilation error

31. What is the output?

```java
class A {
    static {
        System.out.println("A static");
    }
    {
        System.out.println("A instance");
    }
    public A() {
        System.out.println("A constructor");
    }
}
class B extends A {
    static {
        System.out.println("B static");
    }
    {
        System.out.println("B instance");
    }
    public B() {
        System.out.println("B constructor");
    }
}
public class Main {
    public static void main(String[] args) {
        new B();
    }
}
```

A) A static B static A instance A constructor B instance B constructor

B) B static A static B instance B constructor A instance A constructor
C) No output
D) Compilation error

32. Which of the following is true about static initializers?

A) They can access static variables

B) They can access instance variables

C) They run after constructors

D) They cannot throw exceptions

E) Static block: a=10, b=0 Static block 2: b=20 10 20

F) 10 20 Static block: a=10, b=0 Static block 2: b=20

G) Static block: a=10, b=0 10 20 Static block 2: b=20

H) Compilation error

33. What is the output?

```java
class Base {
    static int a = 10;
    static int b;
    static {
        System.out.println("Static block: a=" + a + ", b=" + b);
        b = 20;
    }
    static {
        System.out.println("Static block 2: b=" + b);
    }
}
public class Main {
    public static void main(String[] args) {
        System.out.println(Base.a);
        System.out.println(Base.b);
    }
}
```

A) Static block: a=10, b=0 Static block 2: b=20 10 20
B) 10 20 Static block: a=10, b=0 Static block 2: b=20
C) Static block: a=10, b=0 10 20 Static block 2: b=20
D) Compilation error

34. Which of the following is true about final static variables?

A) They must be initialized at declaration

B) They can be initialized in static blocks

C) They cannot be initialized

D) They can only be initialized in constructors

E) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5

F) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5

G) Static block Instance block Default constructor

H) Compilation error

35. What is the output?

```java
class Example {
    static {
        System.out.println("Static block");
    }
    {
        System.out.println("Instance block");
    }
    public Example() {
        System.out.println("Default constructor");
    }
    public Example(int x) {
        this();
        System.out.println("Constructor with x: " + x);
    }
}
public class Main {
    public static void main(String[] args) {
        Example e1 = new Example();
        Example e2 = new Example(5);
    }
}
```

A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5
B) Instance block Default constructor Static block Instance block Default constructor Constructor with x: 5
C) Static block Instance block Default constructor
D) Compilation error

36. Which of the following is true about static blocks?

A) They run only once per classloader

B) They run every time an object is created

C) They can only initialize static variables

D) They must be final

E) A static B static A instance A constructor B instance B constructor

F) B static A static B instance B constructor A instance A constructor

G) No output

H) Compilation error

37. What is the output?

```java
class A {
    static {
        System.out.println("A static");
    }
    {
        System.out.println("A instance");
    }
    public A() {
        System.out.println("A constructor");
    }
}
class B extends A {
    static {
        System.out.println("B static");
    }
    {
        System.out.println("B instance");
    }
    public B() {
        System.out.println("B constructor");
    }
}
public class Main {
    public static void main(String[] args) {
        A obj = new B();
    }
}
```

A) A static B static A instance A constructor B instance B constructor
B) B static A static B instance B constructor A instance A constructor
C) No output
D) Compilation error

38. Which of the following is true about static variables?

A) They are initialized in the order they appear

B) They are initialized after constructors

C) They cannot be accessed in static blocks

D) They must be final

E) Static block: a=10, b=0 Static block 2: b=20 10 20

F) 10 20 Static block: a=10, b=0 Static block 2: b=20

G) Static block: a=10, b=0 10 20 Static block 2: b=20

H) Compilation error

39. What is the output?

```java
class Base {
    static int a = 10;
    static int b;
    static {
        System.out.println("Static block: a=" + a + ", b=" + b);
        b = 20;
    }
    static {
        System.out.println("Static block 2: b=" + b);
    }
}
public class Main {
    public static void main(String[] args) {
        System.out.println(Base.a);
        System.out.println(Base.b);
    }
}
```

A) Static block: a=10, b=0 Static block 2: b=20 10 20
B) 10 20 Static block: a=10, b=0 Static block 2: b=20
C) Static block: a=10, b=0 10 20 Static block 2: b=20
D) Compilation error

40. Which of the following is true about static blocks?

A) They can access static variables
B) They can access instance variables
C) They run after constructors
D) They cannot throw exceptions

41. What is the output?

```java
class A { public void f() { System.out.println("A.f"); } public void call() { f(); } }
class B extends A { public void f() { System.out.println("B.f"); } }
public class Main { public static void main(String[] args) { A x = new B(); x.call(); } }
```

A) A.f
B) B.f
C) Compilation error
D) Runtime error

42. Which of the following is true about method overriding?

A) The method in the subclass must have the same name and parameter list
B) The method in the subclass can have a different return type
C) The method in the subclass can have a different access modifier
D) The method in the subclass can be static

43. What is the output?

```
class A { public void f() { System.out.println("A.f"); } public void call() { f(); } }
class B extends A { public void f() { System.out.println("B.f"); } public void call() { f()
public class Main { public static void main(String[] args) { A x = new B(); x.call(); } }
```

A) A.f
B) B.f
C) Compilation error
D) Runtime error

44. Which of the following is true about method overloading?

A) Overloaded methods must have different parameter lists

B) Overloaded methods must have different return types only

C) Overloaded methods must have different names

D) Overloaded methods cannot be static

E) long long Integer long int...

F) Integer long long int... long

G) int... long Integer long long

H) Compilation error

45. What is the output?

```
public class Main {
    static void f(long x) { System.out.println("long"); }
    static void f(Integer x) { System.out.println("Integer"); }
    static void f(int... x) { System.out.println("int..."); }
    public static void main(String[] args) {
        f(1);
        f((short)1);
        f(Integer.valueOf(1));
        f((byte)1);
        f(new int[]{1});
    }
}
```

A) long long Integer long int...
B) Integer long long int... long
C) int... long Integer long long

D) Compilation error

46. Which of the following is true about overload resolution priority?

A) Widening > Boxing > Varargs

B) Boxing > Widening > Varargs

C) Varargs > Boxing > Widening

D) Boxing > Varargs > Widening

E) long

F) Integer

G) Compilation error

H) Runtime error

47. What is the output?

```
class Demo {
    void a(long i) { System.out.println("long"); }
    void a(Integer i) { System.out.println("Integer"); }
}
public class Main {
    public static void main(String[] args) {
        Demo demo = new Demo();
        demo.a(5);
    }
}
```

A) long
B) Integer
C) Compilation error
D) Runtime error

48. Which of the following is true about method references?

A) They can refer to overloaded methods

B) They cannot refer to static methods

C) They can only refer to constructors

D) They cannot be used with functional interfaces

E) S O

F) O S

G) S S

H) O O

49. What is the output?

```
import java.util.function.Function;
class S {
    String m(Object o) { return "O"; }
    String m(String s) { return "S"; }
}
public class Main {
    public static void main(String[] args) {
        Function<String, String> f = new S()::m;
        Function<Object, String> f2 = new S()::m;
        System.out.println(f.apply(null));
        System.out.println(f2.apply(null));
    }
}
```

A) S O
B) O S
C) S S
D) O O

50. Which of the following is true about interface default methods?

A) They can be overridden by classes

B) They cannot be overridden

C) They must be static

D) They cannot have a body

E) A B

F) B A

G) Compilation error

H) Runtime error

51. What is the output?

```
interface A {
    default void ping() { System.out.println("A"); }
}
interface B {
    default void ping() { System.out.println("B"); }
}
class C implements A, B {
    public void ping() {
        A.super.ping();
        B.super.ping();
    }
}
public class Main {
    public static void main(String[] args) {
        new C().ping();
    }
}
```

A) A B
B) B A
C) Compilation error
D) Runtime error

52. Which of the following is true about static methods in interfaces?

A) They can be called on the interface

B) They can be called on instances

C) They can be overridden

D) They must be private

E) X ID Util ID

F) Util ID X ID

G) Compilation error

H) Runtime error

53. What is the output?

```
interface Util {
    static String id() { return "Util ID"; }
}
class X implements Util {
    static String id() { return "X ID"; }
}
public class Main {
    public static void main(String[] args) {
        Util u = new X();
        System.out.println(X.id());
        System.out.println(Util.id());
    }
}
```

A) X ID Util ID
B) Util ID X ID
C) Compilation error
D) Runtime error

54. Which of the following is true about private methods in interfaces?

A) They can be called from default methods

B) They can be called from outside the interface

C) They must be static

D) They cannot have a body

E) hello world

F) Hello World

G) Compilation error

H) Runtime error

55. What is the output?

```
interface Parser {
    default String sanitize(String s) { return trimAndLower(s); }
    private String trimAndLower(String s) { return s.trim().toLowerCase(); }
}
class Impl implements Parser {
    public String demo(String s) { return sanitize(s); }
}
public class Main {
    public static void main(String[] args) {
        Impl impl = new Impl();
        System.out.println(impl.sanitize("  Hello World  "));
    }
}
```

A) hello world
B) Hello World
C) Compilation error
D) Runtime error

56. Which of the following is true about method overloading vs overriding?

A) Overloading is compile-time, overriding is runtime

B) Overloading is runtime, overriding is compile-time

C) Both are compile-time

D) Both are runtime

E) A B

F) B A

G) Compilation error

H) Runtime error

57. What is the output?

```
interface I {
    void m(Number n);
}
class A {
    public void m(Integer i) { System.out.println("A"); }
}
class B extends A implements I {
    public void m(Number n) { System.out.println("B"); }
}
public class Main {
    public static void main(String[] args) {
        new B().m(10);
        ((I) new B()).m(10);
    }
}
```

A) A B
B) B A
C) Compilation error
D) Runtime error

58. Which of the following is true about covariant return types?

A) The return type in the subclass can be a subclass of the return type in the superclass

B) The return type in the subclass must be the same as the superclass

C) The return type in the subclass can be unrelated

D) The return type in the subclass must be Object

E) Circle

F) Shape

G) Compilation error

H) Runtime error

59. What is the output?

```
class Shape {}
class Circle extends Shape {}
interface Factory {
    Shape create();
}
class CircleFactory implements Factory {
    public Circle create() { return new Circle(); }
}
public class Main {
    public static void main(String[] args) {
        Factory factory = new CircleFactory();
        Shape shape = factory.create();
        System.out.println(shape.getClass().getSimpleName());
    }
}
```

A) Circle
B) Shape
C) Compilation error
D) Runtime error

60. Which of the following is true about exception compatibility in overriding?

A) The overriding method cannot throw broader checked exceptions

B) The overriding method can throw any exception

C) The overriding method must throw the same exception

D) The overriding method cannot throw unchecked exceptions

E) C

F) Compilation error

G) IOException

H) SQLException

61. What is the output?

```
import java.io.IOException;
import java.sql.SQLException;
abstract class A {
    abstract void run() throws IOException;
}
interface I {
    void run() throws SQLException;
}
class C extends A implements I {
    public void run() { System.out.println("C"); }
}
public class Main {
    public static void main(String[] args) {
        new C().run();
    }
}
```

A) C
B) Compilation error
C) IOException
D) SQLException

62. Which of the following is true about sealed interfaces?

A) They restrict which classes can implement them

B) They can only be used with abstract classes

C) They must be final

D) They cannot have default methods

E) Persian

F) Dog

G) Cat

H) Compilation error

63. What is the output?

```
sealed interface Animal permits Dog, Cat {}
non-sealed interface Dog extends Animal {}
interface Cat extends Animal {}
class Persian implements Dog {}
public class Main {
    public static void main(String[] args) {
        Dog dog = new Persian();
        System.out.println(dog.getClass().getSimpleName());
    }
}
```

A) Persian
B) Dog
C) Cat
D) Compilation error

64. Which of the following is true about static methods in classes?

A) They can be overridden

B) They can be hidden

C) They must be final

D) They cannot be called from subclasses

E) Hello from P Hello from Q

F) Hello from Q Hello from P

G) Compilation error

H) Runtime error

65. What is the output?

```
class P {
    static void hello() { System.out.println("Hello from P"); }
}
class Q {
    static void hello() { System.out.println("Hello from Q"); }
}
public class Main {
    public static void main(String[] args) {
        P.hello();
        Q.hello();
    }
}
```

A) Hello from P Hello from Q
B) Hello from Q Hello from P
C) Compilation error
D) Runtime error

66. Which of the following is true about generics and type erasure?

A) Type information is erased at runtime

B) Type information is preserved at runtime

C) Generics can be used with primitive types

D) Generics cannot be used with interfaces

E) Hello Hello

F) Compilation error

G) Hello null

H) null Hello

67. What is the output?

```
interface Box<T> {
    T get();
}
class StringBox implements Box<String> {
    public String get() { return "Hello"; }
}
public class Main {
    public static void main(String[] args) {
        Box b = new StringBox();
        Box<String> b2 = new StringBox();
        System.out.println(b.get());
        System.out.println(b2.get());
    }
}
```

A) Hello Hello

B) Compilation error
C) Hello null
D) null Hello

68. Which of the following is true about overloading: most specific method chosen?

A) The most specific method is chosen at compile time

B) The most specific method is chosen at runtime

C) The least specific method is chosen at compile time

D) The least specific method is chosen at runtime

E) String

F) Object

G) Compilation error

H) Runtime error

69. What is the output?

```
class Overload {
    void test(String s) { System.out.println("String"); }
    void test(Object o) { System.out.println("Object"); }
}
public class Main {
    public static void main(String[] args) {
        new Overload().test(null);
    }
}
```

A) String
B) Object
C) Compilation error
D) Runtime error

70. Which of the following is true about overloading vs overriding: parameter types?

A) Overloading is based on parameter types

B) Overriding is based on parameter types

C) Both are based on parameter types

D) Neither is based on parameter types

E) Parent

F) Child

G) Compilation error

H) Runtime error

71. What is the output?

```
class Parent {
    void show(Number n) { System.out.println("Parent"); }
}
class Child extends Parent {
    void show(Integer n) { System.out.println("Child"); }
}
public class Main {
    public static void main(String[] args) {
        Parent p = new Child();
        p.show(10);
    }
}
```

A) Parent
B) Child
C) Compilation error
D) Runtime error

72. Which of the following is true about final methods?

A) They cannot be overridden

B) They can be overridden

C) They must be static

D) They must be private

E) Parent

F) Compilation error

G) Child

H) Runtime error

73. What is the output?

```
class Parent {
    public final void ping() { System.out.println("Parent"); }
}
class Child extends Parent {
    // public void ping() { ... } // Error
}
public class Main {
    public static void main(String[] args) {
        new Child().ping();
    }
}
```

A) Parent
B) Compilation error
C) Child
D) Runtime error

74. Which of the following is true about class vs interface default method precedence?

A) Class methods take precedence over interface default methods

B) Interface default methods take precedence over class methods

C) Both are equal

D) Neither is used

E) Super

F) I

G) Compilation error

H) Runtime error

75. What is the output?

```
interface I {
    default void ping() { System.out.println("I"); }
}
class Super {
    public void ping() { System.out.println("Super"); }
}
class Sub extends Super implements I {}
public class Main {
    public static void main(String[] args) {
        new Sub().ping();
    }
}
```

A) Super
B) I
C) Compilation error
D) Runtime error

76. Which of the following is true about overloading: widening vs boxing?

A) Widening is preferred over boxing

B) Boxing is preferred over widening

C) Both are equal

D) Neither is used

E) Long

F) double

G) Compilation error

H) Runtime error

77. What is the output?

```
class Demo {
    void a(Long i) { System.out.println("Long"); }
    void a(double i) { System.out.println("double"); }
}
public class Main {
    public static void main(String[] args) {
        Demo demo = new Demo();
        demo.a(5L);
    }
}
```

A) Long
B) double
C) Compilation error
D) Runtime error

78. Which of the following is true about interface default method ambiguity?

A) The class must override the method to resolve ambiguity

B) The first interface's method is used

C) The second interface's method is used

D) Compilation error

E) T1

F) Compilation error

G) T2

H) Runtime error

79. What is the output?

```
interface T1 {
    default void m() { System.out.println("T1"); }
}
interface T2 {
    void m();
}
class C implements T1, T2 {} // Error: class C inherits unrelated defaults for m() from typ
// To fix:
class C2 implements T1, T2 {
    public void m() { T1.super.m(); }
```

```java
}
public class Main {
    public static void main(String[] args) {
        new C2().m();
    }
}
```

A) T1
B) Compilation error
C) T2
D) Runtime error

80. Which of the following is true about abstract class vs interface default method?

A) Abstract class method takes precedence

B) Interface default method takes precedence

C) Both are equal

D) Neither is used

E) I

F) Compilation error

G) D

H) Runtime error

81. What is the output?

```java
interface I {
    default void m() { System.out.println("I"); }
}
abstract class A {
    abstract void m();
}
class D extends A implements I {} // Error: D is not abstract and does not override abstrac
// To fix:
class D2 extends A implements I {
    public void m() { I.super.m(); }
}
public class Main {
    public static void main(String[] args) {
        new D2().m();
    }
}
```

A) I
B) Compilation error

C) D
D) Runtime error

82. Which of the following is true about overloading with null: ambiguity?

A) Reference to method is ambiguous

B) The first method is used

C) The second method is used

D) Compilation error

E) Compilation error

F) Integer

G) Long

H) Runtime error

83. What is the output?

```java
public class Main {
    void g(Integer x) { System.out.println("Integer"); }
    void g(Long x) { System.out.println("Long"); }
    public static void main(String[] args) {
        // new Main().g(null); // Error: reference to g is ambiguous
    }
}
```

public class Main { void g(Integer x) { System.out.println("Integer"); } void g(Long x) { System.out.println("Long"); } public static void main(String[] args) { /* new Main().g(null); */ } }

~~~
A) Compilation error
B) Integer
C) Long
D) Runtime error

84. Which of the following is true about interface fields and hiding?
A) Interface fields are static and final
B) Interface fields can be overridden
C) Interface fields are instance variables
D) Interface fields can be private

85. What is the output?
```java
interface K {
    int X = 1;
}
class Imp1 implements K {
~~~

```
    public int X = 2;
}
public class Main {
    public static void main(String[] args) {
        K k = new Imp1();
        Imp1 imp1 = new Imp1();
        System.out.println(k.X + " " + imp1.X);
    }
}
```

A) 1 2
B) 2 1
C) Compilation error
D) Runtime error

86. Which of the following is true about overloading with varargs and widening?

A) Widening is preferred over varargs
B) Varargs is preferred over widening
C) Both are equal
D) Neither is used

87. What is the output?

```
public class Main {
    static void f(long s) {
        System.out.println("long");
    }
    static void f(int... s) {
        System.out.println("varargs");
    }
    static void f(Integer... s) {
        System.out.println("varargs Integer");
    }
    public static void main(String[] args) {
        byte n = 5;
        f(n);
        f((short)n);
    }
}
```

A) long long
B) varargs varargs Integer
C) varargs Integer varargs
D) Compilation error

88. Which of the following is true about method references and overloading?

A) Method references can refer to overloaded methods

B) Method references cannot refer to overloaded methods
C) Method references can only refer to static methods
D) Method references can only refer to constructors

89. What is the output?

```
import java.util.function.Function;
class S {
    String m(Object o) {
        return "O";
    }
    String m(String s) {
        return "S";
    }
}
public class Main {
    public static void main(String[] args) {
        Function<String, String> f = new S()::m;
        Function<Object, String> f2 = new S()::m;
        System.out.println(f.apply(null));
        System.out.println(f2.apply(null));
    }
}
```

A) S O
B) O S
C) S S
D) O O

90. Which of the following is true about sealed and non-sealed interfaces: ambiguity?

A) The class must override the method to resolve ambiguity
B) The first interface's method is used
C) The second interface's method is used
D) Compilation error

91. What is the output?

```
sealed interface A permits B, C {}
non-sealed interface B extends A {
    default void hello() {
        System.out.println("Hello from B");
    }
}
non-sealed interface C extends A {
    default void hello() {
        System.out.println("Hello from C");
    }
```

```java
}
class D implements B, C {
    public void hello() {
        B.super.hello();
    }
}
public class Main {
    public static void main(String[] args) {
        new D().hello();
    }
}
```

A) Hello from B
B) Hello from C
C) Compilation error
D) Runtime error

92. Which of the following is true about overriding and narrowing visibility?

A) You cannot reduce visibility when overriding
B) You can reduce visibility when overriding
C) You must reduce visibility when overriding
D) Visibility does not matter

93. What is the output?

```java
interface Box<T> {
    T get();
}
class Sbox implements Box<String> {
    public String get() {
        return "X";
    }
}
public class Main {
    public static void main(String[] args) {
        Box<String> b = new Sbox();
        System.out.println(b.get());
    }
}
```

A) X
B) Compilation error
C) null
D) Runtime error

94. Which of the following is true about static initialization order?

A) Static blocks run in the order they appear
B) Static blocks run after constructors

C) Static blocks run before static variables
D) Static blocks run only in subclasses

95. What is the output?

```java
class J {
    static {
        System.out.println("Init");
    }
    static final int A = 1;
    static final Integer B = 2;
}
public class Main {
    public static void main(String[] args) {
        System.out.println(J.A);
        System.out.println(J.B);
    }
}
```

A) Init 1 2
B) 1 2 Init
C) 1 2
D) Compilation error

96. Which of the following is true about exception overriding: subclass exception?

A) The overriding method can throw a subclass of the exception
B) The overriding method must throw the same exception
C) The overriding method cannot throw any exception
D) The overriding method must throw a superclass exception

97. What is the output?

```java
import java.io.FileNotFoundException;
import java.io.IOException;
class A {
    void read() throws IOException {
        System.out.println("A");
    }
}
class B extends A {
    void read() throws FileNotFoundException {
        System.out.println("B");
    }
}
public class Main {
    public static void main(String[] args) throws IOException {
        A a = new B();
        a.read();
    }
}
```

A) B
B) A
C) Compilation error
D) Runtime error

98. Which of the following is true about multi-catch variable?

A) It is final and cannot be assigned
B) It can be assigned
C) It must be null
D) It must be static

99. What is the output?

```
class R implements AutoCloseable {
    public void close() {
        throw new RuntimeException("close");
    }
}
public class Main {
    public static void main(String[] args) {
        try (R r = new R()) {
            throw new RuntimeException("body");
        } catch (RuntimeException e) {
            System.out.println(e.getMessage());
            for (Throwable t : e.getSuppressed()) {
                System.out.println("Suppressed: " + t.getMessage());
            }
        }
    }
}
```

A) body Suppressed: close
B) close Suppressed: body
C) body close
D) Compilation error

100. Which of the following is true about static initializer exception?

A) If static initialization fails, main is never reached
B) If static initialization fails, main is always reached
C) Static initialization cannot fail
D) Static initialization runs after main

Answers and Explanations for Java Practice Questions

---

1. A) Base:5 Derived:5 **Explanation:** The constructor of Base is called first (super(x)), then Derived's constructor prints its message.
2. B) They run once when the class is loaded **Explanation:** Static initializer blocks execute only once when the class is loaded into memory.
3. D) Compilation error **Explanation:** You cannot call both `super()` and `this()` in the same constructor; only one is allowed and it must be the first statement.
4. A) A static B static **Explanation:** Static blocks execute in the order of class hierarchy when the class is loaded.
5. C) They are used for one-time use **Explanation:** Anonymous classes are typically used for one-time, short-lived implementations.
6. B) 7 **Explanation:** The overridden method in Impl returns 7. Covariant return types allow Integer to override Number.
7. D) internal **Explanation:** 'internal' is not a valid Java access modifier.
8. C) Compilation error **Explanation:** If two interfaces provide the same default method, the class must override it to resolve ambiguity.
9. B) Super **Explanation:** Class methods take precedence over interface default methods.
10. B) Overloaded methods must have different parameter lists **Explanation:** Overloading requires different parameter lists, not just different return types.
11. A) Base static Derived static **Explanation:** Static blocks in the base class run before those in the derived class when the derived class is loaded.
12. B) They run every time an object is created **Explanation:** Instance initializer blocks execute before the constructor, every time an object is created.
13. A) A instance B instance **Explanation:** Instance initializers run in order of inheritance, so A's runs before B's.
14. C) Both `super()` and `this()` can be used, but only one as the first statement **Explanation:** Only one can be the first statement in a constructor.
15. A) Base Derived **Explanation:** Base constructor runs first, then Derived's.
16. B) They are shared among all instances of a class **Explanation:** Static variables are class-level, not instance-level.
17. A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5 **Explanation:** Static block runs once, then each object triggers the instance block and constructor.
18. C) Initializing instance variables **Explanation:** Static blocks are for

static initialization, not instance variables.

19. A) A static B static A instance A constructor B instance B constructor **Explanation:** Static blocks run in order of inheritance, then instance blocks and constructors.
20. A) They can be used to override methods of a class or interface **Explanation:** Anonymous classes are often used to override methods for one-time use.
21. B) Base:5 Derived:5 Derived:5,10 **Explanation:** Chained constructors: Base(5), Derived(5), then Derived(5,10).
22. A) They can access static variables **Explanation:** Static blocks can access static variables.
23. A) Static block: a=10, b=0 Static block 2: b=20 10 20 **Explanation:** Static variables and blocks run in order, then main prints values.
24. B) They can be initialized in static blocks **Explanation:** Final static variables can be initialized in static blocks.
25. A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5 **Explanation:** Static block runs once, then each object triggers the instance block and constructor.
26. A) They run only once per classloader **Explanation:** Static blocks run once per classloader, not per object.
27. A) A static B static A instance A constructor B instance B constructor **Explanation:** Static blocks run in order, then instance blocks and constructors.
28. A) They are initialized in the order they appear **Explanation:** Static variables are initialized in the order they appear in the class.
29. A) Static block: a=10, b=0 Static block 2: b=20 10 20 **Explanation:** Static variables and blocks run in order, then main prints values.
30. A) They can access static variables **Explanation:** Static blocks can access static variables.
31. A) A static B static A instance A constructor B instance B constructor **Explanation:** Static and instance blocks run in order of inheritance.
32. A) They can access static variables **Explanation:** Static initializers can access static variables.
33. A) Static block: a=10, b=0 Static block 2: b=20 10 20 **Explanation:** Static variables and blocks run in order, then main prints values.
34. B) They can be initialized in static blocks **Explanation:** Final static variables can be initialized in static blocks.
35. A) Static block Instance block Default constructor Instance block Default constructor Constructor with x: 5 **Explanation:** Static block runs once, then each object triggers the instance block and constructor.
36. A) They run only once per classloader **Explanation:** Static blocks run once per classloader, not per object.
37. A) A static B static A instance A constructor B instance B constructor **Explanation:** Static blocks run in order, then instance blocks and

constructors.

38. A) They are initialized in the order they appear **Explanation:** Static variables are initialized in the order they appear in the class.
39. A) Static block: a=10, b=0 Static block 2: b=20 10 20 **Explanation:** Static variables and blocks run in order, then main prints values.
40. A) They can access static variables **Explanation:** Static blocks can access static variables.
41. B) B.f **Explanation:** Method call is resolved at runtime to the subclass implementation.
42. A) The method in the subclass must have the same name and parameter list **Explanation:** Overriding requires the same method signature.
43. B) B.f **Explanation:** Overridden call() in B calls B's f().
44. A) Overloaded methods must have different parameter lists **Explanation:** Overloading requires different parameter lists.
45. A) long long Integer long int... **Explanation:** Overload resolution: widening, boxing, varargs.
46. A) Widening > Boxing > Varargs **Explanation:** This is the overload resolution priority in Java.
47. A) long **Explanation:** int argument is widened to long.
48. A) They can refer to overloaded methods **Explanation:** Method references can refer to overloaded methods.
49. A) S O **Explanation:** f.apply(null) matches String, f2.apply(null) matches Object.
50. A) They can be overridden by classes **Explanation:** Classes can override default methods from interfaces.
51. A) A B **Explanation:** C's ping() calls both A's and B's default methods.
52. A) They can be called on the interface **Explanation:** Static interface methods are called on the interface, not instances.
53. A) X ID Util ID **Explanation:** Static methods are called on the class/interface, not the instance.
54. A) They can be called from default methods **Explanation:** Private interface methods are for use within the interface.
55. A) hello world **Explanation:** sanitize() uses the private method to trim and lowercase.
56. A) Overloading is compile-time, overriding is runtime **Explanation:** Overloading is resolved at compile time, overriding at runtime.
57. A) A B **Explanation:** m(Integer) is called for new B().m(10), m(Number) for ((I) new B()).m(10).
58. A) The return type in the subclass can be a subclass of the return type in the superclass **Explanation:** This is called covariant return type.
59. A) Circle **Explanation:** The factory returns a Circle, and getSimpleName() prints "Circle".
60. A) The overriding method cannot throw broader checked exceptions **Explanation:** The overriding method can only throw the same or narrower checked exceptions.

61. A) C **Explanation:** The implementation in C does not throw checked exceptions.
62. A) They restrict which classes can implement them **Explanation:** Sealed interfaces restrict which classes can implement them.
63. A) Persian **Explanation:** The runtime type is Persian.
64. B) They can be hidden **Explanation:** Static methods can be hidden, not overridden.
65. A) Hello from P Hello from Q **Explanation:** Both static methods are called directly.
66. A) Type information is erased at runtime **Explanation:** Java generics use type erasure.
67. A) Hello Hello **Explanation:** Both get() calls return "Hello".
68. A) The most specific method is chosen at compile time **Explanation:** Overloading resolution is compile-time.
69. A) String **Explanation:** The String version is more specific than Object.
70. A) Overloading is based on parameter types **Explanation:** Overloading is based on parameter types, overriding is not.
71. A) Parent **Explanation:** p.show(10) calls Parent's show(Number), not Child's show(Integer).
72. A) They cannot be overridden **Explanation:** final methods cannot be overridden.
73. A) Parent **Explanation:** Child inherits the final method from Parent.
74. A) Class methods take precedence over interface default methods **Explanation:** Class methods always win over interface default methods.
75. A) Super **Explanation:** Class method takes precedence over interface default method.
76. A) Widening is preferred over boxing **Explanation:** Java prefers widening over boxing in overload resolution.
77. A) Long **Explanation:** 5L matches the Long parameter.
78. A) The class must override the method to resolve ambiguity **Explanation:** Ambiguity must be resolved by the implementing class.
79. A) T1 **Explanation:** C2 resolves ambiguity by calling T1's default method.
80. A) Abstract class method takes precedence **Explanation:** Abstract class method must be implemented, even if interface provides a default.
81. A) I **Explanation:** D2 implements m() and calls I's default method.
82. A) Reference to method is ambiguous **Explanation:** Overloading with null is ambiguous if multiple methods match.
83. A) Compilation error **Explanation:** The call is ambiguous; both g(Integer) and g(Long) match null.
84. A) Interface fields are static and final **Explanation:** All interface fields are implicitly public, static, and final.
85. A) 1 2 **Explanation:** k.X is the interface field, imp1.X is the class field.

86. A) Widening is preferred over varargs **Explanation:** Java prefers widening over varargs in overload resolution.
87. A) long long **Explanation:** Both calls use widening to long.
88. A) Method references can refer to overloaded methods **Explanation:** Method references can refer to overloaded methods.
89. A) S O **Explanation:** f.apply(null) matches String, f2.apply(null) matches Object.
90. A) The class must override the method to resolve ambiguity **Explanation:** Ambiguity must be resolved by the implementing class.
91. A) Hello from B **Explanation:** D overrides hello() and calls B's default method.
92. A) You cannot reduce visibility when overriding **Explanation:** Overriding methods cannot reduce visibility.
93. A) X **Explanation:** get() returns "X".
94. A) Static blocks run in the order they appear **Explanation:** Static blocks run in the order they appear in the class.
95. A) Init 1 2 **Explanation:** Static block runs first, then static fields are printed.
96. A) The overriding method can throw a subclass of the exception **Explanation:** Overriding methods can throw narrower (subclass) exceptions.
97. A) B **Explanation:** B's read() is called, which prints "B".
98. A) It is final and cannot be assigned **Explanation:** Multi-catch variables are implicitly final.
99. A) body Suppressed: close **Explanation:** The exception from close() is suppressed and attached to the main exception.
100. A) If static initialization fails, main is never reached **Explanation:** If a static initializer throws an exception, the class cannot be used and main is never called.