Java Interfaces — Interview Cheat Sheet

1. What is an Interface?

An interface in Java is a contract that defines what a class should do, without saying how. It contains method declaratio

Example:

```
interface Animal {
    void sound();
}
```

2. Why Do We Need Interfaces?

Key reasons:
- Achieve abstraction
- Enable multiple inheritance
- Promote loose coupling
- Encourage plug-and-play design

Example:

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

3. Method Types in Interface (Java 1 to 9+)

- Abstract (Java 1): Must be implemented
- Default (Java 8): Has a body, can be overridden
- Static (Java 8): Called on interface, not instance
- Private (Java 9): For reuse in default methods
- Private Static (Java 9): Reuse logic in static methods

4. Fields in Interfaces

All interface fields are public static final.
Why?
- public: Accessible everywhere
- static: Belongs to the interface
- final: Value cannot change

Example:

```
interface Config {
    int TIMEOUT = 5000;
}
```

5. Implementing Interfaces

```
interface Flyable {
    void fly();
}

class Bird implements Flyable {
    public void fly() {
        System.out.println("Flying...");
    }
}
```

6. Nested Interfaces

```
class Outer {
    interface Inner {
        void display();
    }
}

class Impl implements Outer.Inner {
    public void display() {
        System.out.println("Hello");
    }
}
```

7. Java 8 Interface Features

- Default Methods: Methods with body
- Static Methods: Utility methods
- Functional Interfaces: One abstract method
- Lambda Expressions: Short form for implementing functional interfaces

8. Lambda Expressions — Breakdown

Syntax: (parameters) -> { method body }

Example:

```
@FunctionalInterface
interface Greeting {
    void sayHello();
}

Greeting g = () -> System.out.println("Hello!");
g.sayHello();
```

Parts:
- () → Parameter list
- -> → Lambda operator
- {} → Method body

Advantages:
- Cleaner code
- Used in streams
- Less boilerplate

9. Java 9 Interface Features

- Private Methods: Reuse logic inside default methods
- Private Static Methods: Reuse logic inside static methods

10. Interface vs Abstract Class

Interface:
- Implements (multiple)
- Only public static final fields
- No constructor

Abstract Class:
- Extends (single)
- Any type of field
- Can have constructors

Final Summary

Interfaces define a contract for behavior without enforcing implementation. They enable abstraction, loose coupling, an