OpenAI

Java Keywords

Here's a beginner-friendly guide to **Java Keywords** from scratch with clear **code explanations**. Java keywords are **reserved words** that have a predefined meaning in the language. They **cannot be used as identifiers (like variable names, class names, etc.)**.

🗂️ List of Important Java Keywords (with code examples)

1. `class`

Used to declare a class.

```java
class Animal {
    // properties and methods
}
```

---

2. `public`, `private`, `protected`

Access modifiers that define the visibility of classes, methods, and variables.

```java
public class Car {
    private String model = "Honda";
    protected int speed = 100;
}
```

---

3. `static`

Used to define a class-level variable or method.

```java
public class MathUtil {
    static int square(int x) {
        return x * x;
    }

    public static void main(String[] args) {
        System.out.println(MathUtil.square(4));  // Output: 16
    }
}
```

---

4. `void`

Specifies that a method doesn't return any value.

```java
public class Printer {
    void printHello() {
        System.out.println("Hello");
    }
}
```

---

5. `int`, `double`, `char`, `boolean`

Primitive data types.

```java
int age = 25;
double price = 99.99;
char grade = 'A';
boolean isJavaFun = true;
```

---

6. `if`, `else`, `else if`

Conditional statements.

```java
int score = 85;
if (score >= 90) {
    System.out.println("A grade");
} else if (score >= 80) {
    System.out.println("B grade");
} else {
    System.out.println("C grade");
}
```

---

7. `for`, `while`, `do`

Looping constructs.

```java
for (int i = 0; i < 3; i++) {
    System.out.println("Count: " + i);
}

int j = 0;
while (j < 3) {
    System.out.println("While Count: " + j);
    j++;
}

int k = 0;
do {
    System.out.println("Do While Count: " + k);
    k++;
} while (k < 3);
```

---

8. `new`

Used to create new objects.

```java
String name = new String("Koti");
```

---

9. `return`

Used to return a value from a method.

```java
int add(int a, int b) {
    return a + b;
}
```

---

10. `this`

Refers to the current object.

```java
class Person {
    String name;
    Person(String name) {
        this.name = name;  // distinguish instance variable from parameter
    }
}
```

---

11. `final`

Used to declare constants or prevent method overriding/class inheritance.

```java
final int MAX_AGE = 100;

final class Vehicle {
    // Cannot be subclassed
}
```

---

12. `abstract`

Used to define abstract classes or methods.

```java
abstract class Animal {
    abstract void sound();  // no body
}
```

```
class Dog extends Animal {
    void sound() {
        System.out.println("Bark");
    }
}
```

---

13. `interface`, `implements`

Used to define and implement interfaces.

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

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

---

14. `extends`

Used for class inheritance.

```java
class Animal {
    void eat() {
        System.out.println("Eating...");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Barking...");
    }
}
```

---

15. `super`

Refers to the parent class.

```java
class Animal {
    void sound() {
        System.out.println("Animal sound");
    }
}

class Dog extends Animal {
    void sound() {
        super.sound();  // call parent method
        System.out.println("Dog barks");
    }
}
```

---

16. `try`, `catch`, `finally`, `throw`, `throws`

Exception handling keywords.

```java
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int x = 5 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");
        } finally {
            System.out.println("Finally block executed");
        }
    }
}
```

---

17. `break`, `continue`

Control flow in loops.

```java
for (int i = 0; i < 5; i++) {
    if (i == 3) break;
    System.out.println(i);
}

for (int i = 0; i < 5; i++) {
    if (i == 3) continue;
    System.out.println(i);
}
```

---

18. `instanceof`

Checks if an object is of a specific type.

```java
String s = "Hello";
System.out.println(s instanceof String);  // true
```

---

19. `enum`

Defines a set of named constants.

```java
enum Day { MON, TUE, WED }

class TestEnum {
    public static void main(String[] args) {
        Day d = Day.MON;
        System.out.println(d);  // MON
    }
}
```

20. `import`, `package`

Used to organize classes and interfaces.

```java
package com.example;

import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
    }
}
```

✅ Summary Table

Keyword	Purpose
`class`	Declares a class
`public`	Access modifier
`static`	Shared/class-level method/variable
`void`	No return value
`int`, etc.	Primitive types
`if`, **`else`**	Conditional flow
`for`, **`while`**	Loops
`new`	Creates an object
`return`	Exits from method with a value
`this`	Refers to current object
`final`	Constant or no override
`abstract`	Incomplete method/class
`extends`	Inherits class
`implements`	Implements interface
`try-catch`	Exception handling

Would you like a **Java practice quiz** on keywords or **assignments** with solutions?

Great! Let's now **focus in detail** on the Java keywords you mentioned:

✅ 1. `static` – Class-level Member

- `static` makes a **method or variable belong to the class**, not the instance.
- You can access it **without creating an object**.

🔹 Example:

```java
public class Demo {
    static int count = 0;

    static void showCount() {
        System.out.println("Count: " + count);
    }

    public static void main(String[] args) {
        Demo.showCount();  // Access without object
    }
}
```

✅ 2. `final` – Constant or Restrict Inheritance/Overriding

- `final` **variable**: can't be reassigned
- `final` **method**: can't be overridden
- `final` **class**: can't be inherited

🔹 Example:

```java
final class Vehicle {
    final int wheels = 4;

    final void start() {
        System.out.println("Vehicle started");
    }
}

// class Car extends Vehicle {}  // ❌ Error: cannot inherit final class
```

✅ 3. `this` – Current Object Reference

- Refers to **current class object** (used to resolve name conflicts).

🔹 Example:

```java
class Student {
    String name;

    Student(String name) {
        this.name = name;  // this refers to the current object
    }

    void showName() {
        System.out.println("Name: " + this.name);
    }
}
```

✅ 4. `super` – Parent Class Reference

- Refers to the **parent class**.
- Used to:
  - Call the **parent constructor**
  - Call **overridden methods**

🔹 Example:

```java
class Animal {
    void sound() {
        System.out.println("Animal sound");
    }
}

class Dog extends Animal {
    void sound() {
        super.sound();  // Call parent class method
        System.out.println("Dog barks");
    }
}
```

✅ 5. `return` – Exit and Send Value from Method

- Ends the method and optionally returns a value.

🔹 Example:

```java
public class MathUtil {
    int square(int n) {
        return n * n;
    }

    void printMsg() {
        return;  // just exit
    }
}
```

✅ 6. `new` – Creates a New Object

- Used to create **object instances** or arrays.

🔹 Example:

```java
class Book {
    String title = "Java";
}

public class Main {
    public static void main(String[] args) {
        Book b = new Book();  // Create new object
        System.out.println(b.title);
    }
}
```

✅ 7. `instanceof` – Type Checker

- Checks if the object is an **instance of a specific class** or subclass.

🔹 Example:

```java
public class Test {
    public static void main(String[] args) {
        String s = "Hello";
        System.out.println(s instanceof String);  // true
    }
}
```

✅ 8. Other Related Keywords:

🔸 `abstract` – for abstract classes/methods

```java
abstract class Shape {
    abstract void draw();
}
```

🔸 `interface`, `implements` – for interfaces

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

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

🔸 `extends` – for inheritance

```
java

class A {}
class B extends A {}
```

✅ Summary Table

Keyword	Purpose
`static`	Class-level method/variable
`final`	Prevents modification/inheritance
`this`	Refers to current object
`super`	Refers to parent class
`return`	Exit method and return value
`new`	Creates new object
`instanceof`	Checks object type

Would you like a **mini-project or small assignment** using all these keywords together?