Java Keywords Reference

Keyword	Description	Example
abstract	Used to declare abstract classes and methods.	abstract class Animal { abstract void makeSound(); }
assert	Used for debugging to test assumptions about the program's state.	assert x > 0 : "x must be positive";
boolean	Data type that can only be true or false.	boolean isActive = true;
break	Used to exit loops or switch statements.	while(true) { if(condition) break; }
byte	8-bit integer data type (-128 to 127).	byte smallNumber = 100;
case	Used in switch statements to define specific blocks of code to execute.	switch(x) { case 1: break; default: }
catch	Catches exceptions generated by the try block.	try { riskyCode(); } catch(Exception e) { handle(); }
char	16-bit Unicode character data type.	char letter = 'A';
class	Used to declare a class.	class MyClass { // class body }

Keyword	Description	Example
const	**Reserved keyword (not used in Java)**.	// Not used in Java - use 'final'
continue	Skips the current iteration of a loop.	for(int i=0; i<10; i++) { if(i % 2 != 0) continue; }
default	Default case in switch statements or specifies a default method in an interface (Java 8+).	switch(x) { case 1: break; default: }
do	Starts a do-while loop, guaranteeing the loop runs at least once.	do { code(); } while(condition);
double	64-bit floating-point data type.	double price = 19.99;
else	Alternative branch for if statements.	if(x>0) { positive(); } else { negative(); }
enum	Used to define a set of named constants.	enum Color { RED, GREEN, BLUE }
extends	Indicates inheritance between classes or interfaces.	class Dog extends Animal { }
final	Makes variables constant, methods non-overridable, or classes non-inheritable.	final double PI = 3.14159;
finally	Block that always executes after try-catch blocks, regardless of an	try { code(); } finally { cleanup(); }

Keyword	Description	Example
	exception.	
float	32-bit floating-point data type.	float temperature = 98.6f;
for	Loop that repeats a block of code (traditional or enhanced/for-each loop).	for(int i=0; i<10; i++) { System.out.println(i); }
goto	**Reserved keyword (not used in Java)**.	// Not used in Java
if	Conditional statement for decision-making.	if(condition) { executeCode(); }
implements	Used to implement interfaces.	class MyClass implements MyInterface { }
import	Imports packages and classes, allowing their use without a fully qualified name.	import java.util.ArrayList;
instanceof	Tests if an object is an instance of a particular class or interface.	if(obj instanceof String) { //... }
int	32-bit integer data type.	int count = 42;
interface	Used to declare an interface.	interface Drawable { void draw(); }

Keyword	Description	Example
long	64-bit integer data type.	long bigNumber = 123456789L;
native	Indicates a method is implemented in platform-specific native code (e.g., C/C++).	native void nativeMethod();
new	Creates new objects or arrays.	String str = new String("Hello");
package	Declares the package for classes.	package com.example.myapp;
private	Access modifier - accessible only within the declared class.	private int secretNumber;
protected	Access modifier - accessible within the package and by subclasses.	protected void familyMethod() { }
public	Access modifier - accessible from anywhere.	public static void main(String[] args) { }
return	Returns a value from a method and exits the method.	public int getValue() { return 10; }
short	16-bit integer data type (-32768 to 32767).	short smallInt = 1000;
static	Belongs to the class itself rather than any specific instance.	static int count = 0;

Keyword	Description	Example
strictfp	Ensures floating-point calculations adhere strictly to the IEEE 754 standard across platforms.	strictfp class Calculator { }
super	Refers to parent class members, methods, or constructors.	super(); // calls parent constructor
switch	Multi-way branch statement based on the value of a variable.	switch(day) { case MONDAY: //... }
synchronized	Provides thread synchronization for methods or blocks of code.	synchronized void safeMethod() { }
this	Refers to the current object instance.	this.name = name;
throw	Throws an exception object explicitly.	throw new IllegalArgumentException();
throws	Declares exceptions a method might throw but does not handle.	void riskyMethod() throws IOException { }
transient	Prevents a field from being serialized when an object is written to a stream.	transient String tempData;
try	Starts a block of exception-handling code.	try { riskyOperation(); }
void	Indicates a method returns no value.	public void doSomething() { }

Keyword	Description	Example
volatile	Ensures variable visibility (changes are written and read directly from main memory) across threads.	volatile boolean running = true;
while	Loop that repeats while a condition is true.	while(condition) { executeCode(); }
var	**Local variable type inference (Java 10+)**. Allows the compiler to determine the type.	var list = new ArrayList<String>();
yield	**Returns a value from a switch expression (Java 14+)**.	int result = switch(day) { default -> yield 1; };
record	**Declares a record class (Java 16+)**. A compact class for transparently storing data.	record Point(int x, int y) { }
sealed	**Controls inheritance hierarchy (Java 17+)**. A sealed class can only be extended by permitted classes.	sealed class Shape permits Circle, Square { }
non-sealed	**Makes a permitted subclass open for extension (Java 17+)**.	non-sealed class Circle extends Shape { }
permits	**Specifies allowed subclasses for a sealed class (Java 17+)**.	sealed class Shape permits Circle, Square { }