# Java Keywords and Their Uses

1. **byte**: Defines an 8-bit integer variable (range: -128 to 127). Used for saving memory.
2. **short**: Defines a 16-bit integer variable. Smaller than int.
3. **int**: Defines a 32-bit integer variable. Commonly used for whole numbers.
4. **long**: Defines a 64-bit integer variable. Used for large integer values.
5. **float**: Defines a 32-bit floating-point variable (for decimals).
6. **double**: Defines a 64-bit floating-point variable (more precise than float).
7. **char**: Defines a single 16-bit Unicode character.
8. **boolean**: Defines a variable that can store only true or false.
9. **public**: Specifies that a class, method, or variable is accessible from anywhere.
10. **private**: Restricts access to within the same class.
11. **protected**: Allows access within the same package and subclasses.
12. **class**: Defines a class (a blueprint for objects).
13. **interface**: Defines an interface (a collection of abstract methods).
14. **extends**: Indicates that a class inherits from a superclass.
15. **implements**: Used by a class to implement an interface.
16. **new**: Creates new objects (instances) of a class.
17. **this**: Refers to the current object (useful for differentiating instance and local variables).
18. **super**: Refers to the parent class (used to call superclass constructors or methods).
19. **abstract**: Defines an abstract class or abstract method (must be overridden).
20. **final**: Used to make a variable constant, prevent method overriding, or inheritance.
21. **static**: Defines class-level variables or methods shared by all instances.
22. **synchronized**: Used to control thread access to a block/method (for thread safety).
23. **volatile**: Marks a variable as "may be changed unexpectedly." Ensures visibility across threads.
24. **transient**: Excludes a variable from serialization.
25. **native**: Declares a method implemented in another language (like C).
26. **if**: Executes a block of code if the condition is true.
27. **else**: Executes a block if the if condition is false.
28. **switch**: Selects one of many code blocks to execute.
29. **case**: Defines a branch in a switch statement.
30. **default**: Defines the block that runs if no case matches in a switch.
31. **for**: Defines a loop that runs a fixed number of times.
32. **while**: Defines a loop that runs while a condition is true.
33. **do**: Used with while for a loop that runs at least once.
34. **break**: Exits a loop or switch immediately.
35. **continue**: Skips the current iteration and continues with the next one.
36. **return**: Exits from a method and optionally returns a value.
37. **yield**: Returns a value from a switch expression.
38. **try**: Defines a block of code to test for exceptions.
39. **catch**: Defines a block of code to handle exceptions.
40. **finally**: Defines a block that always executes after try (used for cleanup).
41. **throw**: Used to throw an exception manually.
42. **throws**: Declares the exceptions that a method can throw.
43. **package**: Defines a namespace (collection of related classes).
44. **import**: Imports classes or entire packages for use in the file.
45. **null**: Represents the absence of any object reference.
46. **true**: Boolean literal value representing truth.
47. **false**: Boolean literal value representing falsehood.
48. **instanceof**: Tests whether an object is an instance of a specific class or subclass.
49. **assert**: Used for debugging; tests assumptions about the program.
50. **enum**: Defines a set of named constants (enumeration).
51. **var**: Allows local variable type inference (the compiler infers the type).
52. **record**: Defines an immutable data class (used for holding data).
53. **sealed**: Restricts which classes can extend or implement a class/interface.
54. **permits**: Used with sealed to specify allowed subclasses.
55. **goto**: Reserved but not used (for backward compatibility with C).
56. **const**: Reserved but not used (use final instead).
57. **default**: Used in interfaces to define a method with a default implementation.