|  |  |  |
| --- | --- | --- |
| 1. | **abstract** | Specifies that a class or method will be implemented later, in a subclass |
| 2. | **assert** | Assert describes a predicate placed in a java program to indicate that the developer thinks that the predicate is always true at that place. |
| 3. | **boolean** | A data type that can hold True and False values only |
| 4. | **break** | A control statement for breaking out of loops. |
| 5. | **byte** | A data type that can hold 8-bit data values |
| 6. | **case** | Used in switch statements to mark blocks of text |
| 7. | **catch** | Catches exceptions generated by try statements |
| 8. | **char** | A data type that can hold unsigned 16-bit Unicode characters |
| 9. | **class** | Declares a new class |
| 10. | **continue** | Sends control back outside a loop |
| 11. | **default** | Specifies the default block of code in a switch statement |
| 12. | **do** | Starts a do-while loop |
| 13. | **double** | A data type that can hold 64-bit floating-point numbers |
| 14. | **else** | Indicates alternative branches in an if statement |
| 15. | **enum** | A Java keyword is used to declare an enumerated type. Enumerations extend the base class. |
| 16. | **extends** | Indicates that a class is derived from another class or interface |
| 17. | **final** | Indicates that a variable holds a constant value or that a method will not be overridden |
| 18. | **finally** | Indicates a block of code in a try-catch structure that will always be executed |
| 19. | **float** | A data type that holds a 32-bit floating-point number |
| 20. | **for** | Used to start a for loop |
| 21. | **if** | Tests a true/false expression and branches accordingly |
| 22. | **implements** | Specifies that a class implements an interface |
| 23. | **import** | References other classes |
| 24. | **instanceof** | Indicates whether an object is an instance of a specific class or implements an interface |
| 25. | **int** | A data type that can hold a 32-bit signed integer |
| 26. | **interface** | Declares an interface |
| 27. | **long** | A data type that holds a 64-bit integer |
| 28. | **native** | Specifies that a method is implemented with native (platform-specific) code |
| 29. | **new** | Creates new objects |
| 30. | **null** | This indicates that a reference does not refer to anything |
| 31. | **package** | Declares a Java package |
| 32. | **private** | An access specifier indicating that a method or variable may be accessed only in the class it’s declared in |
| 33. | **protected** | An access specifier indicating that a method or variable may only be accessed in the class it’s declared in (or a subclass of the class it’s declared in or other classes in the same package) |
| 34. | **public** | An access specifier used for classes, interfaces, methods, and variables indicating that an item is accessible throughout the application (or where the class that defines it is accessible) |
| 35. | **return** | Sends control and possibly a return value back from a called method |
| 36. | **short** | A data type that can hold a 16-bit integer |
| 37 | **static** | Indicates that a variable or method is a class method (rather than being limited to one particular object) |
| 38. | **strictfp** | A Java keyword is used to restrict the precision and rounding of floating-point calculations to ensure portability. |
| 39. | **super** | Refers to a class’s base class (used in a method or class constructor) |
| 40. | **switch** | A statement that executes code based on a test value |
| 41. | **synchronized** | Specifies critical sections or methods in multithreaded code |
| 42. | **this** | Refers to the current object in a method or constructor |
| 43. | **throw** | Creates an exception |
| 44. | **throws** | Indicates what exceptions may be thrown by a method |
| 45. | **transient** | Specifies that a variable is not part of an object’s persistent state |
| 46. | **try** | Starts a block of code that will be tested for exceptions |
| 47. | **void** | Specifies that a method does not have a return value |
| 48. | **volatile** | This indicates that a variable may change asynchronously |
| 49. | **while** | Starts a while loop |