Identifiers:

A name in java program is called identifier. It may be class name, method name, variable name and label name.

Rules to define java identifiers:

1) The only allowed characters in java identifiers are:
   - → a to z
   - → A to Z
   - → 0 to 9
   - → _ (underscore)
   - → $

2) If we are using any other character we get compile time error.

   Eg: total_number → valid.
   Total## → invalid.

3) Identifiers are not allowed to start with digit

   Eg: ABCD123 → valid
   1234ABCD → invalid.

4) Java identifiers are case-sensitive because the Java language itself is case sensitive

   Eg:
```
class Test{
    int number = 10;
    int Number = 20;
    int NUMBER = 20;
    int NuMbEr = 30;
}
```
   } we can differentiate with case

5). There is no length limit for java identifiers but it is not recommended to take more than 15 lengths.

6). We can't use reserved words as identifiers.

Eg: int if = 10; ⟶ invalid.

7). In Java, predefined class and interface names (like String, List, Math) can be used as identifiers for variables & methods, but its not recommended because it causes confusion.

Eg: int String = 5; // legal, but confusing.
S.O.P (String); // Prints 5, not a String object.

Q: Why Java allows this even though its risky?

1). Only keywords are reserved.

* Java reserves a fixed set of keywords (like class, if, while, static etc)
* Predefined class names (String, System, List etc) are not keywords - they are just names from Java API.
* Since they are not reserved, you can legally reuse them as identifiers.

2) Compiler can tell the difference:

* When you write: `String s = "Hello";`
the compiler knows: a) First String = a reference to predefined class java.lang.String.
b) s = new variable name.
* If you write: `int String = 10;` now String is just a variable name. In that scope, you've shadowed class name, so can't directly

refer to java.lang.String unless you use the full package name:

```
int String = 10;
java.lang.String text = "Hello"; // must fully qualify
```

3) Why it's risky?

- It confuses both reader & the compiler context.
- Can lead to bugs if you accidentally shadow an important type.
- Makes code harder to read and maintain.

Best Practice: Avoid reusing predefined Java class (or) interface names for your identifiers.

⇒ It's legal, but it's like naming your dog "Dog"
↳ technically correct, but you'll confuse everyone at the park.

Q: Which of the following are valid Java identifiers?

1) myVar
2) _count
3) $price
4) 2ndValue ✗
5) total$Sum
6) class ✗
7) My_Var_1
8) @amount
9) HelloWorld
10) void ✗

Java identifier rules recap:

1) Can contain letters, digits, _ & $.
2) Cannot start with a digit
3) Cannot be a reserved keyword.
4) Can be of any length.
5) Case sensitive (myVar ≠ MyVar)