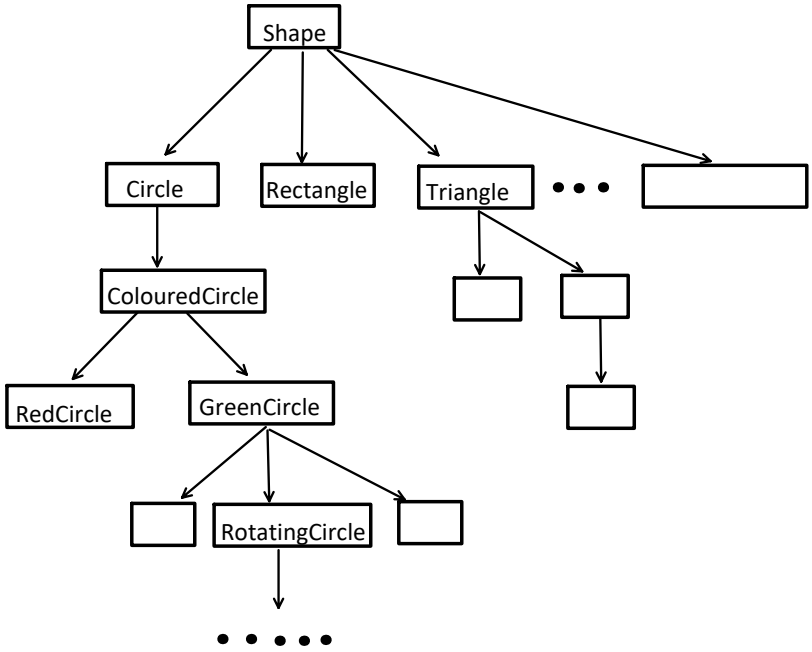
Java 17: Sealed Classes and Interfaces

Wednesday, 13 August 2025 7:53 PM

Lets understand the problem first:

Biggest problem is Lack of Control in Inheritance



Its not possible to control the hierarchy, any class can join it by implementing the interface or extending the class, even we did not plan for it.

Like in above use case: say I never planned for a "RotatingCircle" class to be part of this hierarchy, but I can not prevent someone from extending the parent class and adding it.

```
void area(Shape  shape)
{
   if (shape instanceof Circle) {
        //do something
    }
   else if (shape instanceof Rectangle) {
        //do something
    }
        •
        •
        •
    else {
        throw exception
    }
}
```

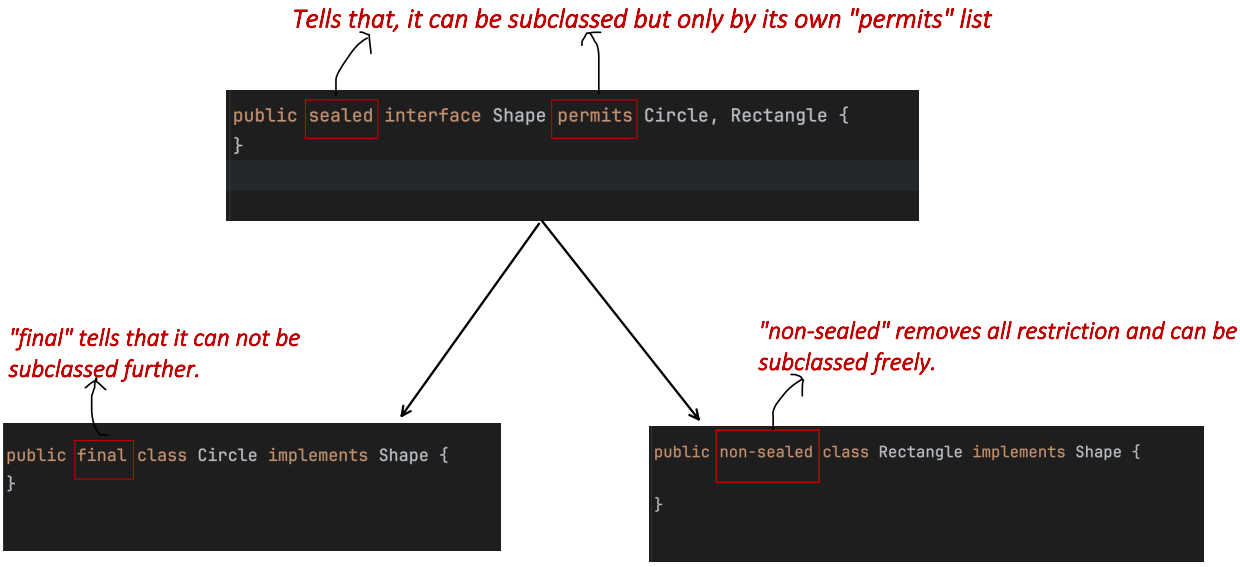
OR

```
void area(Shape  shape)
{
    switch (shape.getType()) {
        case CIRCLE:
                // do something
                 break;
        case RECTANGLE:
                // do something
                break;
            •
            •
            •
        default:
                throw exception
    }
}
```

This default condition becomes important, so that we can handle unexpected subclass safely

So how to bring the control, like who can extend or implement an interface or class?

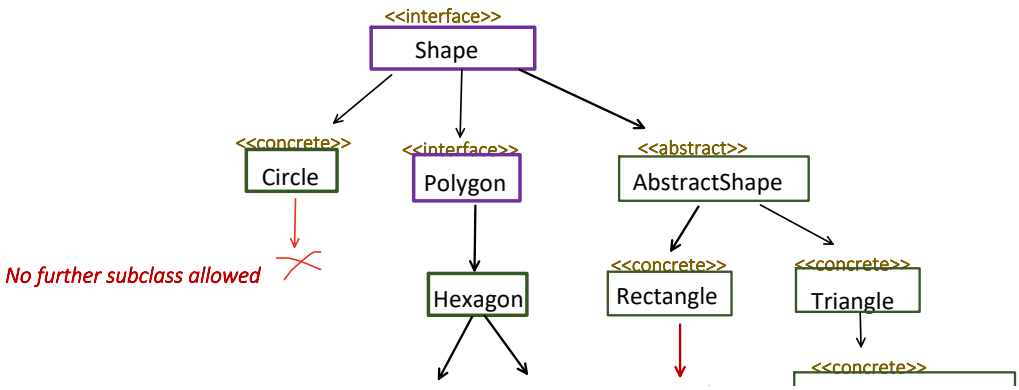
We can achieve this through Sealed classes/interfaces in java.



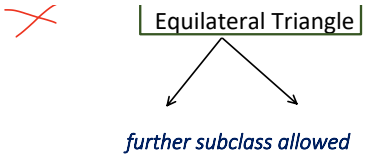
Few things to consider are:

- "permits" type should be direct subclass of sealed interface or class.
- "permits" type, should be either "final", "sealed" or "non-sealed".
- All "permits" type should be present (future classes or interfaces not considered).

Lets try to implement below hierarchy:



further subclass allowed



further subclass allowed