Java SE 17 Creational Design Patterns

Singleton Pattern

Bryan Hansen
Director of Software Development

@bh5k

Version Check

This version was created by using:

- Java 17
- Maven 3
- IDE

Concepts

Only one instance created

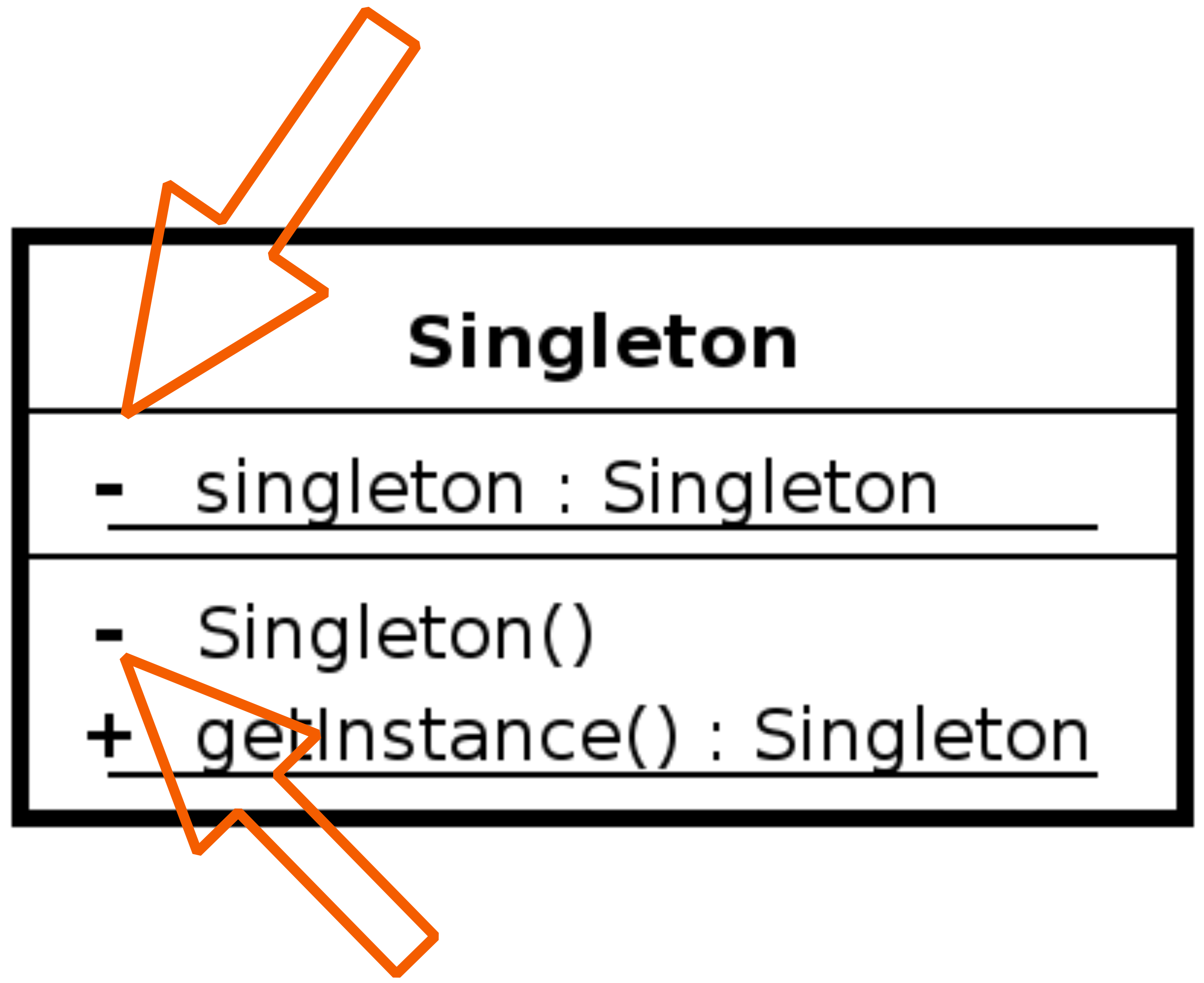
Guarantees control of a resource

Lazily loaded

Examples:

- Runtime
- Logger
- Spring Beans
- Graphic Managers

Design



Class is responsible for lifecycle

Static in nature

Needs to be thread safe

Private instance

Private constructor

No parameters required for construction

Everyday Example - Runtime Env

```
Runtime singletonRuntime = Runtime.getRuntime();

singletonRuntime.gc();

System.out.println(singletonRuntime);

Runtime anotherInstance = Runtime.getRuntime();

System.out.println(anotherInstance);

if(singletonRuntime == anotherInstance) {
      System.out.println(“They are the same instance”);
}
```

Demo

Create Singleton

Demonstrate only one instance created

Lazy Loaded

Thread safe operation

Pitfalls

Often overused

Difficult to unit test

If not careful, not thread-safe

Sometimes confused for Factory

- Prototype

Contrast

Singleton	Factory
Returns same instance	Returns various instances
One constructor - no args	Multiple constructors
No Interface	Interface driven
	Adaptable to environment more easily

Summary

Guarantee one instance

Easy to implement

Solves a well defined problem

Don’t abuse it

Consider Factory is Singleton doesn’t fit