Singleton Pattern

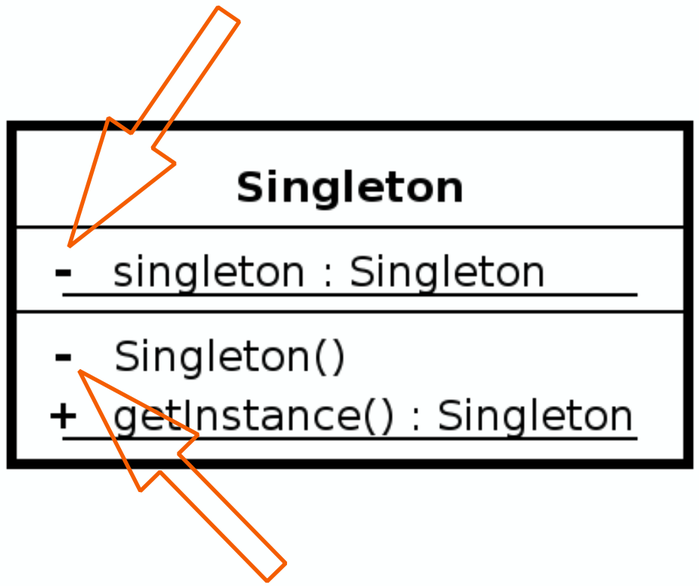
Bryan Hansen

twitter: bh5k | http://www.linkedin.com/in/hansenbryan

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”);
}
```

Exercise Singleton

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
- java.util.Calendar is NOT a Singleton
  - Prototype

Contrast

Singleton

- Returns same instance
  - One constructor method - no args
- No Interface

Factory

- Returns various instances
  - Multiple constructors
- Interface driven
- Adaptable to environment more easily

Singleton Summary

- Guarantee one instance
- Easy to implement
- Solves a well defined problem
- Don’t abuse it