# Singleton
# Design Pattern

## Singleton

### Intent

Ensure a class has only one instance, and provide a global point of access to it.

Encapsulated "just-in-time initialization" or "initialization on first use".

## Singleton

### Motivation

We only need single instance of a concrete factory class (Abstract Factory design pattern).

We could do with a sole instance of the NullImage object in our previous example.

## Singleton

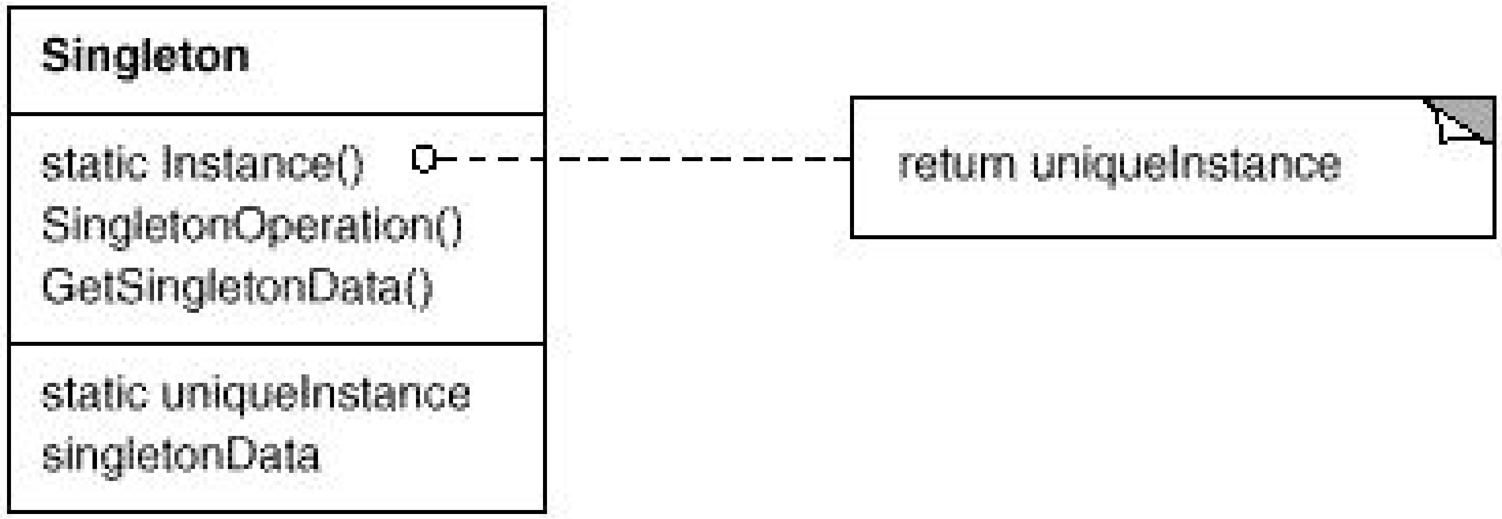
### Applicability

Use the Singleton pattern when

> there must be exactly one instance of a class, and it must be accessible to clients from a well known access point.
>
> when the sole instance should be extensible by subclassing, and clients should be able to use an extended instance without modifying their code.

## Singleton

### Structure



## Singleton

### Participants

Singleton

> defines an Instance operation that lets clients access its unique instance. Instance is a class operation (that is, a static method in Java).
>
> may be responsible for creating its own unique instance.

## Singleton

### Consequences

The Singleton pattern has several benefits

- Controlled access to sole instance.
- Permits a variable number of instances.
- Permits refinement of operations and representation.
- More flexible than class operations. Anyway, static methods in Java are never virtual, so subclasses can't override them. Polymorphism is lost.

# Singleton

## Implementation

## Sample Code

Check the code in the *src* folder

## Known Uses

## Related Patterns

Abstract Factory

Builder

Prototype