Singleton Pattern – Creational

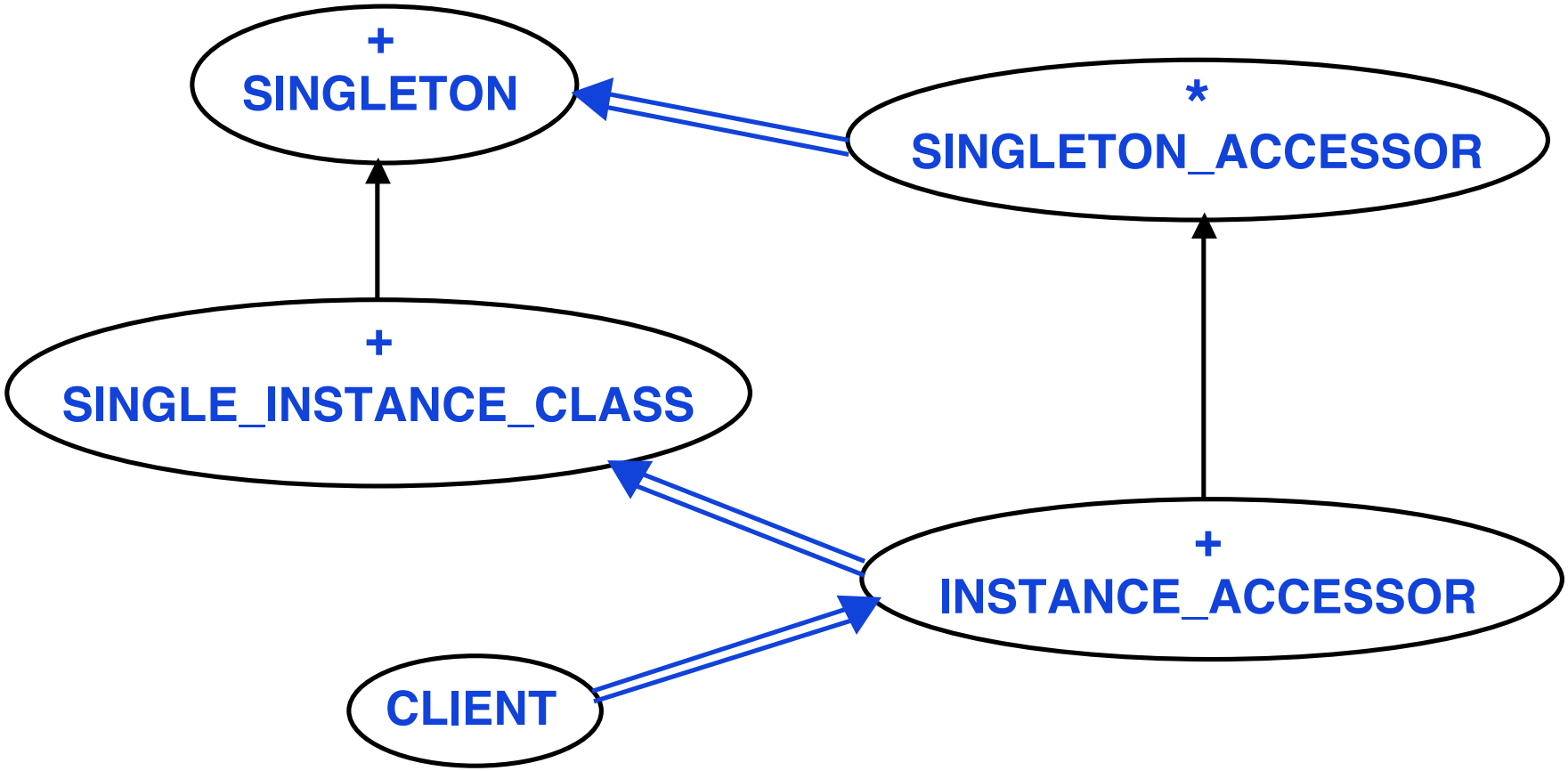
- Intent
  - » **Ensure a class has only one instance**
  - » **Provide a global point of access**
- Motivation

  Some classes must only have one instance

  file system, window manager
- Applicability
  - » **Must have only one instance of a class**
  - » **Must be accessible from a known location**

Singleton 1 – Abstract Architecture

Specific to Eiffel
– has once function but not static variables



Singleton – Participants

- Singleton

  Used to type a class as a singleton

- Single instance class

  The class that should have only one instance

- Singleton accessor

  Declares access point for a single instance

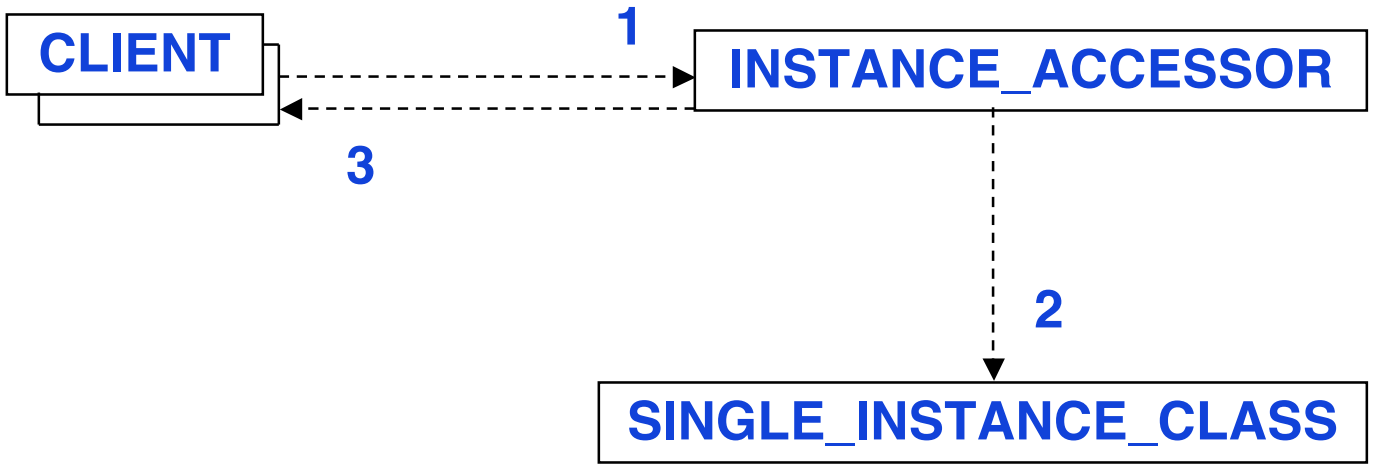
- Instance accessor

  Access point (storage location) for the single instance

- Client

  Uses instance accessor to get the single instance

Singleton – Scenario



Scenario: Get instance

1 Create instance_accessor
2 Create the_instance
(only once)
3 Get the_instance

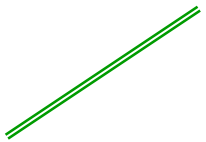
Singleton 1 Class

```
class SINGLETON

feature {NONE}
        frozen the_singleton : SINGLETON
                -- The unique instance of this class
        once
                Result := Current
        end

invariant
  only_one_instance:  Current = the_singleton

end
```



Enforces single instance property

Singleton Accessor Class

```
deferred class SINGLETON_ACCESSOR

feature {NONE}

        singleton : SINGLETON
                -- Access to a unique instance.
                -- Must be redefined as once function.
                deferred end

        is_real_singleton : BOOLEAN
                do
                        Result := singleton = singleton
                end

invariant
  singleton_is_real_singleton: is_real_singleton
end
```

Enforces single instance property

Instance Accessor Class

```
class INSTANCE_ACCESSOR

inherit SINGLETON_ACCESSOR
        rename singleton as the_instance end

feature
        the_instance: SINGLE_INSTANCE_CLASS
                -- Create the only instance in the system
                once
                        create Result.make(…)
                end

end
```

Singleton 1 Single_Instance Class

```
class SINGLE_INSTANCE

inherit SINGLETON

...

end
```

Only need to inherit from SINGLETON class.
No other changes

Singleton 1 – Consequences

- Sole instance is extensible by sub-classing

  Clients use extended instance without modification dynamically

- Reduce name space

  Avoids adding global variables storing single instance

Singleton 1 – Problem

As defined only one SINGLETON is permitted in the system.

The once feature in SINGLETON is common to all instances

The solution is to have a once feature for each needed singleton

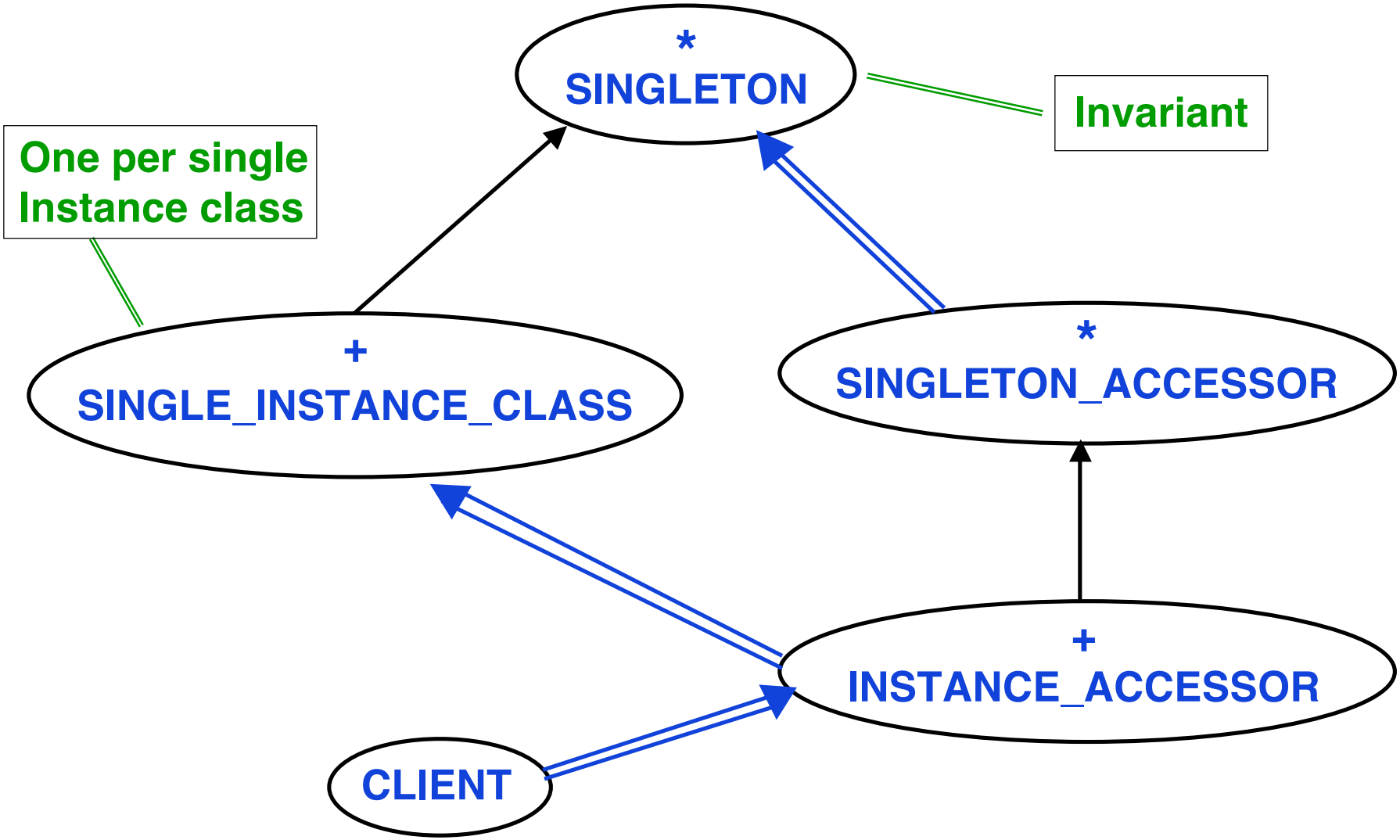
The invariant remains in the SINGLETON class

Singleton – Solution 2

- SINGLETON class – as for solution 1
  - » **Make the Singleton class deferred**
  - » **Make the_singleton deferred**
  - » **Keep the invariant**

- SINGLE_INSTANCE class
  - » **Inherit from SINGLETON**
  - » **Make the_singleton effective**

Solution 2 – Abstract Architecture



Singleton Class – Solution 2

```
deferred class SINGLETON

feature {NONE}
        the_singleton : SINGLETON
                -- The unique instance of this class
                -- Should be redefined as a once function
                -- returning Current in concrete subclasses
                deferred end

invariant
  only_one_instance:  Current = the_singleton

end
```

Enforces single instance property

Singleton 2 Single_Instance Class

```
class SINGLE_INSTANCE

inherit SINGLETON

feature {NONE}
        frozen the_singleton : SINGLETON
            -- The unique instance of this class
        once
                Result := Current
        end
...

end
```

Add to the single instance class
- Inherit from SINGLETON class.
- Make the_singleton effective

Solution 1 & 2 Tradeoffs

- Solution 1
  - » **Only need to inherit from SINGLETON**
  - » **Compiler catches invalid create attempts**

- Solution 2
  - » **In addition to inheriting from SINGLETON, need to add the feature the_singleton**
  - » **Invalid create attempts can only be caught at run time**

Singleton – Related Patterns

- Abstract Factory, Builder and Prototype can use Singleton

Memory Diagram