Singleton Pattern – Creational

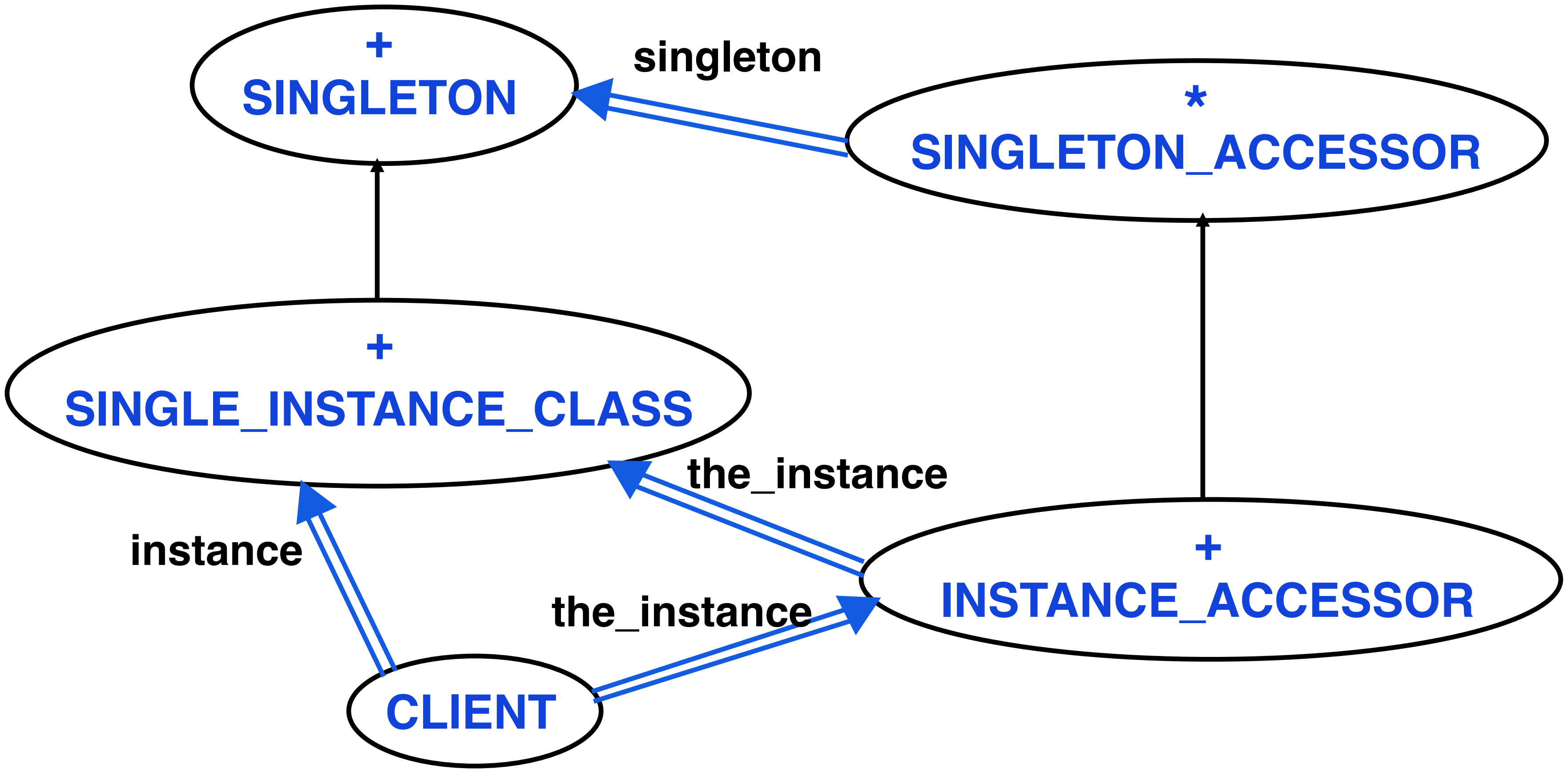
- Intent
  - » **Ensure a class has only one instance**
  - » **Provide a global point of access**
- Motivation

  Some classes must only have one instance

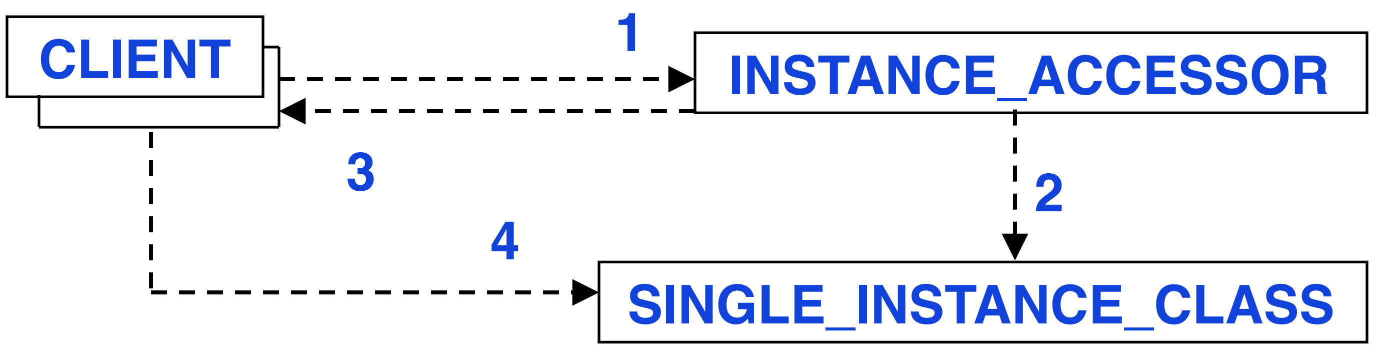
  file system, window manager
- Applicability
  - » **Must have only one instance of a class**
  - » **Must be accessible from a known location**

One Singleton – Abstract Architecture

Eiffel has once function but not static variables
More complex architecture



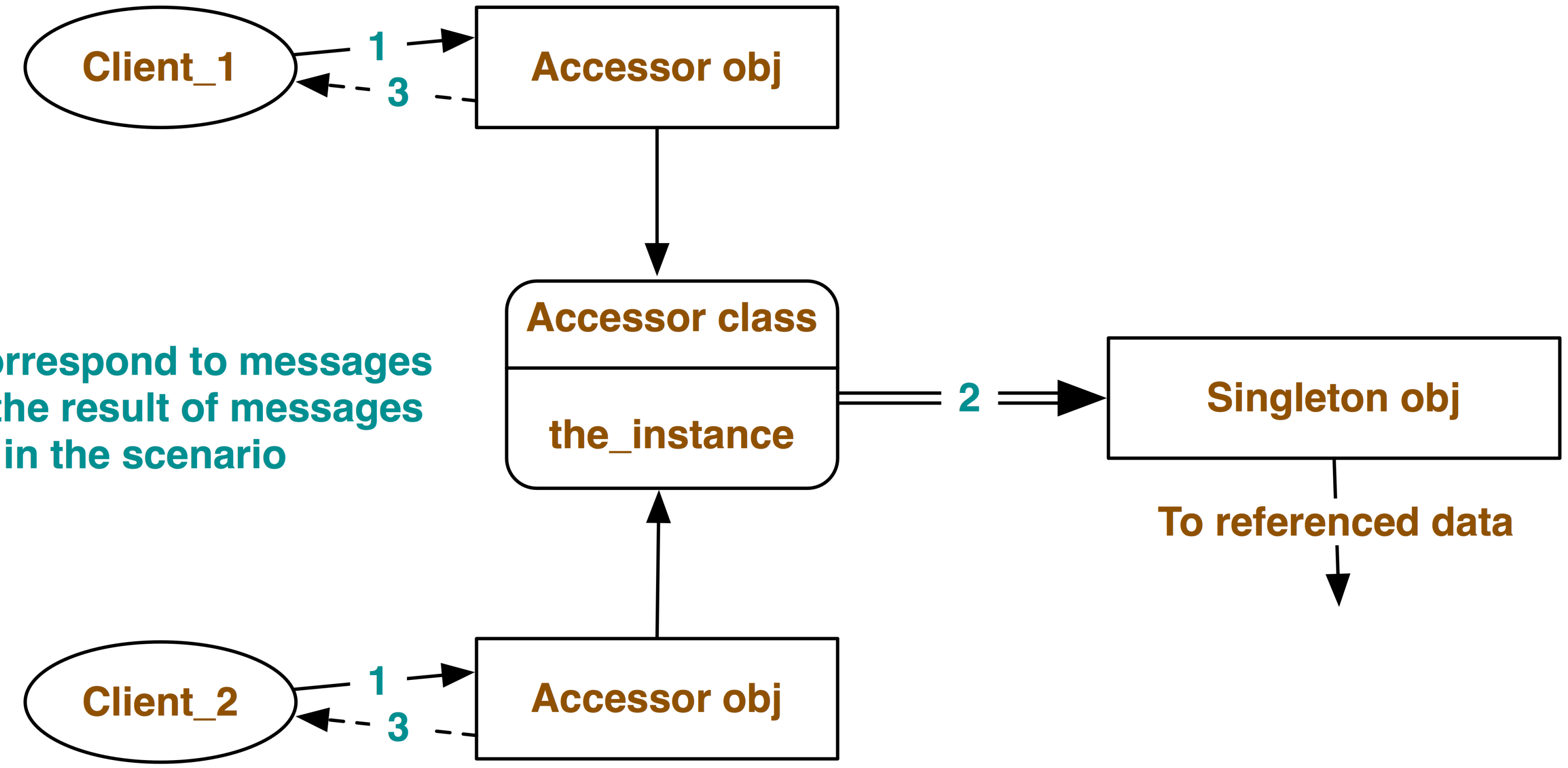
Scenario



Scenario: Get & use instance

1	Create instance_accessor
2	Create the_instance -- only once, thereafter return it
3	Get the_instance
4	Use instance

Memory Diagram



1,2,3 correspond to messages or are the result of messages in the scenario

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 for the single instance

- Client

  Uses instance accessor to get the single instance

One Singleton Class

```
class SINGLETON

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


invariant                Enforces single instance property
  only_one_instance:  Current = the_singleton

end
```

Singleton Accessor Class

```
deferred class SINGLETON_ACCESSOR
feature {NONE}
        singleton : SINGLETON
                -- Access to unique instance.
                -- Must be redefined as once function.
              deferred end
        is_real_singleton : BOOLEAN
                do
                        Result := singleton = singleton
                end


invariant               Enforces single instance property
  singleton_is_real_singleton: is_real_singleton
end
```

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
```

One Singleton Single_Instance Class

```
class SINGLE_INSTANCE

inherit SINGLETON

...

end
```

Only need to inherit from SINGLETON class.
No other changes

One Singleton – 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

One Singleton – Problem

As defined only one SINGLETON is permitted in the system.

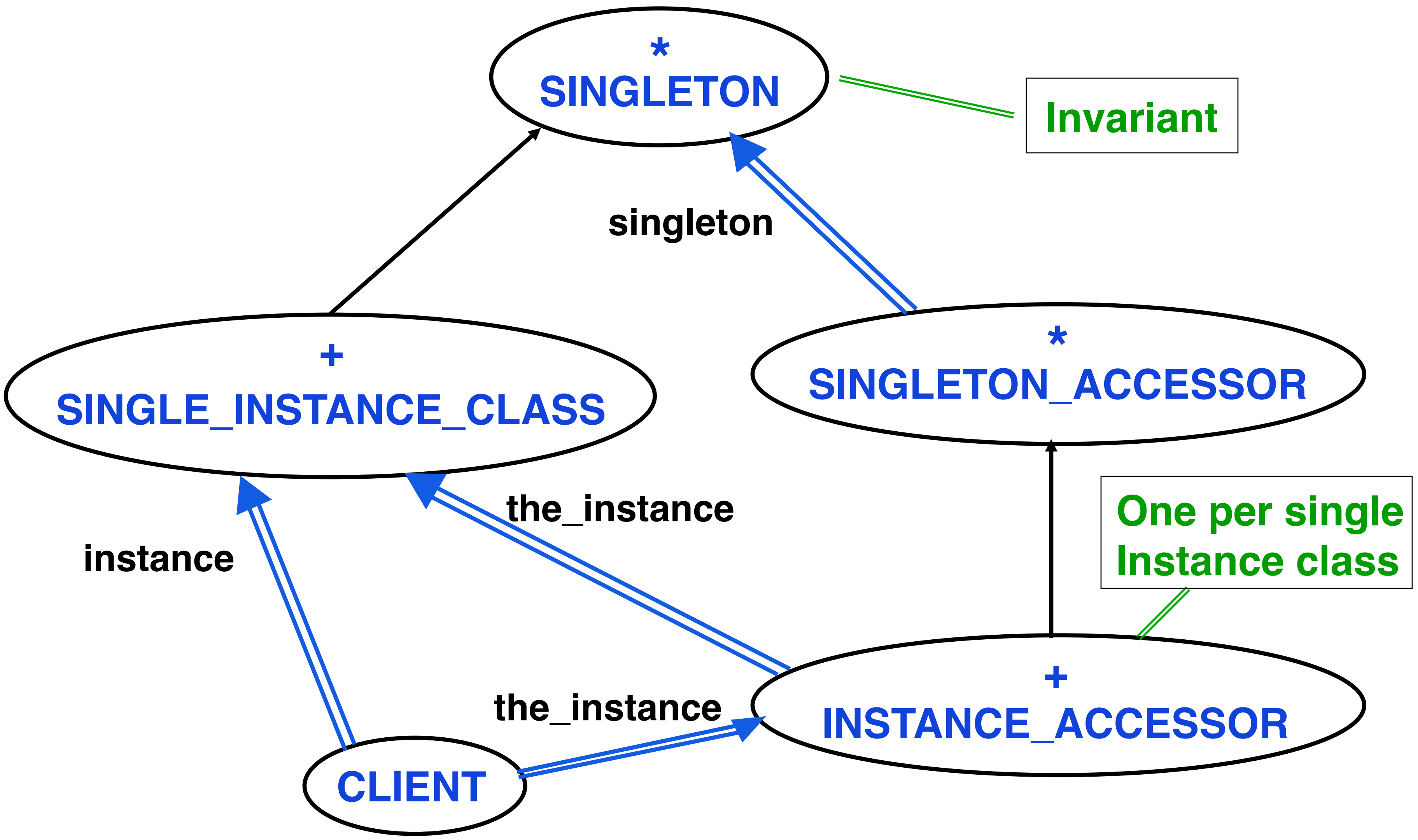
- **The once feature in SINGLETON is common to all instances**
  - **The solution is to have a once feature for each needed singleton**
  - **The invariant remains in the SINGLETON class**

Multiple Singletons

- 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**

Multiple Solution – Abstract Architecture



Multiple Singleton Class

```
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                  Enforces single instance property
  only_one_instance:  Current = the_singleton

end
```

Multiple Singleton 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

Tradeoffs

- One singleton technique
  - » **Only need to inherit from SINGLETON**
  - » **Compiler catches invalid create attempts**

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

Related Patterns

- Abstract Factory, Builder and Prototype can use Singleton

Singleton Java class AcctNumber

- Singleton is easy due to having static variables

```
public class AcctNumber {
  private AcctNumber () { /* Only AcctNumber can construct  */  }

  private static AcctNumber instance = null;

                                              Give client access to the single instance

  public static AcctNumber getInstance() {
      if ( instance == null ) { instance = new AcctNumber(); }
      return instance; }

          /* See next slide for Singleton data and data access */
}
```

Singleton Java class AcctNumber – 2

```
// The singleton data is not directly accessible

  private int lastAcctNumber = 0;

// Give clients appropriate access to the data

  public int getNumber { return lastAcctNumber; }

  public void nextAcctNumber { lastAcctNumber++; }
```

Singleton Java class AcctNumber – 3

- Client side

```
// Customer 1 wants a couple of account numbers

AcctNumber customer_1 = AcctNumber.getInstance();
customer_1 . nextAcctNumber();
acct_number = customer_1 . getNumber();   … use acct_number

customer_1 . nextAcctNumber();
acct_number = customer_1 . getNumber();   … use acct_number

// Customer 2 wants an account number

AcctNumber customer_2 = AcctNumber.getInstance();
customer_2 . nextAcctNumber();
acct_number = customer_2 . getNumber();   … use acct_number
```