Service Locator vs. Object-Oriented Code

Zoran Horvat
CTO at InterVenture GmbH

@zoranh75 www.codinghelmet.com

Service Locator

Commonly known as anti-pattern

- But, is it the pattern,
  or its use that is anti-?

There are legitimate uses of the Service Locator pattern

Try to understand benefits vs. drawbacks of the Service Locator pattern

We use Service Locator all the time

Calling DateTime.Now or DateTime.UtcNow comes with consequences

- Code that depends on time is hard to test

How would you test this *IsValid* property?

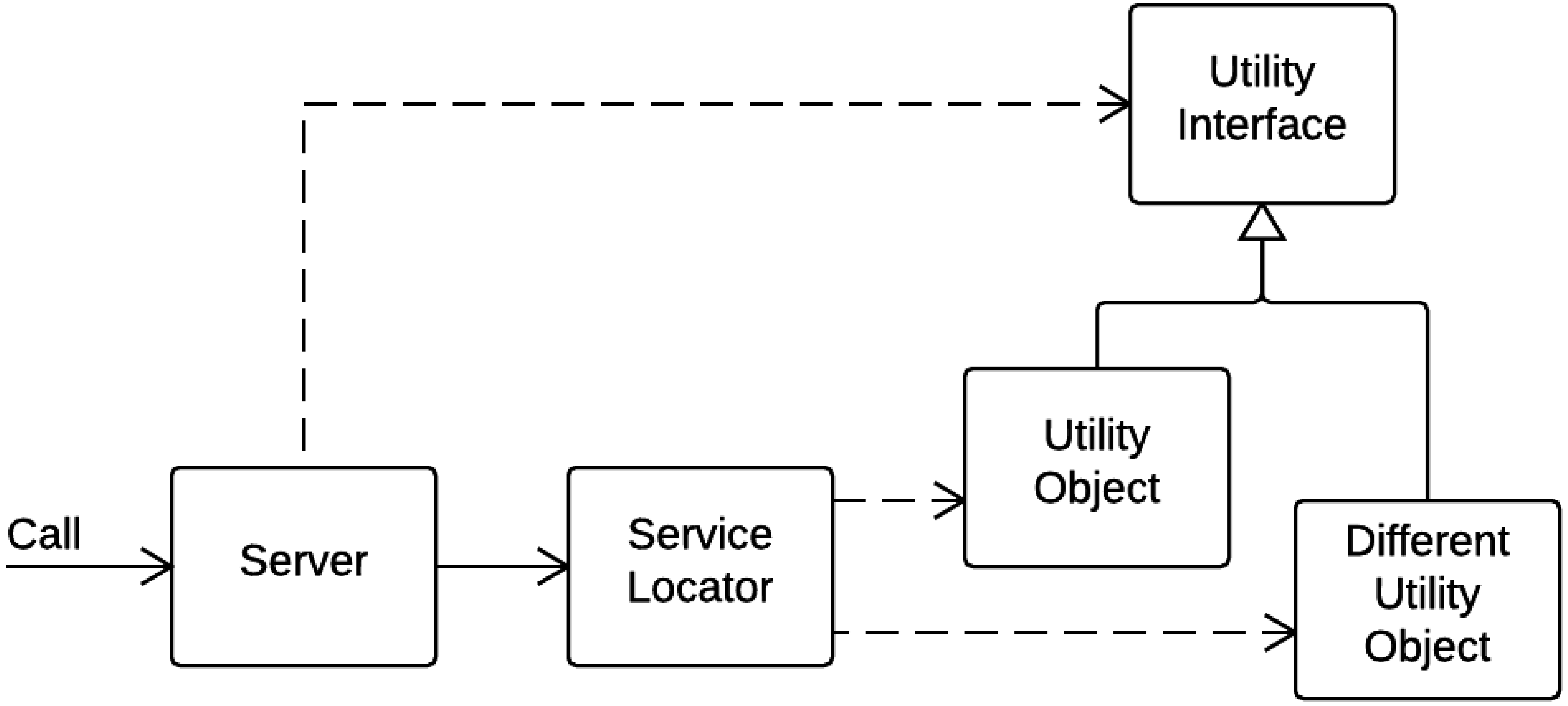
```
if (DateTime.UtcNow <= this.expiresAt)
    return true;
```

```
class Token
{
    private DateTime expiresAt;

    public Token(TimeSpan expirationPeriod)
    {
        this.expiresAt = DateTime.UtcNow.Add(expirationPeriod);
    }

    public bool IsValid
    {
        get
        {
            return DateTime.UtcNow <= this.expiresAt;
        }
    }
}
```

The Purpose of Service Locator



What Follows in This Module

Service Locator vs. Dependency Injection analysis

Service Locator applied to different situations

- DateTime structure as Service Locator
- Generic message handler with multiple message types

When to Apply Service Locator

Service Locator breaks Object-Oriented Design principles

- It damages the design which is highly object-oriented

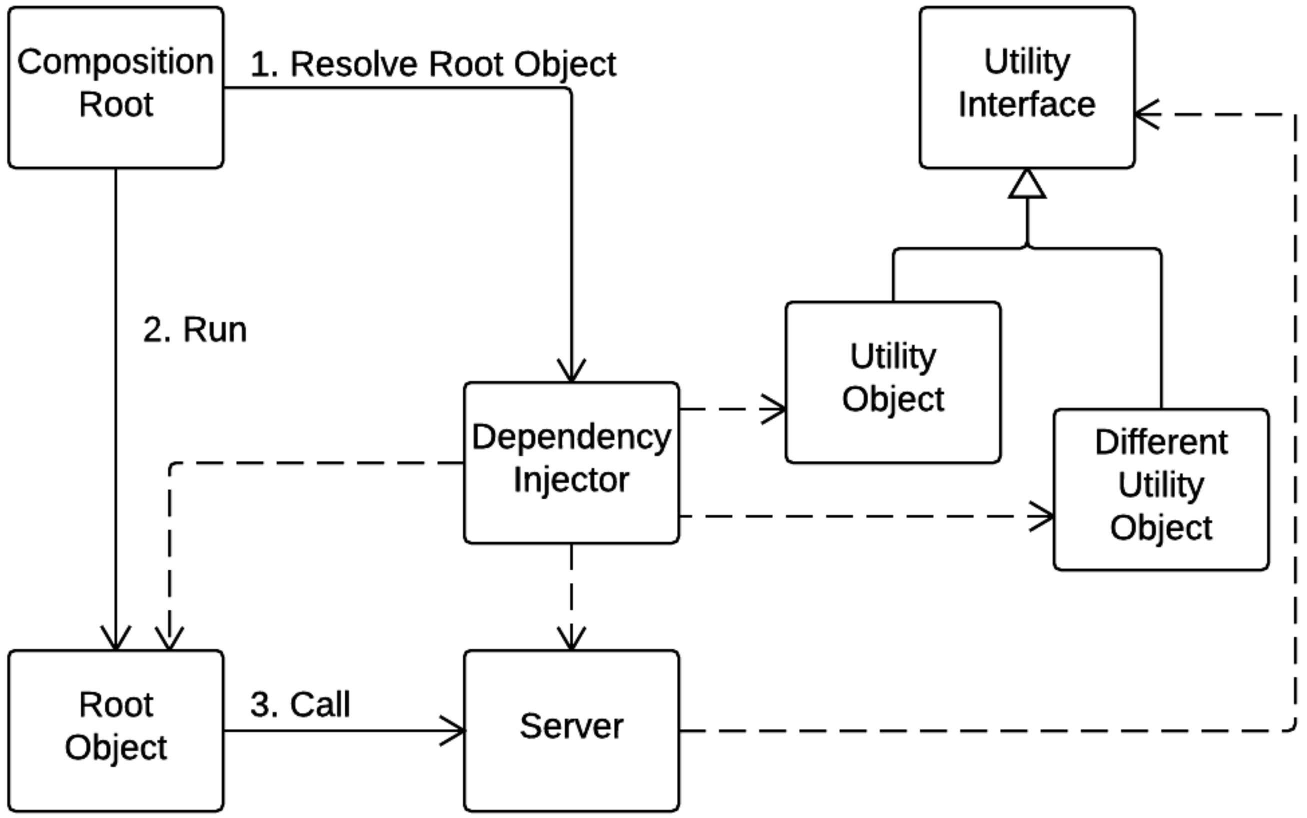
Parts of code are never object-oriented

- Network adapters,
- User interface…

Next in this module: Service Locator vs. Dependency Injection

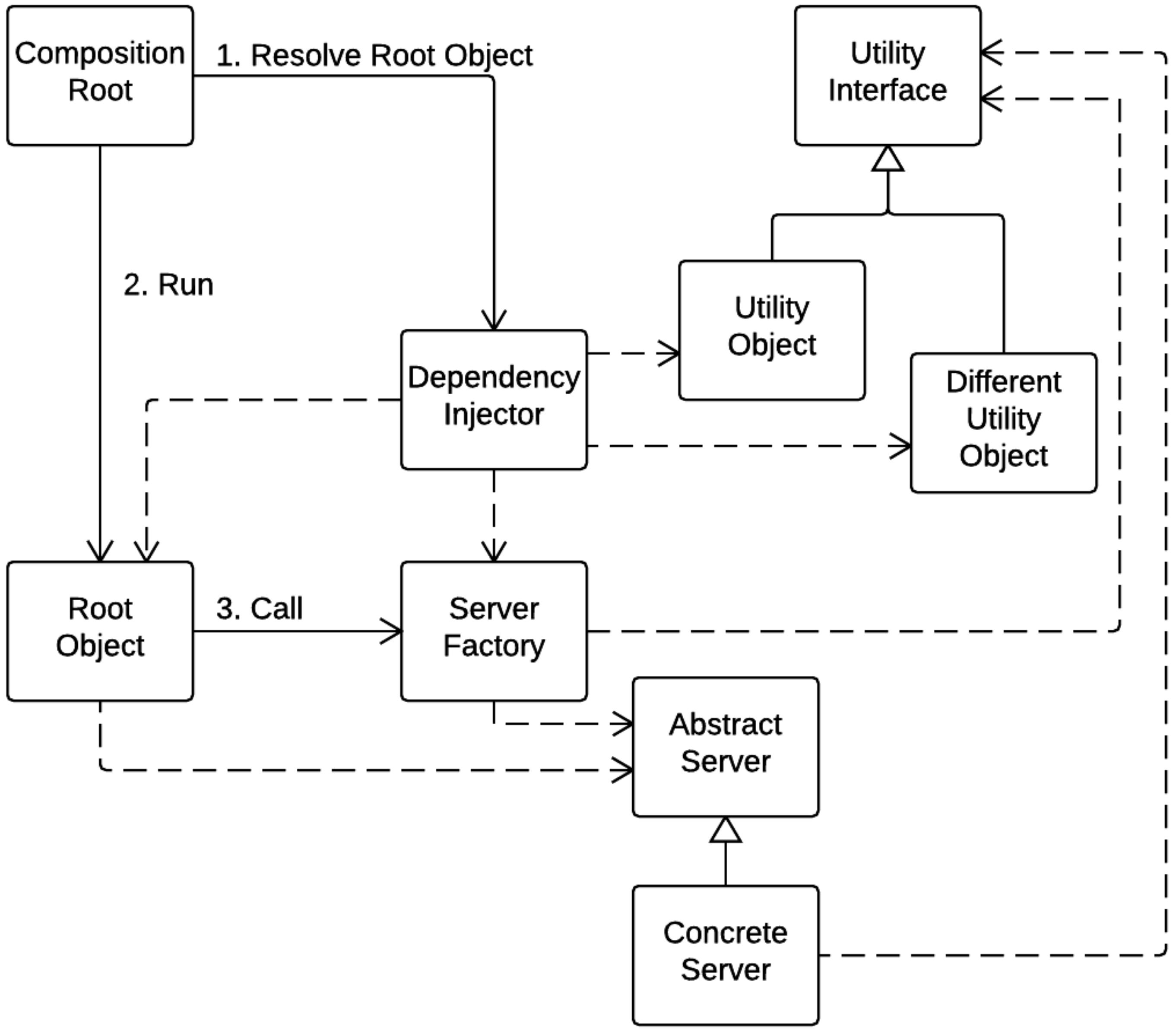
- … or how not to use dependency injection like a service locator…

Injecting Dependencies



1. **Construct dependency injector**
2. **Resolve root object**
3. **Invoke root object**
4. **Terminate after execution**

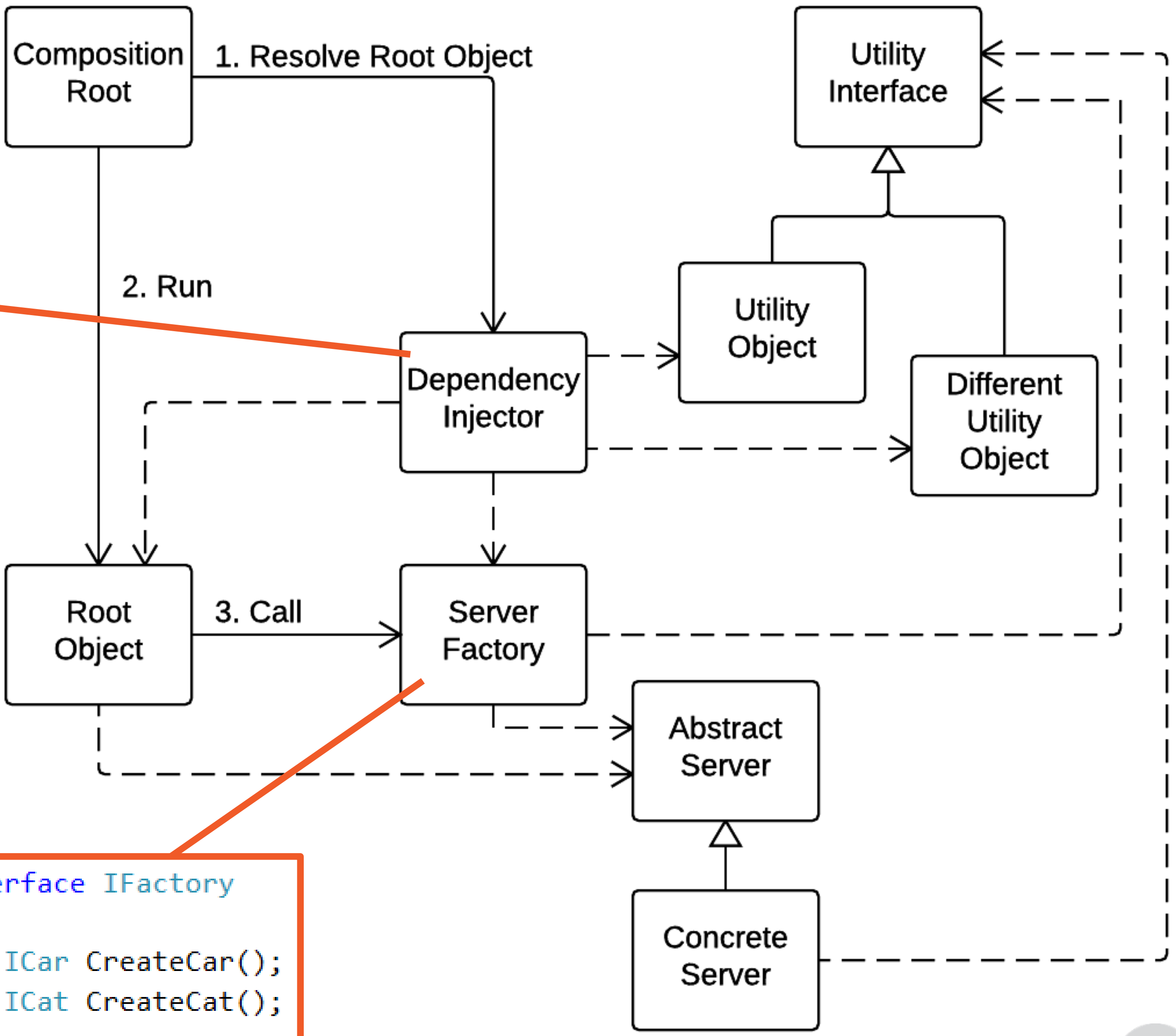
Injecting Dependencies



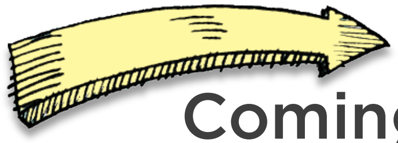
```
interface IServiceLocator
{
    T ResolveService<T>();
}
```

```
interface IDependencyInjector
{
    T Resolve<T>();
}
```

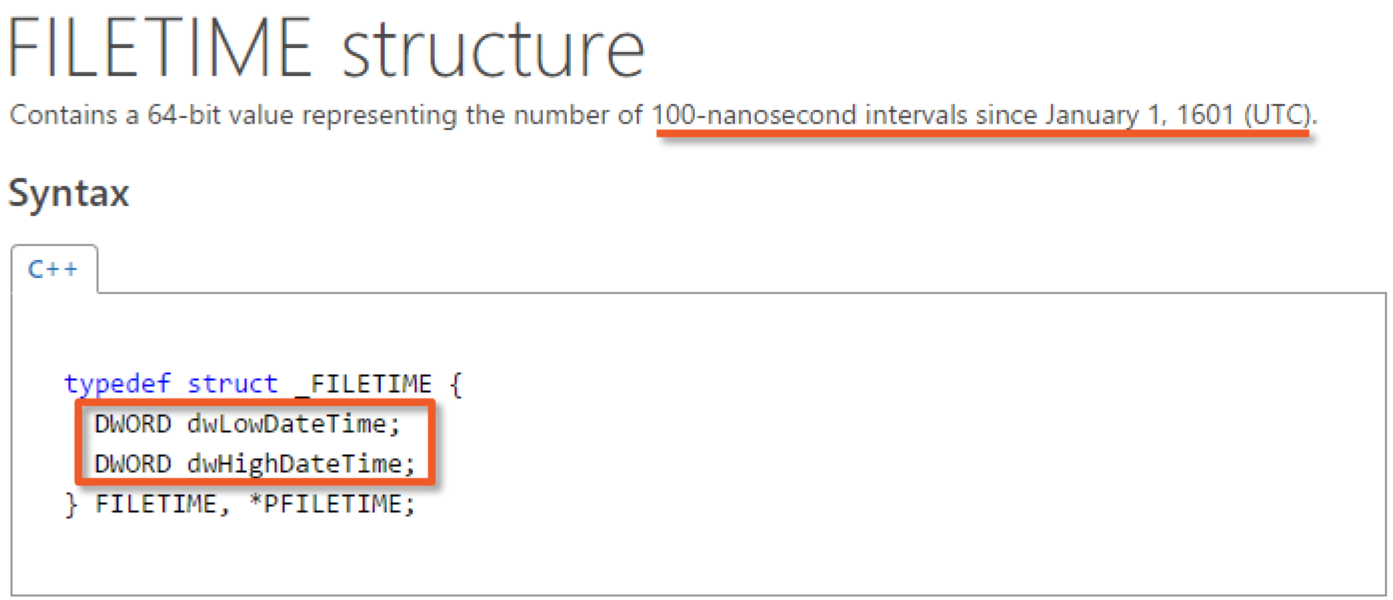
```
ICar car = dep.Resolve<ICar>();
ICat cat = dep.Resolve<ICat>();
```



```
interface IFactory
{
    ICar CreateCar();
    ICat CreateCat();
}
```



Coming next:
DateTime structure as service locator



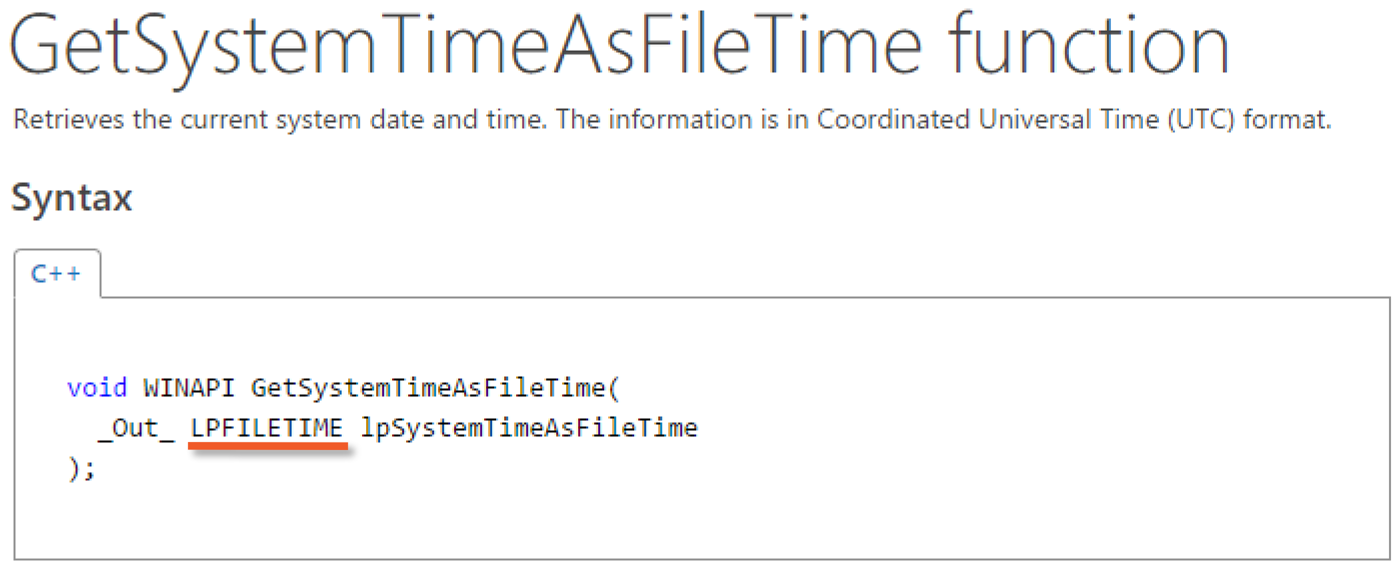
FILETIME structure

Contains a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

Syntax

C++

```
typedef struct _FILETIME {
  DWORD dwLowDateTime;
  DWORD dwHighDateTime;
} FILETIME, *PFILETIME;
```



GetSystemTimeAsFileTime function

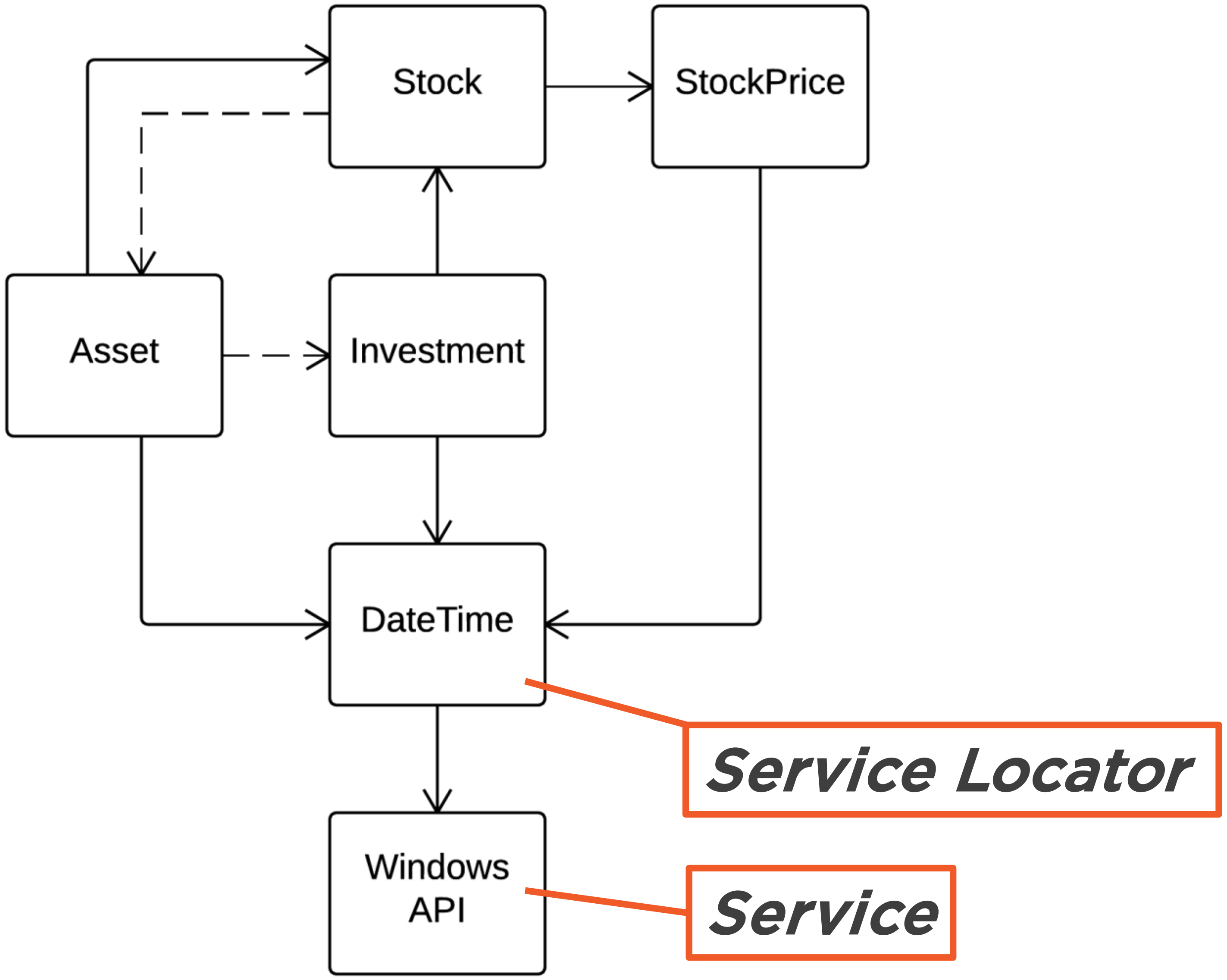
Retrieves the current system date and time. The information is in Coordinated Universal Time (UTC) format.

Syntax

C++

```
void WINAPI GetSystemTimeAsFileTime(
  _Out_ LPFILETIME lpSystemTimeAsFileTime
);
```

DateTime Demo



Testing in Presence of Service Locator

The test has passed. Now what?

- Is the code correct?
  - We don't know!

Test was performed at specific time

- At some other time it could as well fail
  - February 29th
  - Daylight saving change
  - During the leap second
  - December 31st midnight
  - Any midnight

Testing in Presence of Service Locator

Service works the same during testing and in production

- Regular operation might not provoke defects to show up during testing
- Tests that do not provoke defects add no value

Execution depends on time

- We have to change local time during testing
- That is often impossible
  - Operating system detects misaligned clocks as an intrusion
- Test that depends on time cannot be performed reliably

Testing in Presence of Service Locator

Other examples where Service Locator ruins tests

- Logging as a requirement
  - Hard to test audit and logging services

Network communication as a requirement

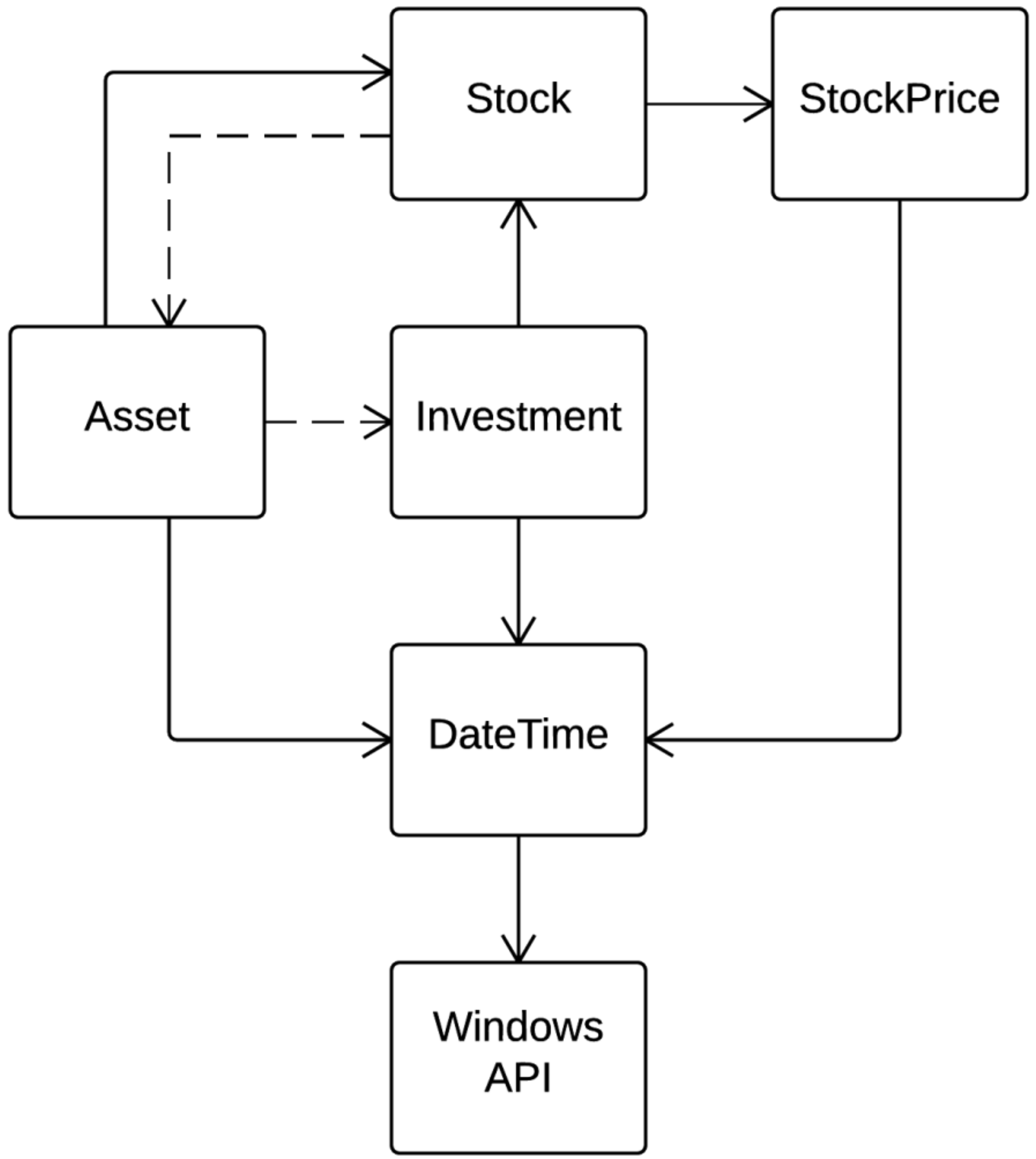
- Hard to test special cases
  - DNS errors
  - Network timeout
  - Name issues

Testing in Presence of Service Locator

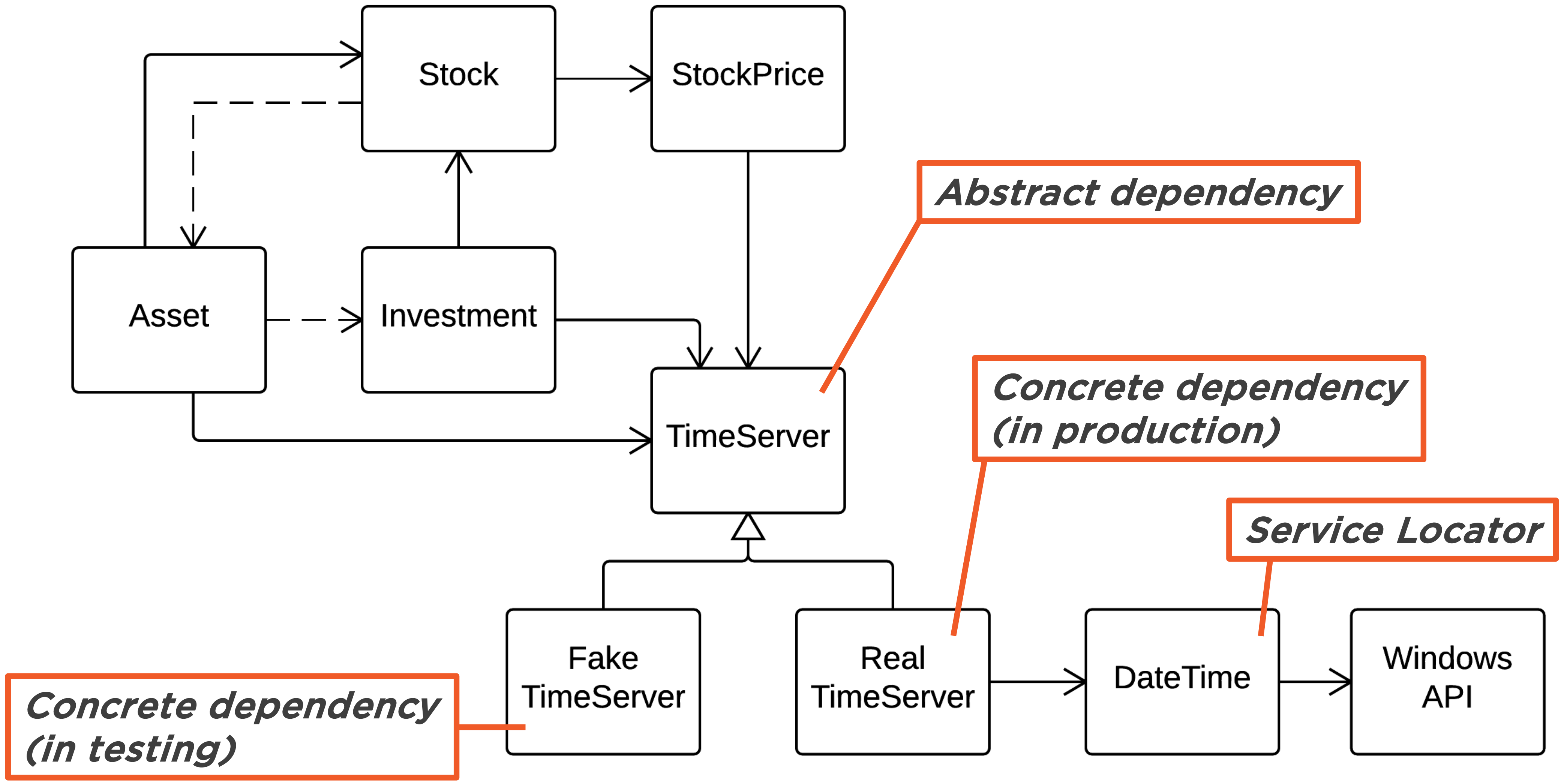
If a concept is a requirement, then try not to implement it as Service Locator

- Use proper abstraction instead
- Inject abstraction as a dependency

Removing Service Locator



Removing Service Locator



Message Handling Example

How do we handle messages?

- Object-oriented approach
- Procedural approach

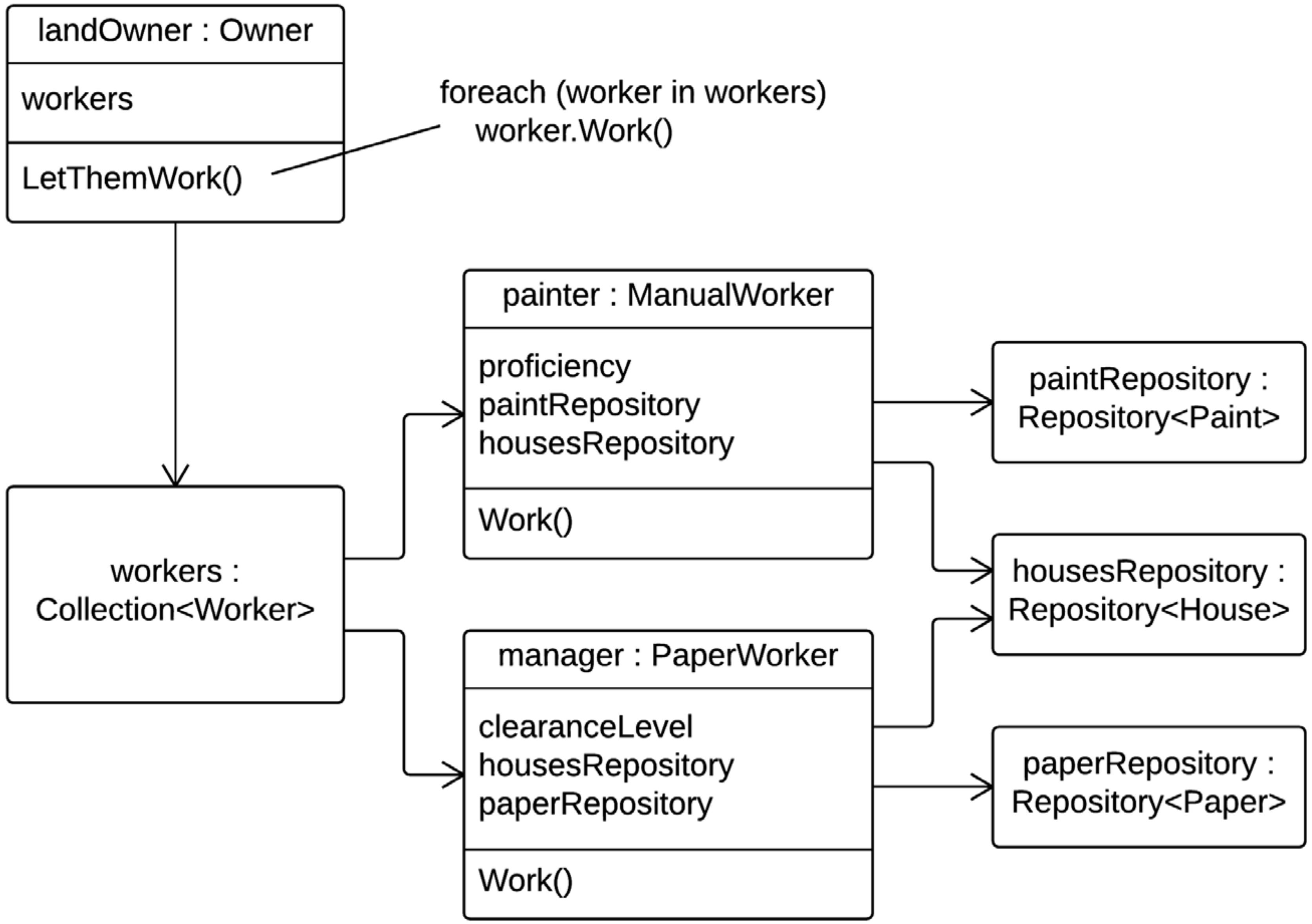
Object-oriented implementation

- Message executes itself

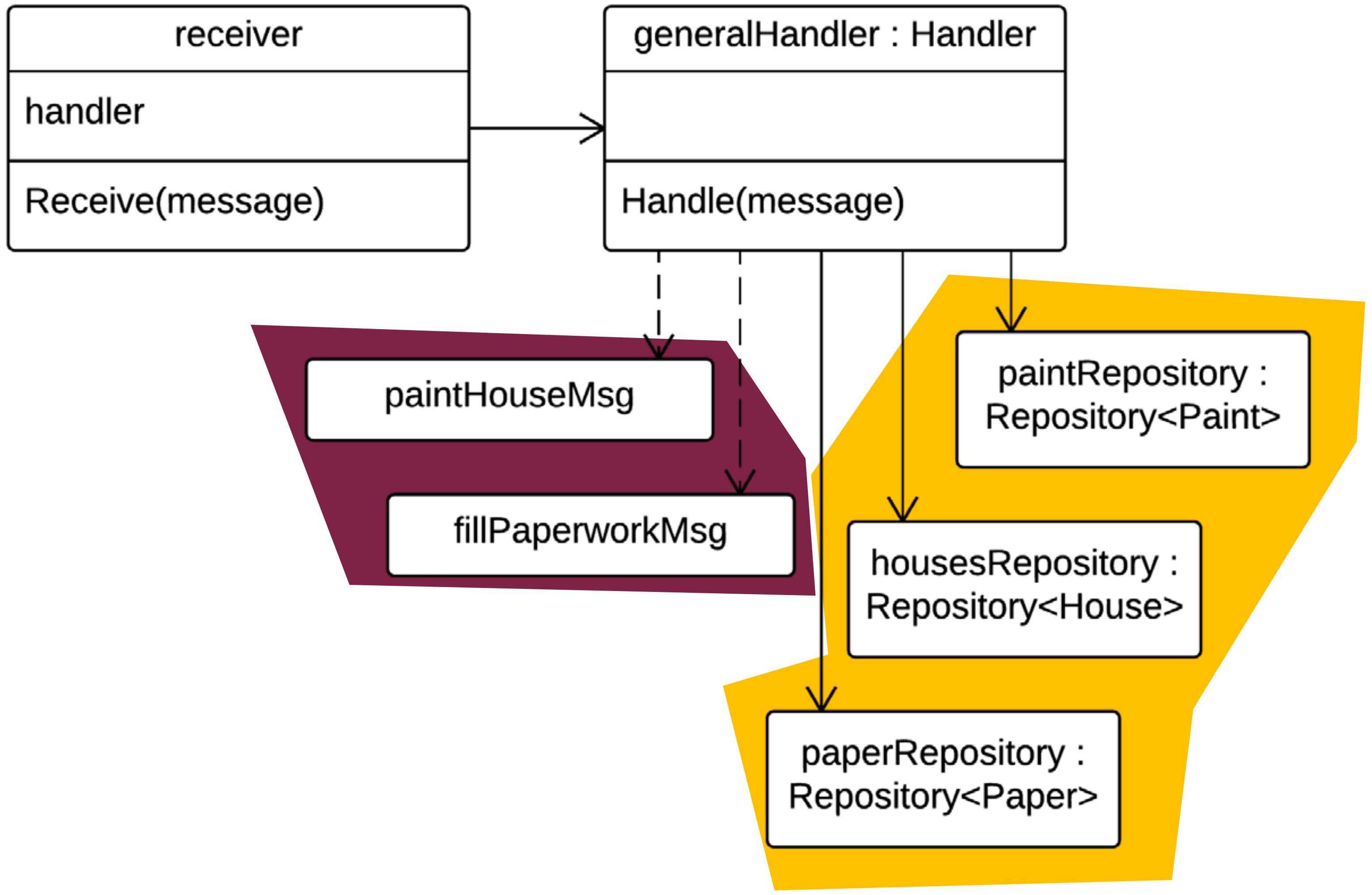
Procedural implementation

- Message is passed to a procedure as an argument
- Procedure executes the message

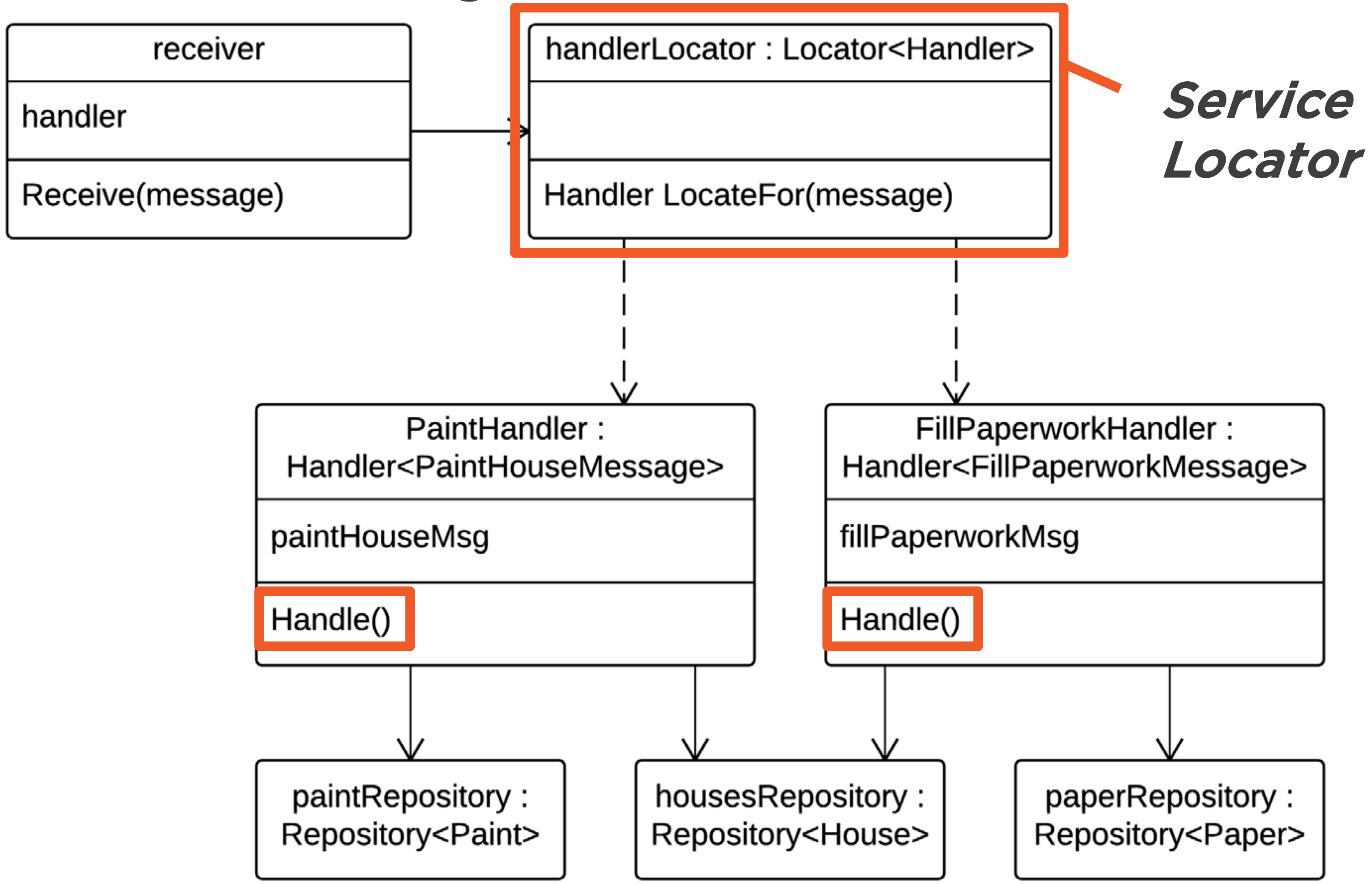
Adding Service Locator



Adding Service Locator

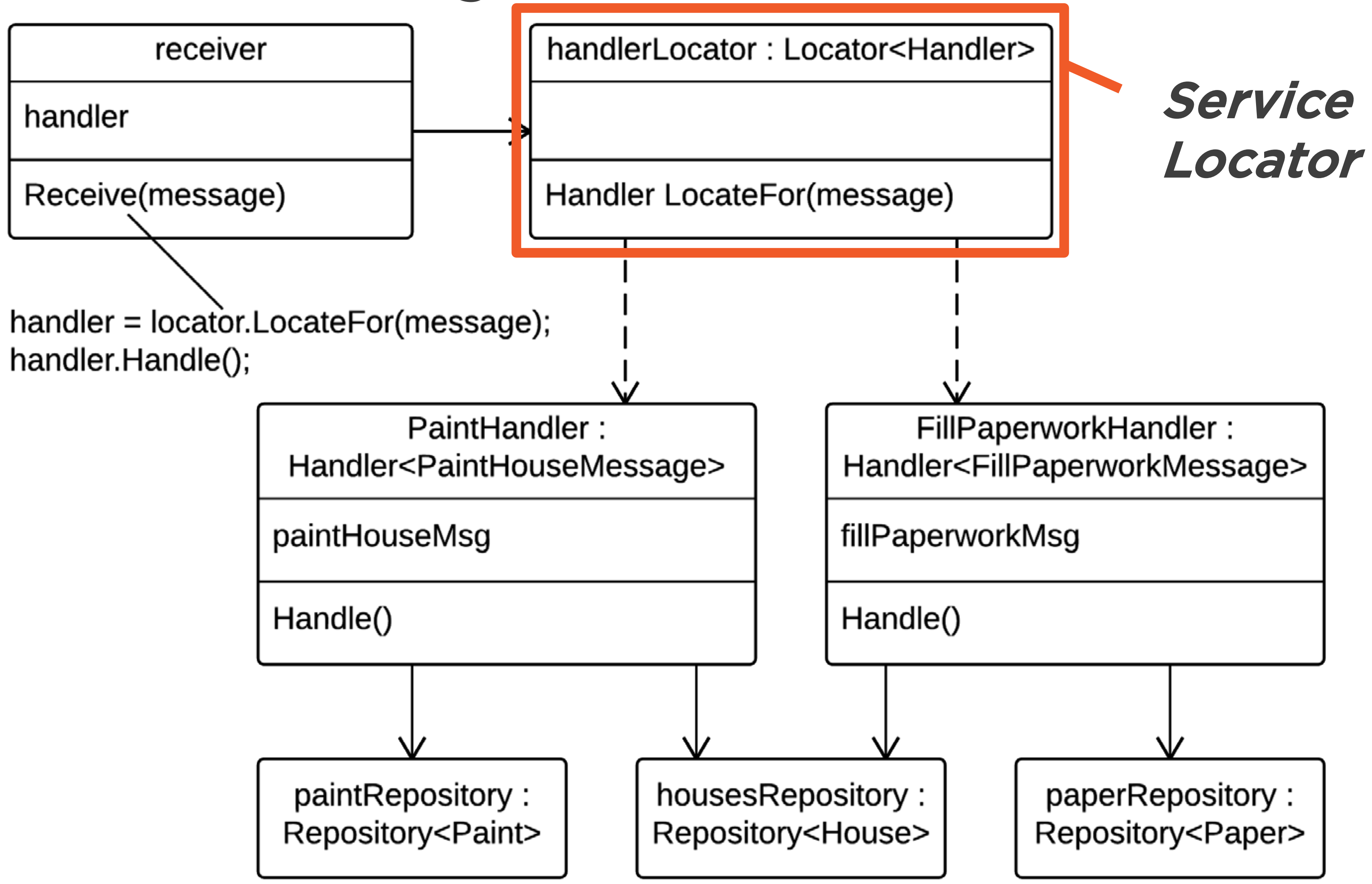

receiver
handler
Receive(message)
generalHandler : Handler
Handle(message)
paintHouseMsg
fillPaperworkMsg
paintRepository : Repository<Paint>
housesRepository : Repository<House>
paperRepository : Repository<Paper>

Adding Service Locator



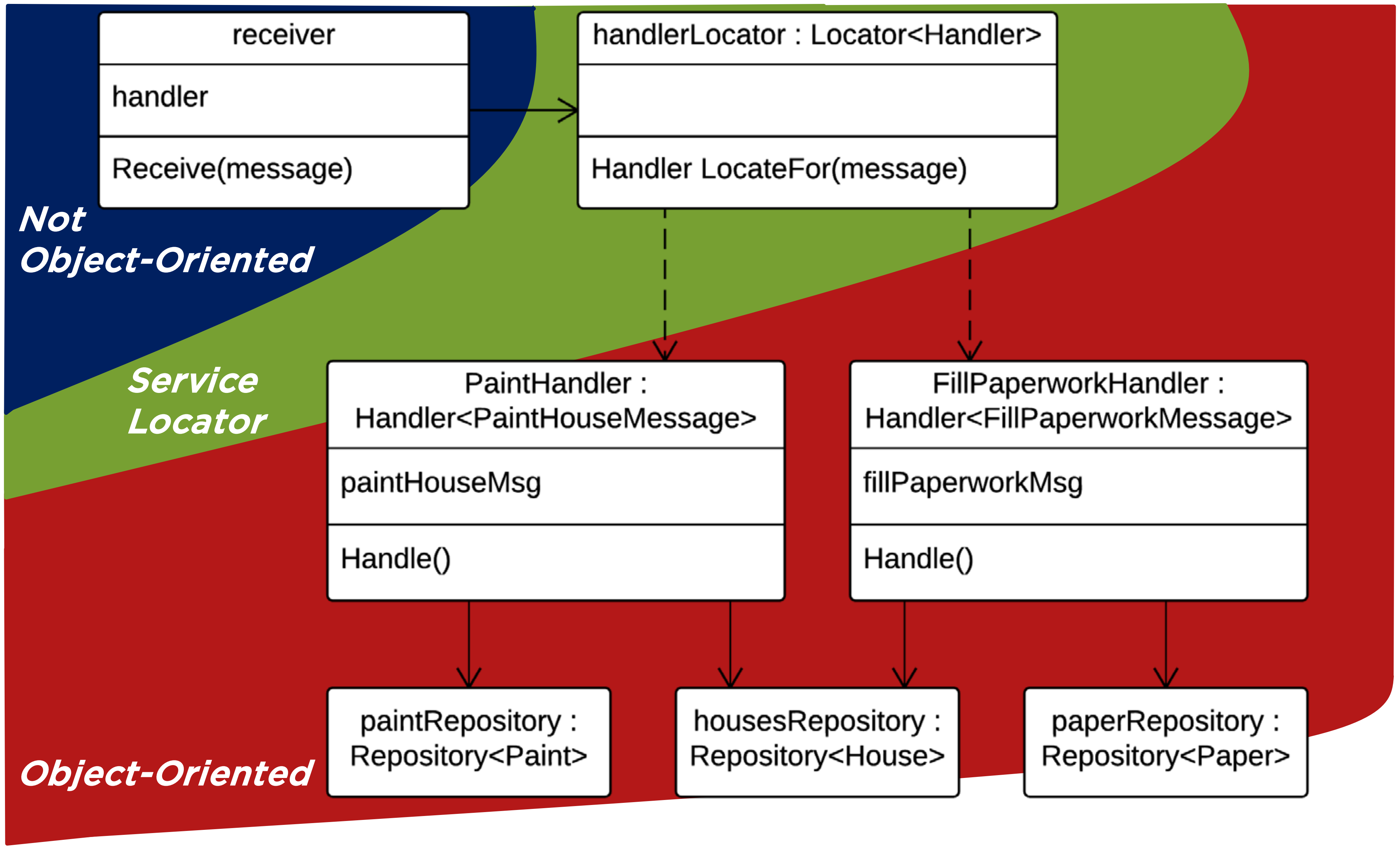
receiver
handler
Receive(message)
handlerLocator : Locator<Handler>
Handler LocateFor(message)
Service Locator
PaintHandler : Handler<PaintHouseMessage>
paintHouseMsg
Handle()
FillPaperworkHandler : Handler<FillPaperworkMessage>
fillPaperworkMsg
Handle()
paintRepository : Repository<Paint>
housesRepository : Repository<House>
paperRepository : Repository<Paper>

Adding Service Locator



receiver
handler
Receive(message)
handler = locator.LocateFor(message);
handler.Handle();
handlerLocator : Locator<Handler>
Handler LocateFor(message)
Service Locator
PaintHandler : Handler<PaintHouseMessage>
paintHouseMsg
Handle()
FillPaperworkHandler : Handler<FillPaperworkMessage>
fillPaperworkMsg
Handle()
paintRepository : Repository<Paint>
housesRepository : Repository<House>
paperRepository : Repository<Paper>

Adding Service Locator


receiver
handler
Receive(message)
handlerLocator : Locator<Handler>
Handler LocateFor(message)
Not Object-Oriented
Service Locator
PaintHandler : Handler<PaintHouseMessage>
paintHouseMsg
Handle()
FillPaperworkHandler : Handler<FillPaperworkMessage>
fillPaperworkMsg
Handle()
Object-Oriented
paintRepository : Repository<Paint>
housesRepository : Repository<House>
paperRepository : Repository<Paper>

Summary

We use Service Locator quite often

- DateTime structure,
- Static logger class…

Problems caused by Service Locator

- Hard to test
- Hard to vary implementation

Design issues regarding Service Locator

- Hides the client’s real dependencies
- Client cannot use different service than the one of the Service Locator

Summary

Service Locator vs. Dependency Injection

- Only if dependencies are injected during initialization
- Using dependency injection later equals using Service Locator

Legitimate Service Locator

- Mapping network messages
- Mapping domain model to UI elements

Where is the Service Locator useful?

- At the Object-Oriented to non-Object-Oriented code boundary

Coming next:
Guard Clauses and If-Then-Throw Pattern