DANCING WITH LEGACY CODE

FONG

ABOUT ME

- **Fong** (Syuan) Liou
- Backend Engineer
- Works at an EC platform startup
- NodeJS, DDD, GraphQL

GITHUB REPO

- https://github.com/ddd-tw/2020-legacycode-studygroup
- 資源存放
- 每次的 ppt, pdf, code
- 活動紀錄
- 開 Issues (?

FACEBOOK RESOURCES

- Fan page: Domain Driven Design Taiwan
- Group: Domain Driven Design(DDD Taiwan)

TABLE OF CONTENTS

CH1

CHANGING SOFTWARE

Photo by Agustin Uscanga on Unsplash

四個起因

1. 添加特性
2. 修正 Bug

“**Behavior** is the most important thing about software. It is what users depend on.”

四個起因 (CONTD.)

1. 添加特性
2. 修正 Bug
3. 改善設計 (重構)
   for software structure to improve maintainability
4. 最佳化資源使用
   for time & memory

POP QUIZ: STRUCTURE, FUNC, RESOURCE CHANGES

	添加特性	修正 Bug	重構	最佳化
Structure				
Func				
Resource				

POP QUIZ: STRUCTURE CHANGES

	添加特性	修正 Bug	重構	最佳化
Structure	V	V	V	
Func				
Resource				

POP QUIZ: FUNC CHANGES

	添加特性	修正 Bug	重構	最佳化
Structure	V	V	V	
Func	V	V		
Resource				

POP QUIZ: RESOURCE CHANGES

	添加特性	修正 Bug	重構	最佳化
Structure	V	V	V	
Func	V	V		
Resource				V

SUM UP

	添加特性	修正 Bug	重構	最佳化
Structure	V	V	V	
New Func	V			
Func		V		
Resource				V

三點不動一點動

- 前天我朋友教的

Photo by yns plt on Unsplash

保持既有行為不變是最具挑戰的任務之一

既有行為　　新行為

RISKY CHANGE (3Q)

1. **What** changes do we have to make?
2. How will we know that we've done them **correctly**?
3. How will we know that we **haven't broken** anything?

保守與避免修改的方式

- 問題不可避免
- 對架構越來越不熟
- 對技術生疏
- 恐懼心理
- 延伸閱讀：反脆弱心理

Image by Gerhard Gellinger from Pixabay

CH 2.

WORKING WITH FEEDBACK

Photo by Joel Fulgencio on Unsplash

重點

- Edit & Pray vs. Cover & Modify
- 回饋的**週期越短越好** (1 晚 vs. 1 min vs. 10 secs)
- 透過單元測試縮短回饋週期

UNIT TEST

- 針對一個**單獨軟體組件 (component)** 的一組獨立測試
- 軟體組件可以是一個**函數 (function)** 或是**類別 (class)**
- 或是一個軟體的**行為作為單位**（跨 class, function, Solitary or Sociable)。

- 比起大型測試：
  - 更精準的定位
  - 更短的執行時間
  - 更安全的覆蓋

I think that the term "unit testing" is appropriate because these tests **are tests of the behavior of a single unit.**

–MARTIN FOWLER

UNIT TEST 原則

- Fast, less then 1s
- Deterministic 確定性的
- Predictive
- Sensitive to behavior changes, insensitive to structure changes
- Cheap to write & read & change
  ~~you don't get paid for tests~~
- Isolation from other tests

TESTING PYRAMID

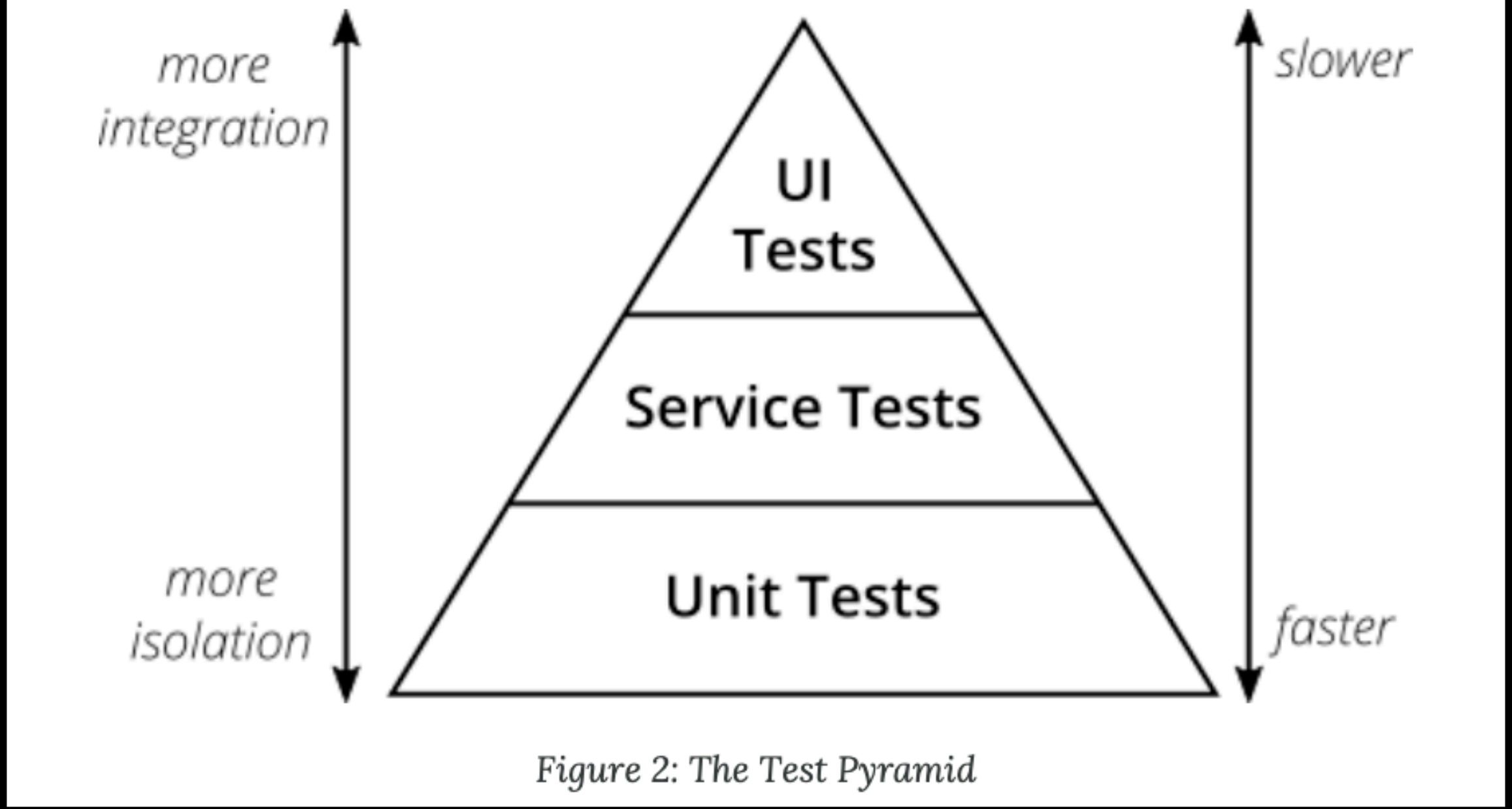

Figure 2: The Test Pyramid

https://martinfowler.com/articles/practical-test-pyramid.html#IntegrationTests

哪些是 UNIT TEST？

1. Function **add** (a, b) { return a + b }
2. Function **save**(obj) { **dao**.insert('table', obj) }
3. Function **writeFile**(path, data) { fs.write(path, data) }
4. Function **newCanvas** () {
   config = readFromConfig()
   newOne = new Canvas()
   newOne.setConfig(config)
   return newOne
   }

修改的演算法

1. **確定變動點**
   程式理解力。可透過畫草圖、架構師溝通提升。

2. **找出測試點**
   找出影響範圍、依賴性關係

3. **解依賴**
   方便在測試中實例化與使用物件

4. **編寫測試**
   為 legacy code 編寫特徵測試

5. **修改、重構**
   prefer TDD

1. 確認變動點

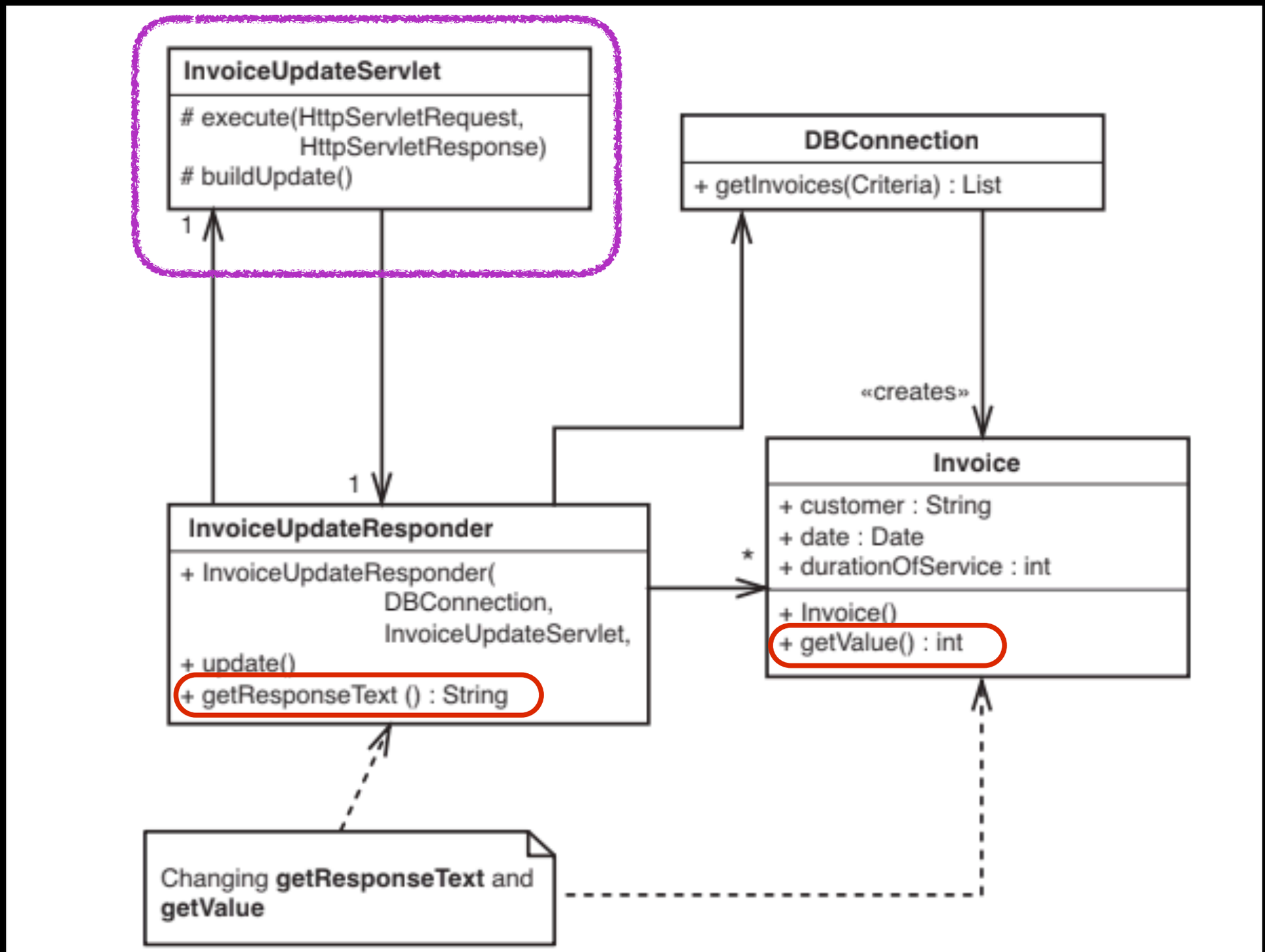

Figure 2.1 *Invoice update classes.*

2. 找出測試點：這兩點也是影響結構的最底層，對他們做測試可以感測可能造成的一切影響。

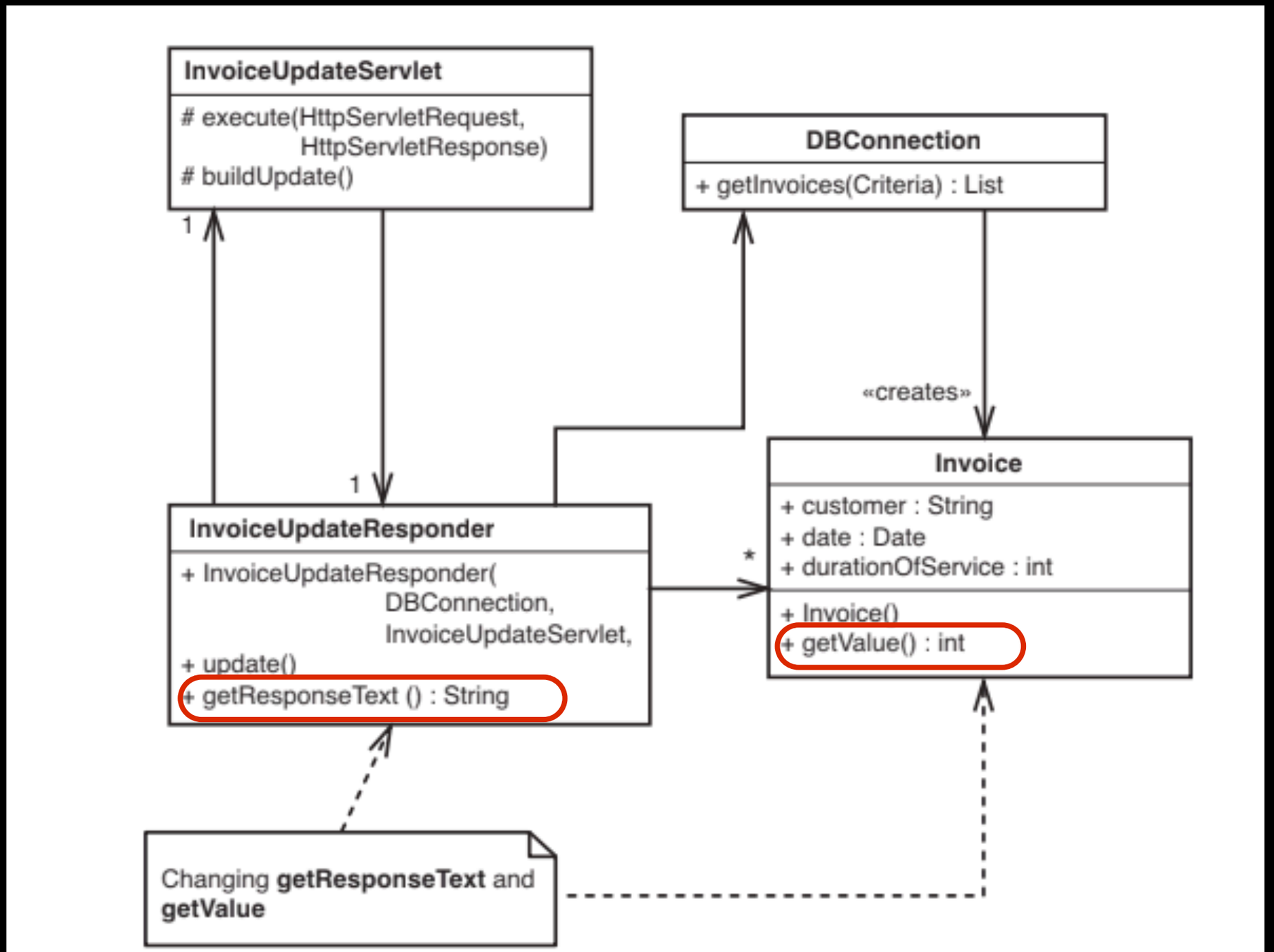

Figure 2.1 *Invoice update classes.*

3. 解依賴

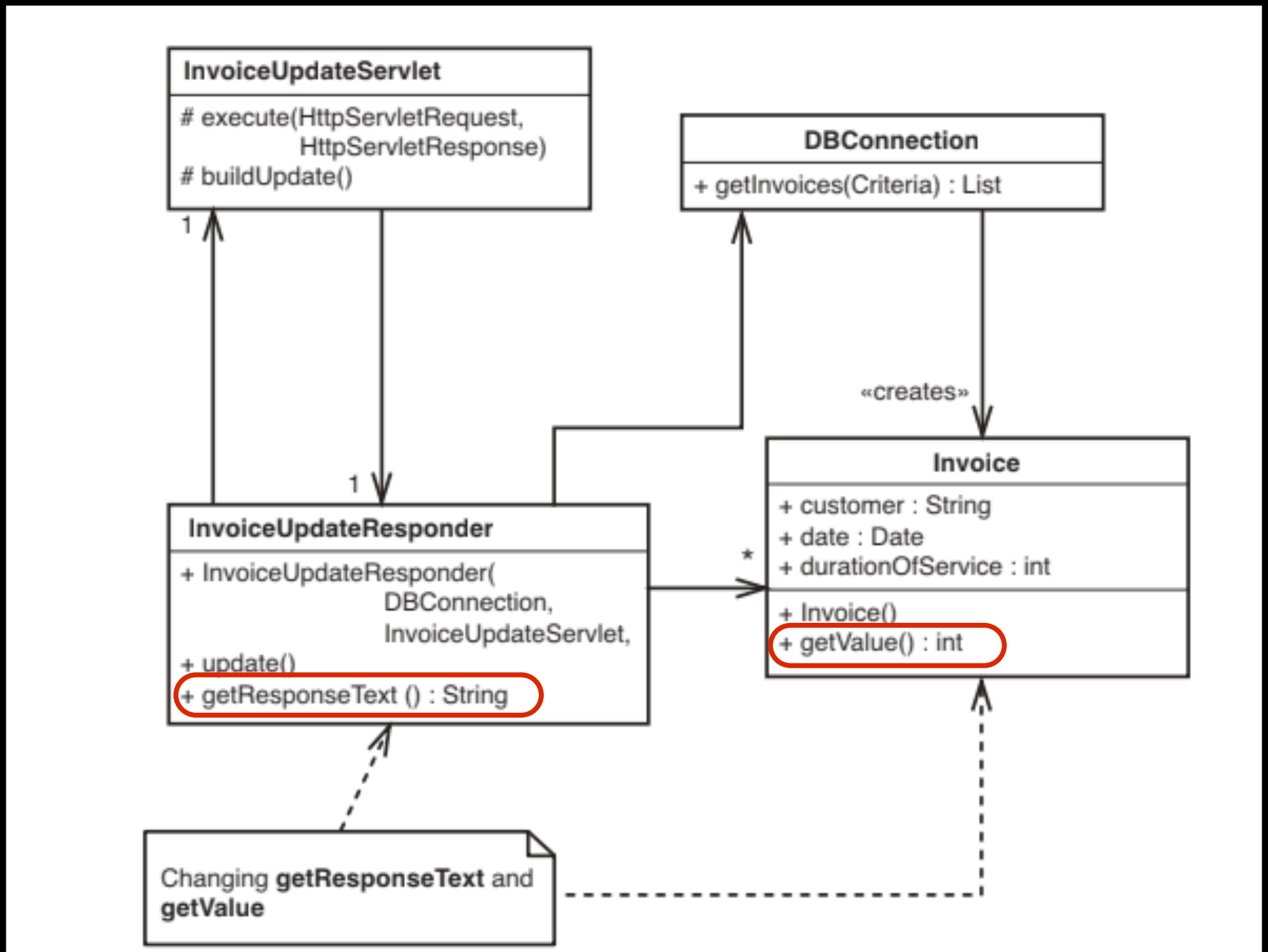

Figure 2.1 *Invoice update classes.*

3. 解依賴

1. **InvoiceUpdateServlet** 可以僅傳值進去
2. **DBConnection** 可以依賴於 interface

當一個類別直接依賴某些難以在測試中使用的東西時，他就會難以修改和處理。

解除依賴性以便使修改變得更容易

–PAGE 19

3．解依賴(CONTD.)

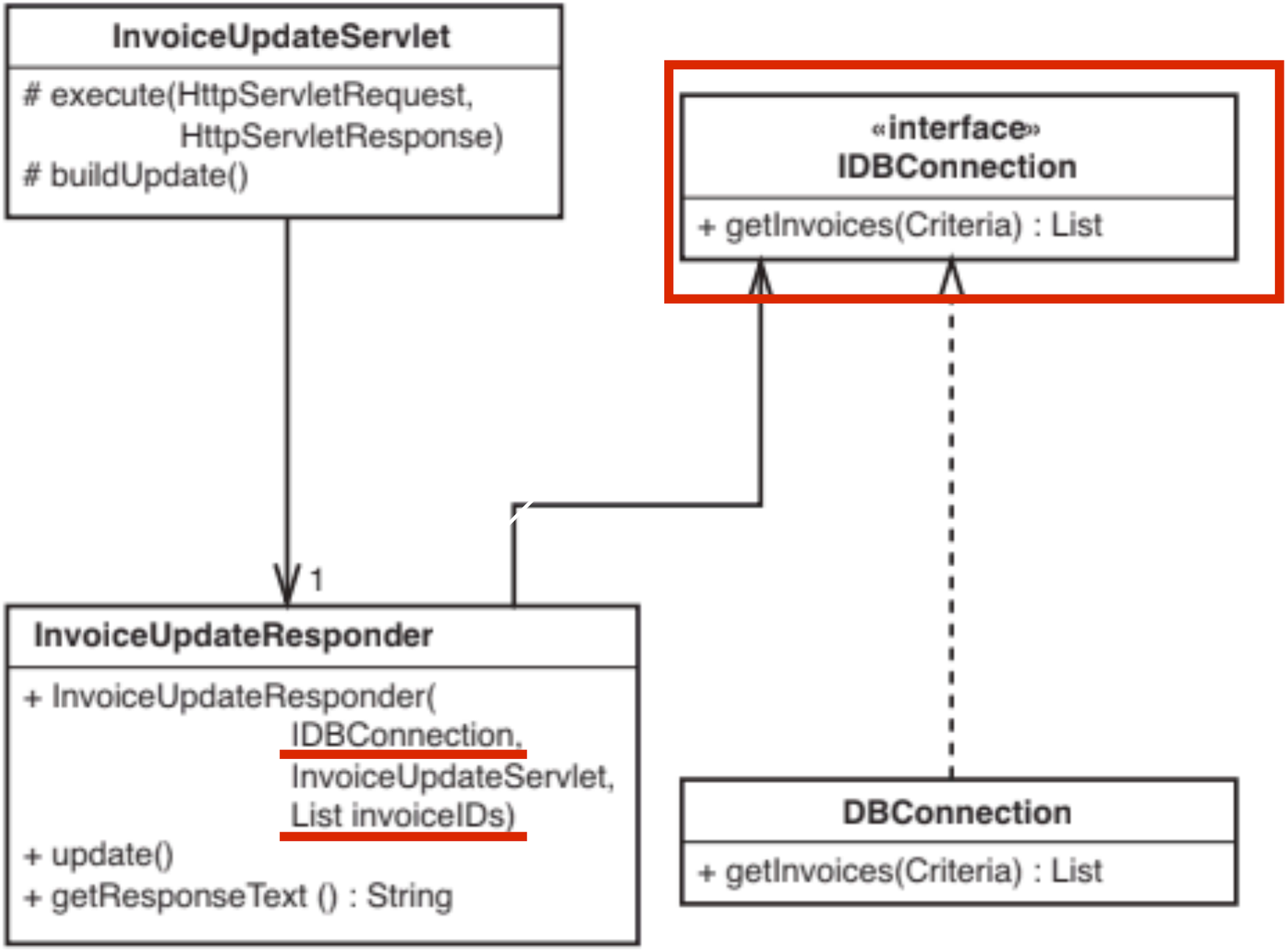

Figure 2.2 *Invoice update classes with dependencies broken.*

4. 編寫測試

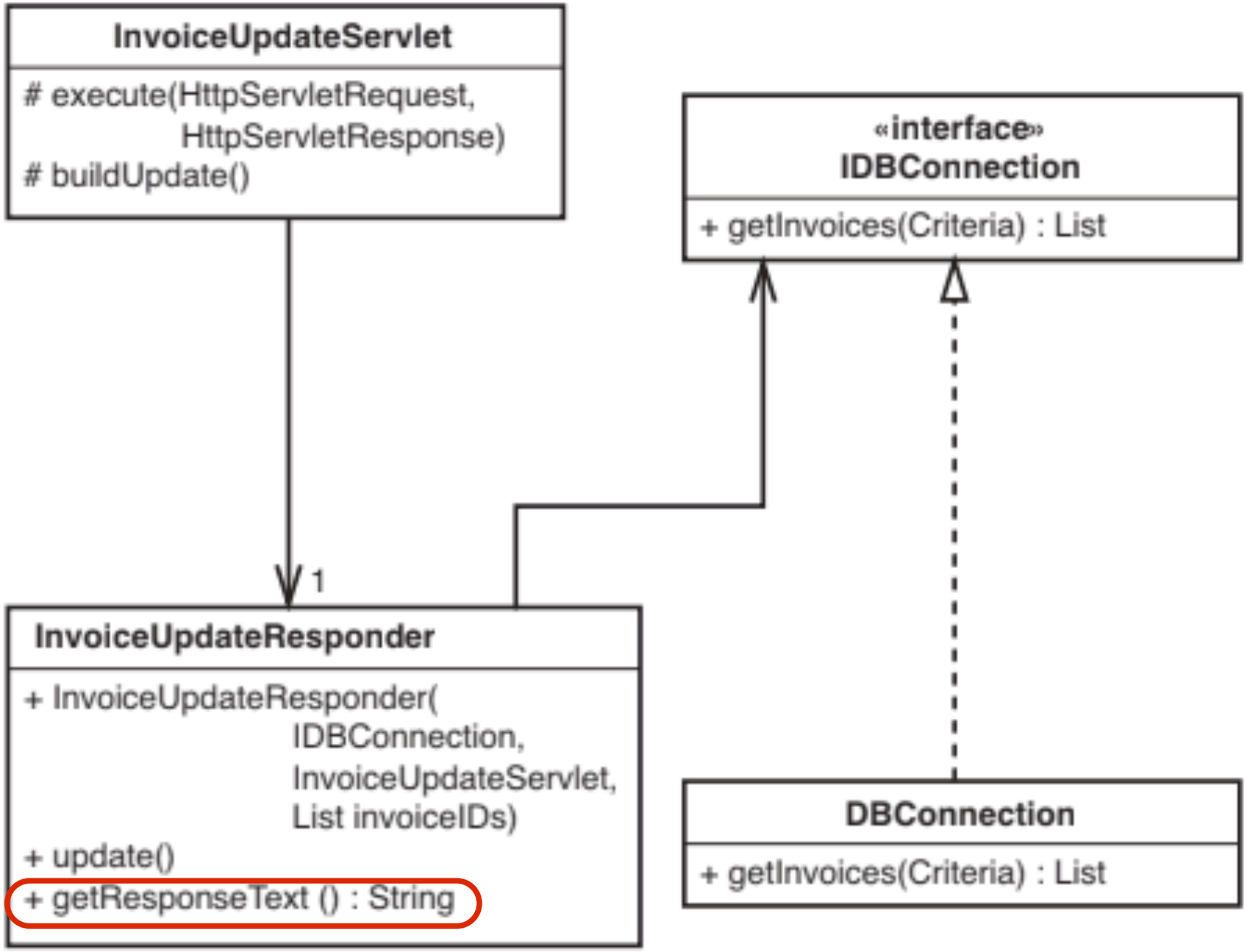

Figure 2.2 *Invoice update classes with dependencies broken.*

CH 3

SENSING AND SEPARATION

汽車意外模擬測試...


GLOBAL NCAP
LATIN NCAP
ADAC
FO2214FPA1


VR
CRASH
SIMU
A CRA
YOU HAVE

Example: **NetworkBridge**

如果每次測試還要感測封包數量等等...測試成本增加

```
public class NetworkBridge
{
    public NetworkBridge(EndPoint [] endpoints) {
        ...
    }

    public void formRouting(String sourceID, String destID) {
        ...
    }
    ...
}
```

感測與分離

- 感測：當我們無法存取到程式碼計算出的值時，我們就必須**解依賴**來**「感測 (取得)」**這些值 ➡ 偵測假人數值

- 分離：當我們無法將一部分程式碼，放入測試控制工具去執行時，就需要透過**解依賴**將這塊程式碼**「分離」**出來。 ➡ 使用假人

偽裝成合作者 - 偽物件

偽裝成合作者 - **分離**職責

- **Sale.scan(barcode: String)** 可以在收銀機上顯示被掃瞄商品之名稱與價格
- 把 Sale 中對收銀機 API 的呼叫取出來成 **ArtR56Display**

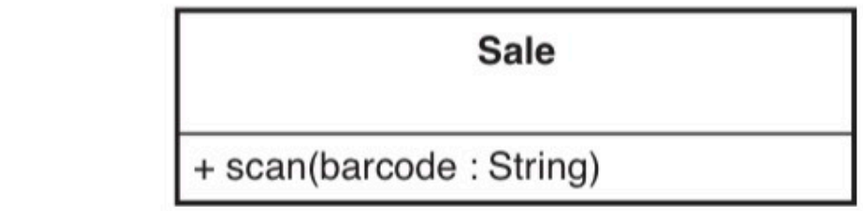

Figure 3.1 *Sale.*

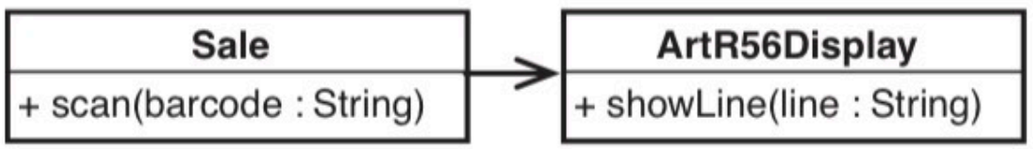

Figure 3.2 *Sale communicating with a display class.*

Image by James Bong from Pixabay

偽裝成合作者 - 用介面分離依賴

- 建立 Display 介面給 Sale 在建構子傳入使用
- Sale.scan 時呼叫 Display.showLine

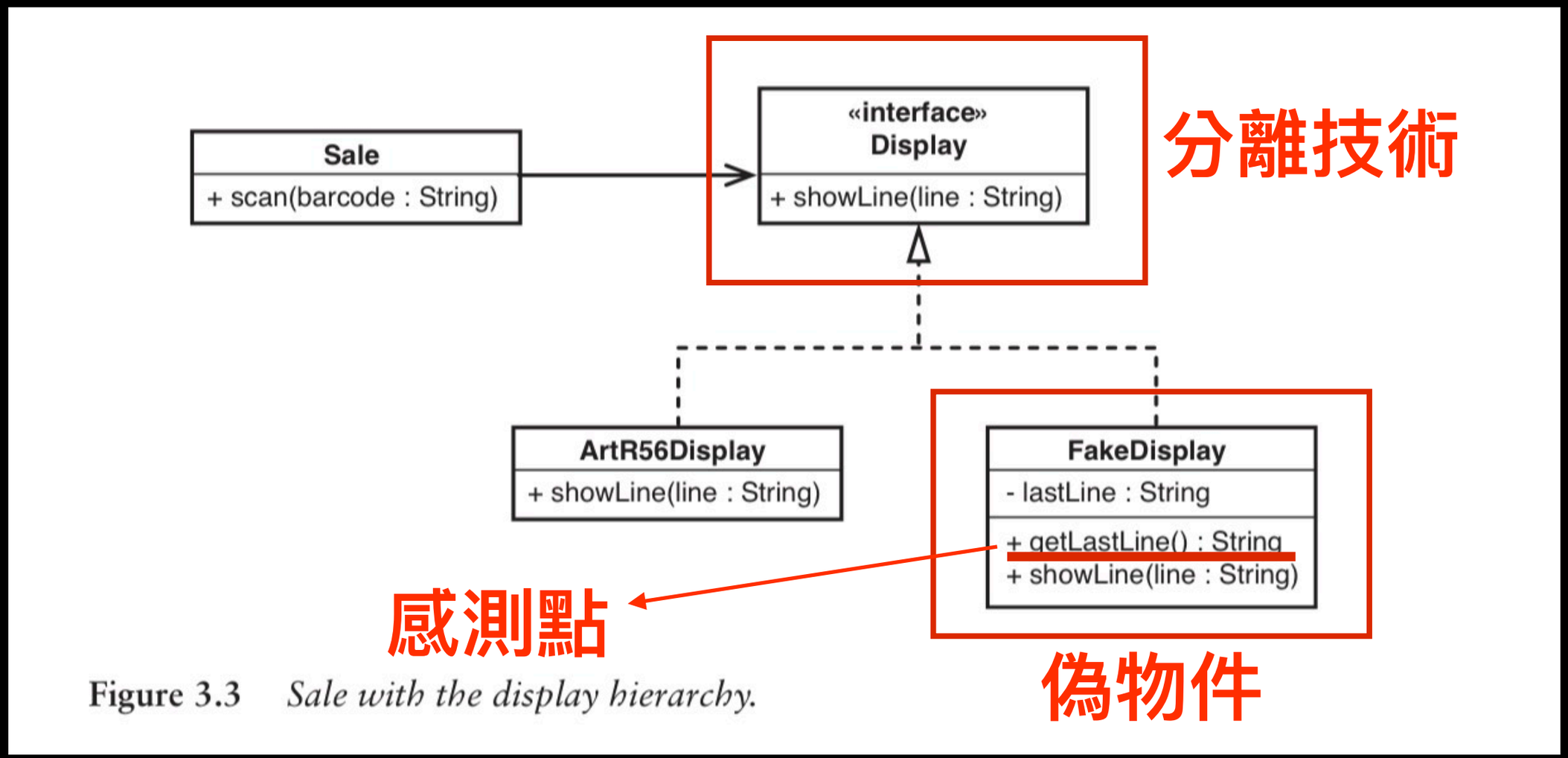

Figure 3.3 *Sale with the display hierarchy.*

```
public interface Display
{
    void showLine(String line);
}
```

```
public class Sale
{
    private Display display;

    public Sale(Display display) {
        this.display = display;
    }

    public void scan(String barcode) {
        ...
        String itemLine = item.name()
                + " " + item.price().asDisplayText();
        display.showLine(itemLine);
        ...
    }
}
```

```
import junit.framework.*;

public class SaleTest extends TestCase
{
    public void testDisplayAnItem() {
        FakeDisplay display = new FakeDisplay();
        Sale sale = new Sale(display);

        sale.scan("1");
        assertEquals("Milk $3.99", display.getLastLine()
    }
}
```

```
public class FakeDisplay implements Display
{
    private String lastLine = "";

    public void showLine(String line) {
        lastLine = line;
    }

    public String getLastLine() {
        return lastLine;
    }
}
```

1 實際使用

2 測試感測用

“偽物件支援真證的測試：在編寫測試時，我們得採用**分而治之（逐一擊破）**的方案。不但輕量、好讀、定位性也強。”

–P30

仿物件（MOCK OBJECT）

- 如果你要編寫許多偽物件的話，可考慮使用仿物件。
- 經驗：大多數不太建議，會把程式碼與行為耦合太深，不利後續維護。除非是要對偽物件進行一連串的驗證，否則優點不大。

```
import junit.framework.*;

public class SaleTest extends TestCase
{
    public void testDisplayAnItem() {
        MockDisplay display = new MockDisplay();
        display.setExpectation("showLine", "Milk $3.99");
        Sale sale = new Sale(display);
        sale.scan("1");
        display.verify();
    }
}
```

DDD EXAMPLE REPOSITORY IN USE CASE

- Make Repository an interface

```java
import tw.fong.clean.usecase.User.add.CreateUserUseCase;

public class CreateUserUseCase {

    private UserRepository repository;

    public CreateUserUseCase(UserRepository repository){
        this.repository = repository;
    }

    @Override
    public void execute(CreateUserInput input, CreateUserOutput output)
        User User = new User(input.getUserName());
        repository.save(User);

        output.setUserName(User.getName());
    }

    public static CreateUserInput createInput(){
        return new CreateUserInputImpl();
    }

    private static class CreateUserInputImpl implements CreateUserInput
        private String UserName;

        @Override
        public void setUserName(String UserName) {
            this.UserName = UserName;
        }

        @Override
        public String getUserName() {
```

DDD EXAMPLE REPOSITORY INTERFACE

- A repository for an aggregate
- Live besides its aggregate

```java
package com.saasovation.identityaccess.domain.model.ident

import java.util.Collection;

public interface UserRepository {

    public void add(User aUser);

    public Collection<User> allSimilarlyNamedUsers(
            TenantId aTenantId,
            String aFirstNamePrefix,
            String aLastNamePrefix);

    public void remove(User aUser);

    public User userFromAuthenticCredentials(
            TenantId aTenantId,
            String aUsername,
            String anEncryptedPassword);

    public User userWithUsername(
            TenantId aTenantId,
            String aUsername);
}
```

DDD EXAMPLE REPOSITORY REAL ONE

- **HibernateUserRepository**
- Use Hibernate ORM

```java
import com.saasovation.common.port.adapter.persistence.hibernate.AbstractHib
import com.saasovation.identityaccess.domain.model.identity.TenantId;
import com.saasovation.identityaccess.domain.model.identity.User;
import com.saasovation.identityaccess.domain.model.identity.UserRepository;

public class HibernateUserRepository
        extends AbstractHibernateSession
        implements UserRepository {

    public HibernateUserRepository() {
        super();
    }

    @Override
    public void add(User aUser) {
        try {
            this.session().saveOrUpdate(aUser);
        } catch (ConstraintViolationException e) {
            throw new IllegalStateException("User is not unique.", e);
        }
    }

    @Override
    @SuppressWarnings("unchecked")
    public Collection<User> allSimilarlyNamedUsers( ⋯
    }

    @Override
    public void remove(User aUser) {
        this.session().delete(aUser);
    }

    @Override
    public User userFromAuthenticCredentials(
            TenantId aTenantId,
            String aUsername,
```

DDD EXAMPLE REPOSITORY MOCK

- InMemoryUserRepository
- In memory
- Map<String, User> repository

```java
import com.saasovation.common.persistence.CleanableStore;
import com.saasovation.identityaccess.domain.model.identity.Full
import com.saasovation.identityaccess.domain.model.identity.Tena
import com.saasovation.identityaccess.domain.model.identity.User
import com.saasovation.identityaccess.domain.model.identity.User

public class InMemoryUserRepository implements UserRepository, C

    private Map<String,User> repository;

    public InMemoryUserRepository() {
        super();
        this.repository = new HashMap<String,User>();
    }

    @Override
    public void add(User aUser) {
        String key = this.keyOf(aUser);

        if (this.repository().containsKey(key)) {
            throw new IllegalStateException("Duplicate key.");
        }

        this.repository().put(key, aUser);
    }
```

DDD EXAMPLE REPOSITORY TEST

- Use InMemoryOne (enabling point)

```java
import static org.junit.Assert.*;

public class CreateUserTest {

    @Before
    public void setUp(){
    }

    @Test
    public void create_user() {
        CreateUserInput input = CreateUserInputImpl.createInput();
        CreateUserOutput output = new CreateUserPresenter();
        input.setUserName("Eric Evans");

        InMemoryUserRepository repository = new InMemoryUserRepository();
        CreateUserUseCase usecase = new CreateUserUseCase(repository);

        usecase.execute(input, output);

        assertEquals(1, repository.findAll().size());
    }

}
```

CH 4

THE SEAM MODEL

如何寫出易於測試的程式碼？

1. 邊開發編寫測試。
2. 前期多花時間，試著將易測試性納入整體設計考量。

SEAM（接縫）& ENABLING POINT（致能點）

- Seam：程式中一個特殊的點，在這些點上你**無需做任何修改**就可以達到**變動程式**的目的。
- Enabling Point：每個 Seam 都有一個 enabling point，在這個點上你可以**決定使用哪種行為**。

- 預處理接縫
- 連接期接縫
- 物件接縫

EXAMPLE 1

想要在測試時不呼叫他避免外部互動。但又不能在正式環境漏掉他

```
bool CAsyncSslRec::Init()
{

  if (m_bSslInitialized) {
    return true;
  }
  m_smutex.Unlock();
  m_nSslRefCount++;
  m_bSslInitialized = true;
  FreeLibrary(m_hSslDll1);
  m_hSslDll1=0;
  FreeLibrary(m_hSslDll2);
  m_hSslDll2=0;

  if (!m_bFailureSent) {
    m_bFailureSent=TRUE;
    PostReceiveError(SOCKETCALLBACK, SSL_FAILURE);
  }

  CreateLibrary(m_hSslDll1,"syncesel1.dll");
  CreateLibrary(m_hSslDll2,"syncesel2.dll");

  m_hSslDll1->Init();
  m_hSslDll2->Init();

  return true;
}
```

Seam

1. 用添加的 method 包覆

```
class CAsyncSslRec
{
    ...
    virtual void PostReceiveError(UINT type, UINT errorcode);
    ...
};
```

2. 實作：把任務轉發出去

```
void CAsyncSslRec::PostReceiveError(UINT type, UINT errorcode)
{
    ::PostReceiveError(type, errorcode);
}
```

3. 子類別化 CAsayncSslRec 並 override PostReceiveError

```
class TestingAsyncSslRec : public CAsyncSslRec
{
    virtual void PostReceiveError(UINT type, UINT errorcode)
    {
    }
};
```

```
class CAsyncSslRec
{
    ...
    virtual void PostReceiveError(UINT type, UINT errorcode);
    ...
};
```

Enabling Point

```
bool CAsyncSslRec::Init()
{

  if (m_bSslInitialized) {
    return true;
  }
  m_smutex.Unlock();
  m_nSslRefCount++;
  m_bSslInitialized = true;
  FreeLibrary(m_hSslDll1);
  m_hSslDll1=0;
  FreeLibrary(m_hSslDll2);
  m_hSslDll2=0;

  if (!m_bFailureSent) {
    m_bFailureSent=TRUE;
    PostReceiveError(SOCKETCALLBACK, SSL_FAILURE);
  }

  CreateLibrary(m_hSslDll1,"syncesel1.dll");
  CreateLibrary(m_hSslDll2,"syncesel2.dll");

  m_hSslDll1->Init();
  m_hSslDll2->Init();

  return true;
}
```

Seam

預處理接縫

- C/C++ 預處理器就像是一個文本替換工具，指示編譯器實際編譯之前需要做預處理。
- 預處理命令須以 '#' 開頭。

```
#include <DFHLItem.h>
#include <DHLSRecord.h>

extern int db_update(int, struct DFHLItem *);

void account_update(
    int account_no, struct DHLSRecord *record, int activated)
{
    if (activated) {
        if (record->dateStamped && record->quantity > MAX_ITEMS) {
            db_update(account_no, record->item);
        } else {
            db_update(account_no, record->backup_item);
        }
    }
    db_update(MASTER_ACCOUNT, record->item);
}
```

預處理

Seam

加入標頭檔 localdefs.h

```
#include <DFHLItem.h>
#include <DHLSRecord.h>

extern int db_update(int, struct DFHLItem *);

#include "localdefs.h"

void account_update(
    int account_no, struct DHLSRecord *record, int activated)
{
    if (activated) {
        if (record->dateStamped && record->quantity > MAX_ITEMS) {
            db_update(account_no, record->item);
        } else {
            db_update(account_no, record->backup_item);
        }
    }
    db_update(MASTER_ACCOUNT, record->item);
}
```

localdefs.h 内容

```
#ifdef TESTING
...
struct DFHLItem *last_item = NULL;
int last_account_no = -1;

#define db_update(account_no,item)\
    {last_item = (item); last_account_no = (account_no);}
...
#endif
```

Enabling Point

連接期接縫

- JAVA: classpath
- NodeJS: Env files for testing, dev, …
- K8s configmap files for different environments
- …

物件接縫

- **不事先指定**哪一個類別會被呼叫：**cell.Recalculate( )**;
- 如果可以在**不用修改**這行以及周圍程式碼的情況下，就讓它呼叫另一個 Recalculate method，就是接縫

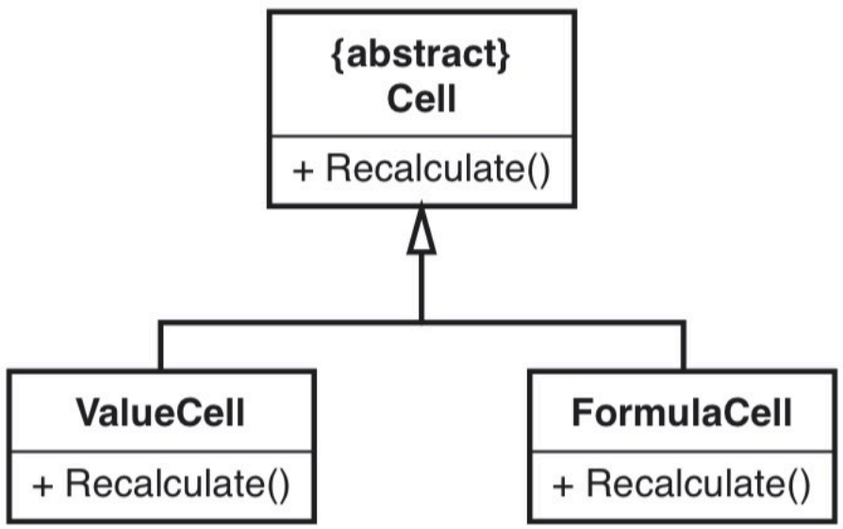

Figure 4.1 *Cell hierarchy.*

這邊是一個接縫點嗎？

```
public class CustomSpreadsheet extends Spreadsheet
{
    public Spreadsheet buildMartSheet() {
        ...
        Cell cell = new FormulaCell(this, "A1", "=A2+A3");
        ...
        cell.Recalculate();
        ...
    }
    ...
}
```

這邊是一個接縫點嗎？

```
public class CustomSpreadsheet extends Spreadsheet
{
    public Spreadsheet buildMarketSheet() {
        ...
        Cell cell = new FormulaCell(this, "A1", "=A2+A3");
        cell.Recalculate();
        ...
    }
    ...
}
```

NO

已經指定好類別，無法控制呼叫不同的 Recalcuate

那 enabling point 在哪裏？

```
ublic class CustomSpreadsheet extends Spreadsheet

public Spreadsheet buildMartSheet(Cell cell) {
    ...
    cell.Recalculate();
    ...
}
...
```

Seam

透過 buildMartSheet 參數列表可以決定要用哪一個類型

```
ublic class CustomSpreadsheet extends Spreadsheet

public Spreadsheet buildMartSheet(Cell cell) {
    ...
    cell.Recalculate();
    ...
}
...
```

Enabling Point

Seam

再一案例：如何在這邊新增 SEAM

```
public class CustomSpreadsheet extends Spreadsheet
{
    public Spreadsheet buildMartSheet(Cell cell) {
        ...
        Recalculate(cell);
        ...
    }

    private static void Recalculate(Cell cell) {
        ...
    }

    ...
}
```

答：改成可覆寫的 METHOD

```
public class CustomSpreadsheet extends Spreadsheet
{
    public Spreadsheet buildMartSheet(Cell cell) {
        ...
        Recalculate(cell);
        ...
    }

    protected void Recalculate(Cell cell) {
        ...
    }

    ...
}

public class TestingCustomSpreadsheet extends CustomSpreadsheet {
    protected void Recalculate(Cell cell) {
        ...
    }
}
```

CH 5

TOOLS

Photo by Haupes Co. on Unsplash

大綱

- 自動化重構工具（現代 IDE)
    - 更改變數名稱
- 仿物件 (Mock Object)
- 單元測試控制工具
    - Junit
    - CppUnitLite
- 一般測試控制工具
    - 整合測試框架 FIT
    - Fitness

MORE

EXERCISES

Photo by dylan nolte on Unsplash

TRIP SERVICE KATA

- https://github.com/FongX777/trip-service-kata
- JAVA
- C#
- JS
- TS
- PHP

RULES

- Cannot change any existing code if not covered by tests. ~~好拉不用那麼嚴格～有練習到就好~~
- The only exception is automated refactors (via IDE)
- The final code should pass all tests

TRIP SERVICE KATA (1)

- 為原本的程式碼新增測試
- **TripService.getTripsByUser(user)**：在登入的情況下，如果該 User 是你的朋友，那就取得他的旅遊資料，否則回傳空資料。
- Seam **UserSession.getLoggeredUser()**, **TripDAO.findTripsByUser(user)** 並測試：
  - should Throw Exception When User Is Not Logged In
  - should Not Return Trips When Logged User Is Not A Friend
  - should Return Trips When Logged User Is A Friend
- Refactor **TripService.getTripsByUser(user)** 。

TRIP SERVICE KATA (2)

- 把 Trip 從 User 分離後，Fake/Mock **TripDAO** (偽物件) 來感測 DAO 裡面的 Trip 資料。

TRIP SERVICE KATA (3)

- 使用分層原理，分成 domain, use case, repository 三層
- 將 **User**, **Trip** 獨立建模成 Aggregate
- 將 **getTripsByUser** 建成 use case
- 將 DAO 轉成 Repository 模式

RECAP

- 三點不動一點動：結構、舊功能、新功能、資源改變
- Cover & Modify

- 用偽物件模擬實際物件，並提供 method 供感測資料
- Seam: 程式中一個特殊的點，在這些點上你無需做任何修改就可以達到變動程式的目的。
- 解依賴是處理 Legacy Code 最困難的部分


THE END
ingresso gruppi