ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

07
Chapitre

Refonte de code (Refactoring)

1I2AC1 : Génie logiciel et Conception orientée objet

Régis Clouard, ENSICAEN - GREYC

« «Si débuguer c'est supprimer des bugs,
alors programmer ne peut être que de les ajouter. »
Edsger Dijkstra

Qu'est ce que la refonte de code ?

- **Définition**
  - La refonte de code est le processus qui consiste à changer le code d'un système logiciel de telle manière
    - qu'il **n'altère pas le comportement extérieur** du code
    - tout en **améliorant sa structure interne**

Pourquoi faire de la refonte de code ?

- Pour ne pas accumuler de **dette technique**
- Pour améliorer la logique interne du logiciel
- Pour rendre le système plus simple à comprendre et donc à maintenir, étendre et vérifier
  - Principe KISS : « Keept It Simple, Stupid. »
    - « La perfection n'est atteinte, non pas lorsqu'il n'y a plus rien à ajouter, mais lorsqu'il n'y a plus rien à enlever. »
      Antoine de Saint-Exupéry
- Pour aider à trouver les bugs (p. ex. les Schroedinbugs)

Quand faire de la refonte de code ?

1/ Lors du développement d'une fonctionnalité

- Induit une nouvelle façon de coder :
  1) Vite fait pas propre : *Quick and Dirty*
  2) Refonte du code : *Refactoring*

- Remarque : en Agilité l'ajout d'une fonctionnalité dans le code commun ne peut se faire que si son code est propre

Quand faire de la refonte de code ?

2/ Lors de la maintenance corrective

- Lors de la correction d'un bug en profiter pour améliorer le code
  - Application de la règle de boy-scouts : « *Laissez le campement (code) plus propre que vous l'avez trouvé en arrivant (en l'éditant)* »

Exemples de refonte de code

- Des indices d'une refonte nécessaire : "*bad smells in the code*"
  - - Duplication de code
  - - Longues méthodes
  - - Grandes classes
  - - Longue liste de paramètres
  - - Nécessité d'ajouter un commentaire
  - …

-> Sélection d'exemples dans le catalogue de Martin Fowler

- http://refactoring.com/catalog/index.html

Remplacer un nombre magique par une constante symbolique

- Vous avez un nombre littéral avec un sens particulier.

```
double potentialEnergy( double mass, double height ) {
  return mass * 9.81 * height;
}
```

```
static final double GRAVITATIONAL_CONSTANT = 9.81;
double potentialEnergy( double mass, double height ) {
  return mass * GRAVITATIONAL_CONSTANT * height;
}
```

Supprimer les doubles négations

- Vous avez une condition avec une double négation.

```
if (!item.isNotFound()) { }
```

```
if (item.isFound()) { }
```

Introduire une variable explicative

- Vous avez une expression de condition compliquée.

```
if (platform.indexOf("MAC") > -1
    && browser.indexOf("IE") > -1
    && wasInitialized() && resize > 0) {
       // do something
}
```

```
boolean isMacOs = platform.indexOf("MAC") > -1;
final boolean isIEBrowser = browser.indexOf("IE") > -1;
final boolean wasResized = resize > 0;
if (isMacOs && isIEBrowser && wasInitialized() && wasResized) {
      // do something
}
```

Préserver un objet entier

- Vous utilisez plusieurs valeurs d'un objet et passez ces valeurs comme paramètres d'une méthode.

```
int low = daysTempRange().getLow();
int high = daysTempRange().getHigh();
withinPlan = plan.withinRange(low, high);
```

```
withinPlan = plan.withinRange(daysTempRange());
```

Remplacer un code d'erreur par une exception

- Vous avez une méthode qui retourne un code spécial pour indiquer une erreur.

```
int withdraw( int amount ) {
   if (amount > _balance) {
     return -1;
    }
    _balance -= amount;
    return 0;
}
```

```
void withdraw( int amount ) throws BalanceException {
  if (amount > _balance) {
    throw new BalanceException();
  }
   _balance -= amount;
}
```

Introduire un objet paramètre

- Vous avez un groupe de paramètres qui vont naturellement ensemble.

```
amountInvoicedIn(start: Date, end: Date)
amountReceivedIn(start: Date, end: Date)
amountOverdueIn(start: Date, end: Date)
```

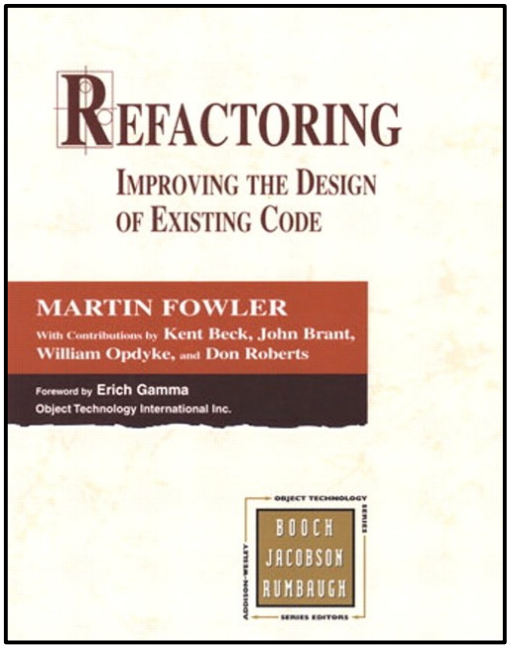
```
amountInvoicedIn(DateRange)
amountReceivedIn(DateRange)
amountOverdueIn(DateRange)

class DateRange {
   Date start;
   Date end;
}
```

Mise en œuvre

- La refonte nécessite l'utilisation d'environnements de développement évolués (menu « *refactor* » *d'IntelliJ*)

Lectures

- Martin Fowler, Refactoring « *Improving the Design of Existing Code* », Addison-Wesley Professional, 1999.



- Martin Fowler, « *Refactoring Home Page* », http://refactoring.com/catalog/index.html

Démo

- Kata Ératosthène
  - Ressources sur la plateforme Moodle
    - Kata
    - Vidéo
- Algorithme pour trouver les nombres premiers < n
  - Ératosthène dessine sur le sable un tableau de taille n
  - Il met une pierre dans chaque case indiquant que c'est un nombre premier
  - Puis en commençant à 2, il supprime tous ses multiples en parcourant les cases de 2 en 2 et en supprimant la pierre
  - Il recommence avec la case suivante qui contient une pierre (ici 3) et il supprimer ses multiples (en parcourant les cases de 3 en 3).

Refuctering

- Le refuctoring est le processus qui consiste à prendre un morceau de code bien conçu et, grâce à une série de petites modifications réversibles, à le rendre totalement impossible à maintenir par quiconque sauf vous-même.