GAUDEIX Edouard
GUOI Kevin

Qualité logicielle

Projet : Médiathèque

Figure 1: Software quality

Rapport de projet

GAUDEIX Edouard

GUOI Kevin

SOMMAIRE

Présentation du sujet

Actuellement en dernière année, nous nous proposons de réaliser des tests sur le projet « Médiathèque » dans le cadre du cours de qualité logicielle. Ce projet représente un système de gestion d'emprunts de documents par les clients d'une médiathèque.

Il existe différents types de clients pouvant emprunter plus ou moins de documents. La quantité de documents empruntable dépend de la catégorie à laquelle il appartient. Les tarifs et la durée d'emprunt varie selon le type de document et la catégorie du client.

En cas de dépassement de la date de rendu, le client est marqué et reçoit périodiquement des lettres de rappel.

Les clients et documents sont uniques déterminés respectivement par leur nom/prénom et leur code.

D'un point de vue technique, il nous ait demandé de faire des tests unitaires avec JUnit. Les spécifications techniques utilisées sont :

- eclipse Oxygen
- JUnit 4

Bugs

I. Package mediatheque

1) Mediatheque.java

l. 135 modifierGenre(String old, String neuf)

```
Genre g = chercherGenre(old);
if (g != null) {
 throw new OperationImpossible("Genre "+ old + " inexistant");
} else {
 g.modifier(neuf);
}
```

Une exception devrait être retournée lorsque le genre fournit n'est pas trouvé et non l'inverse.

Ainsi, il faut modifier le code comme suit :

```
Genre g = chercherGenre(old);
if (g == null) {
 throw new OperationImpossible("Genre "+ old + " inexistant");
} else {
 g.modifier(neuf);
}
```

l. 235 modifierLocalisation(Localisation loc, String s, String r)

```
Localisation inVector = chercherLocalisation(loc.getRayon(),
loc.getRayon());
```

La recherche de la localisation se fait avec la salle et le rayon.

Ainsi, il faut modifier le code comme suit :

```
Localisation inVector = chercherLocalisation(loc.getSalle(),
loc.getRayon());
```

l. 278 chercherCatClient(String catName)

```
int index = lesCatsClient.indexOf(searched);
if (index == 0) {
  return lesCatsClient.elementAt(index);
} else {
  return null;
}
```

La méthode retourne la catégorie client correspondant à l'index donné en paramètre. L'index doit retourner tout client présent dans la liste non pas juste le premier client.

Ainsi, il faut modifier le code comme suit :

```
int index = lesCatsClient.indexOf(searched);
if (index >= 0) {
  return lesCatsClient.elementAt(index);
} else {
  return null;
}
```

l. 355 modifierCatClient(CategorieClient co, String name, int max, double cot, double coefDuree, double coefTarif, boolean codeReducUsed)

```
CategorieClient c = chercherCatClient(co.getNom());
if (c == null) {
  throw new OperationImpossible("Categorie client " +
co.getNom() + " inexistante");
} else {
  if (!co.getNom().equals(name)) {
    co.modifierNom(name);
  }
  if (co.getNbEmpruntMax() != max) {
    co.modifierMax(max);
  }
  if (co.getCotisation() != cot) {
    co.modifierCotisation(cot);
  }
  if (co.getCoefDuree() != coefDuree) {
    co.modifierCoefDuree(coefDuree);
  }
  if (co.getCoefTarif() != coefTarif) {
    co.modifierCoefTarif(coefTarif);
  }
  if (co.getCodeReducUtilise() != codeReducUsed) {
    co.modifierCodeReducActif(codeReducUsed);
  }
}
return c;
```

La méthode ne modifie pas la catégorie client de la médiathèque mais celle donnée par l'appelant. Il faut soit enlever c de la liste des catégories clients, modifier co puis ajouter co soit modifier c directement.

Ainsi, il faut modifier le code comme suit :

```
CategorieClient c = chercherCatClient(co.getNom());
if (c == null) {
  throw new OperationImpossible("Categorie client " +
co.getNom() + " inexistante");
} else {
  if (!c.getNom().equals(name)) {
    c.modifierNom(name);
  }
  if (c.getNbEmpruntMax() != max) {
    c.modifierMax(max);
  }
  if (c.getCotisation() != cot) {
    c.modifierCotisation(cot);
  }
  if (c.getCoefDuree() != coefDuree) {
    c.modifierCoefDuree(coefDuree);
  }
  if (c.getCoefTarif() != coefTarif) {
    c.modifierCoefTarif(coefTarif);
  }
  if (c.getCodeReducUtilise() != codeReducUsed) {
    c.modifierCodeReducActif(codeReducUsed);
  }
}
return c;
```

l.471 metEmpruntable():

```
doc.metConsultable();
```

La méthode metEmpruntable doit rendre le document empruntable et non l'inverse.

Ainsi, il faut modifier le code comme suit :

```
doc.metEmpruntable();
```

l.924 getClientAt (int n):

```
Client cl = null;
for (i = 0; i <= n; i++) {
  if (ic.hasNext()) {
    cl = ic.next();
  } else {
    break;
  }
}
return cl;
```

La méthode retourne le client à l'index donné n. Or si n est plus grand que la liste, la valuer retournée ne sera pas null mais le dernier client de la liste.

Il faut modifier le code comme suit :

```
Client cl = null;
for (i = 0; i <= n; i++) {
  if (ic.hasNext()) {
    cl = ic.next();
  } else {
    cl = null;
  }
}
return cl;
```

2) Genre.java

l. 20 attribut valeur par défaut

```
private int nbEmprunts=10;
```

Le nombre d'emprunt par défaut de chaque genre doit être à 0.
Il faut modifier le code comme suit :

```
private int nbEmprunts=0;
```

l. 28 constructeur (String n)

```
//nbEmprunts=0;
```

Le nombre d'emprunt par défaut de chaque genre doit être à 0. Cette solution et/ou la précédente est nécessaire.

Ainsi, il faut modifier le code comme suit :

```
nbEmprunts=0;
```

l. 36 emprunter()

```
nbEmprunts = nbEmprunts+2;
```

Le nombre d'emprunt s'incrémente à chaque emprunt. L'incrémentation devrait être d'un et non de deux.

Ainsi, il faut modifier le code comme suit :

```
nbEmprunts = nbEmprunts+1;
```

II. Package mediatheque.client

1) CategorieClient.java

l. 72 modifierCotisation(double cot)

```
cotisation = 4;
```

La méthode doit setter la valeur de cotisation par la valeur fournit, cot, et non à 4.

Ainsi, il faut modifier le code comme suit :

```
cotisation = cot;
```

2) Client.java

l. 188 getNbEmpruntsEnRetard ()

```
return 1;
```

La méthode doit retourner le nombre d'emprunts en retard, nbEmpruntsDepasses, et non 1.

Ainsi, il faut modifier le code comme suit :

```
return nbEmpruntsDepasses;
```

l. 188 emprunter ()

```
public void emprunter() {
  assert peutEmprunter();
  nbEmpruntsEffectues++;
  nbEmpruntsEnCours++;
}
```

La méthode doit incrémenter toutes les variables statistiques, effectuées, en cours, total.

Ainsi, il faut modifier le code comme suit :

```
public void emprunter() {
  assert peutEmprunter();
  nbEmpruntsEffectues++;
  nbEmpruntsEnCours++;
  nbEmpruntsTotal++;
}
```

l. 300 marquer ()

```
if(nbEmpruntsDepasses == nbEmpruntsEnCours)
```

La méthode devrait prendre en compte des cas spécifiques telle que le cas 0 ou le cas d'erreurs système avec nbEmpruntsDepasses supérieur à nbEmpruntsEnCours.

Il faut modifier le code comme suit :

```
if((nbEmpruntsDepasses >= nbEmpruntsEnCours) &&
nbEmpruntsDepasses != 0))
```

Ceci n'est pas une véritable erreur mais plus des bonnes pratiques.

l.114 constructor (String nom, String prenom)

Les variables ne sont pas toutes initialisées.
Par exemple, catClient peut causer une exception, NullPointerException.

III. Package mediatheque.document

1) Document.java

l.198 metEmpruntable()

```
empruntable = false;
```

Cette fonction autorise l'emprunt du document donc « empruntable » ne doit pas être « false ».

Ainsi, il faut modifier le code comme suit :

```
empruntable = true;
```

2) Livre.java

l. 53 constructor (String code, Localisation localisation, String titre, String auteur, String annee, Genre genre, int nombrePage)

```
if (nombrePage < 0)
```

Dans ce constructeur, l'opération impossible doit être déclenché si le nombre de pages est inférieur à 0 mais également s'il est égal à 0.

Ainsi il faut modifier le code comme suit :

```
if (nombrePage < 1) {
```

l. 78 emprunter()

```
public boolean emprunter() throws InvariantBroken,
OperationImpossible {
  super.emprunter();
  return true;
}
```

Pour cette fonction, il est nécessaire d'auto-incrémenter le nombre d'emprunts total.

Ainsi, il faut modifier le code comme suit :

```
public boolean emprunter() throws InvariantBroken,
OperationImpossible {
  super.emprunter();
  nbEmpruntsTotal++;
  return true;
}
```

3) Video.java

l. 58 constructor(String code, Localisation localisation, String titre, String auteur, String annee, Genre genre, int dureeFilm, String mentionLegale)

```
if (dureeFilm == 0 || mentionLegale == null)
```

Dans ce constructeur, l'opération impossible doit être déclenché si la durée du film est égale à 0 mais également s'il est inférieur à 0.

Ainsi, il faut modifier le code comme suit :

```
if (dureeFilm < 1|| mentionLegale == null) {
```

IV. Package util

1) Datutil.java

l. 59 addDate (Date date, int nbjour)

```
greg.add(Calendar.DATE, nbjour-10);
```

Le « nbjour-10 » cause un décalage de 10 jours sur le calendrier comparé au nombre de jour souhaité. Il faut prendre en compte uniquement le nombre de jour « nbjour ».

Ainsi, il faut modifier le code comme suit :

```
greg.add(Calendar.DATE, nbjour);
```

Bilan

Bilan de l'équipe

Le projet de qualité logicielle « Médiathèque » a grandement été utile à tous les membres de l'équipe.

En effet, il nous a permis non seulement de revoir les bases en Java mais également de nouvelles notions telles que les tests unitaires avec JUnit.

En conclusion, malgré les difficultés rencontrées nous gardons une expérience positive de ce projet. Et nous espérons pouvoir utiliser ces notions dans nos futurs projets qu'ils soient scolaires ou professionnels.