Projet Ingénierie Test Intégration Continue : Gestion d'une Médiathèque

Castel Aurélien - Delamotte Guillaume - Descroix Guillaume
Doz Louka - Forget Nicolas

ensiie
Paris | Évry-Courcouronnes

Sommaire

- Tests fonctionnels
  - Emprunter & Restituer
  - Mettre consultable
- Tests structurels
  - Emprunter()
  - ChercherEmprunt()
  - ModifierClient()
- Tests mutationnels
  - Emprunter()

ensiie

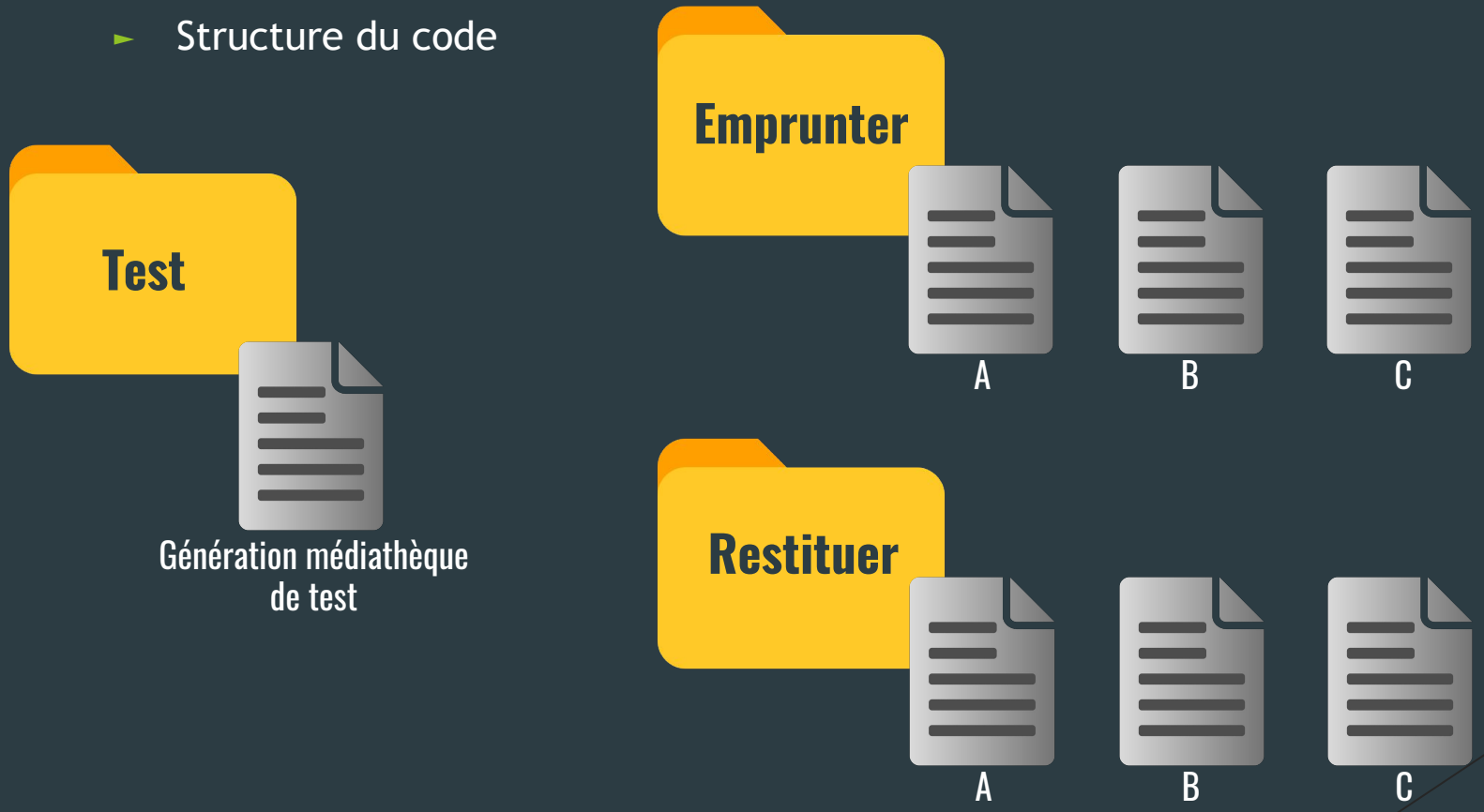
Tests fonctionnels
Emprunter & Restituer

- Démarches
  - Recherche des classes d'équivalences et des résultats espérés
    - **Test combinatoire** : 3 catégories du client * 3 types de documents * 2 (changement de catégorie)
    - **Tests aux limites**
  - Détermination **des préfixes** (conditions de départ) et **des postfixes** (conditions d'arrivée)
  - Assimilation avec les méthodes de construction d'une médiathèque
  - Implémentation des différentes classes d'équivalences

Tests fonctionnels

Emprunter & Restituer

- Implémentations
  - Structure du code



Code	Classe de test	Fonction de test	Classes d'équivalence	Préfixe	Postfixe	Résultat attendu
CT1	TestEmprunter.java	emprunterArcticleInexistant()	Emprunter un document inexistant	Créer une médiathèque avec un cleint	Le nombre de document emprunté de l'utilisateur ne change pas	exception

Tests fonctionnels

Emprunter & Restituer

- Implémentations
  - Exemple : Réservation article inexistant

```
public class TestEmprunter{

   private Mediatheque M;

   // Création d'un environnement de test
   @BeforeEach
   public void creerEnvironnementTest() {
      try {
         // Génération de la médiathèque de tests
         GenerationMediatheque.MediathequeTestEmprunter();
         // Récupération de la médiathèque de tests
         M = new Mediatheque("MediathequeTestEmprunter");
      } catch (Exception e) {
         fail();
      }
   }

   // Objectif du test : (CT1) Vérifier l'impossibilité de réserver un article inexistant
   // Résultat attendu : Levée d'une exception de type OperationImpossible
   @Test
   public void emprunterArcticleInexistant() {
      int nbEmpruntCli = M.chercherClient("Descroix","Guillaume").getNbEmpruntsEffectues();
         Assertions.assertThrows(OperationImpossible.class, () -> {
            M.emprunter("Descroix","Guillaume","codeArticleInexistant");
         }
      );
      Assertions.assertEquals(nbEmpruntCli,M.chercherClient("Descroix","Guillaume").getNbEmpruntsEffectues());
   }
   ...
}
```

Médiathèque de test

Initialisation de la médiathèque

2 Tests :

Génération d'une exception

Le compteur d'emprunt de l'utilisateur n'a pas été incrémenté

Tests fonctionnels

Emprunter & Restituer

- Résultats
  - Tous nos tests passent sans erreur
  - La réalisation des tests a permis de mettre en évidence une anomalie dans le code
- Point sur l'avancement

  - Finir l'implémentation des tests pour les différentes classes

Tests fonctionnels

Mettre consultable

Mettre un article (livre) consultable (cas normal)	Pas d'exception	Le test passe cas normal par rapport au sujet
Mettre un article (livre) consultable en vérifiant que l'état renvoyé est true (cas normal)	Pas d'exception	Le test passe cas normal par rapport au sujet
Mettre consultable un article qui est null	exception	Impossible à réaliser car null ne représente rien
Mettre consultable un livre consultable	Pas d'exception	Le test passe cas normal par rapport au sujet
Mettre consultable un livre non consultable	exception	Impossible à réaliser car le livre n'est plus consultable
Mettre consultable un livre en cours d'emprunt	exception	Impossible à réaliser car le livre n'est pas en cours d'emprunt
Mettre consultable un audio consultable	Pas d'exception	Le test passe cas normal par rapport au sujet
Mettre consultable un audio non consultable	exception	Impossible à réaliser car l'audio n'est plus consultable
Mettre consultable un audio en cours d'emprunt	exception	Impossible à réaliser car l'audio n'est pas en cours d'emprunt
Mettre consultable une video consultable	Pas d'exception	Le test passe cas normal par rapport au sujet
Mettre consultable une video non consultable	exception	Impossible à réaliser car la video n'est plus consultable
Mettre consultable une video en cours d'emprunt	exception	Impossible à réaliser car la video n'est pas en cours d'emprunt

Tests fonctionnels

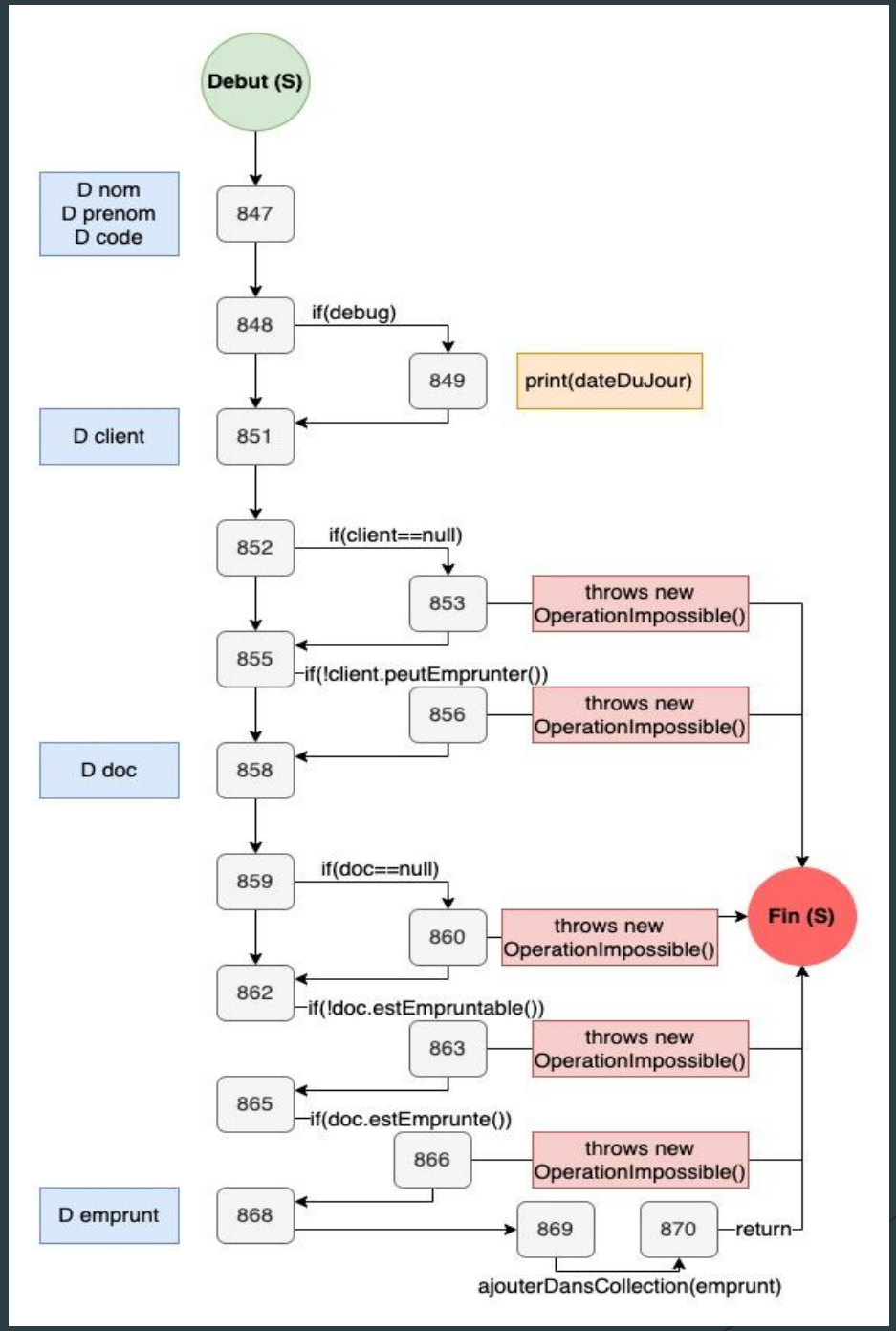
Mettre consultable

Exemple: Mettre consultable un article qui est null

```
// Code CT2
// Objectif de test : Mettre consultable un article qui est null
// Resultat attendu : exception
// Resultat obtenu: exception
@Test
public void mettreConsultableArticleNull() {
    try {
        //Si on ne catch pas une exception : echec
        M.metConsultable(null);
        fail();
    } catch (OperationImpossible oi) {
        //si une exception de type OperationImpossible est levee, le test reussit
        oi.printStackTrace();
    } catch (Exception e) {
        fail();
    }

}
```

Tests structurels
emprunter()



Tests structurels

emprunter()

All-paths : 1 chemins possibles

All-nodes, All-arcs, All-2-paths

Numéro	All-paths	Faisable
1	848.849.851.852.853	Non
2	848.849.851.852.855.856	Non
3	848.849.851.852.855.858.859.860	Non
4	848.849.851.852.855.858.859.860.862.863	Non
5	848.849.851.852.855.858.859.860.862.865.866	Non
6	848.849.851.852.855.858.859.860.862.865.868.869.670	Oui
Numéro	**All-nodes**	**Faisable**
CH1	848.849.851.852.853	Non
CH2	848.849.851.852.855.856	Non
CH3	848.849.851.852.855.858.859.860	Non
CH4	848.849.851.852.855.858.859.860.862.863	Non
CH5	848.849.851.852.855.858.859.860.862.865.866	Non
CH6	848.849.851.852.855.858.859.860.862.865.868.869.670	Oui
Numéro	**All-arcs**	**Faisable**
1	S.(848 - 849).(849 - 851).(851 - 852).(852 - 853).E	Non
2	S.(848 - 849).(849 - 851).(851 - 852).(852 - 855).(855 - 856).E	Non
3	S.(848 - 849).(849 - 851).(851 - 852).(852 - 855).(855 - 858).(858 - 859).(859 - 860).E	Non
4	S.(848 - 849).(849 - 851).(851 - 852).(852 - 855).(855 - 858).(858 - 859).(859 - 862).(862 - 863).E	Non
5	S.(848 - 849).(849 - 851).(851 - 852).(852 - 855).(855 - 858).(858 - 859).(859 - 862).(862 - 865).(865 - 866).E	Non
6	S.(848 - 849).(849 - 851).(851 - 852).(852 - 855).(855 - 858).(858 - 859).(859 - 862).(862 - 865).(865 - 868).(868 - 870).E	Oui
Numéro	**All-2-paths**	**Faisable**
Pas de boucles		

Tests structurels

emprunter()

Prédicats chemins

CH6	848.849.851.852.855.858.859.860.862.865.868.869.670	Oui

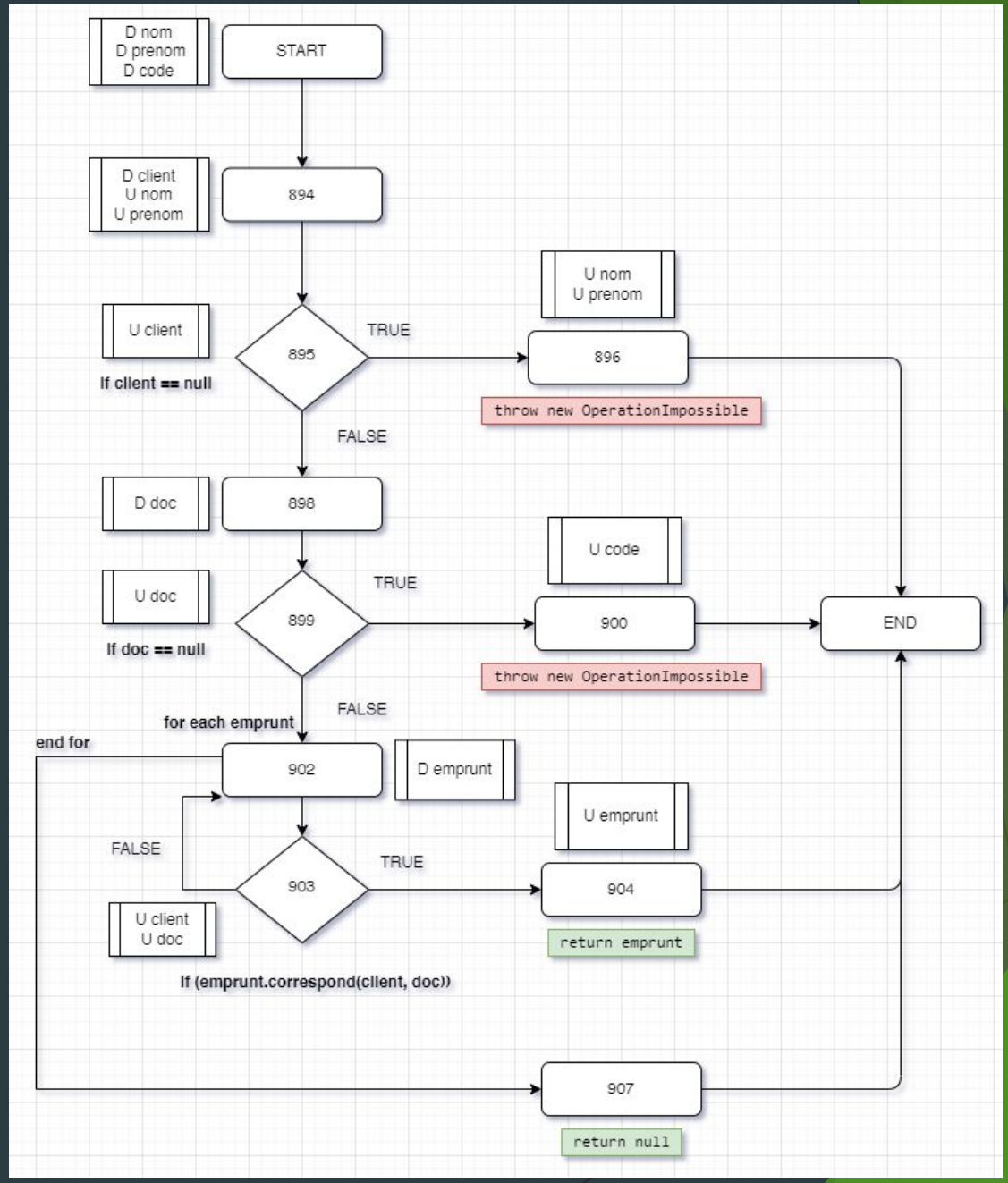
Numéro	Chemin	Description		
CH6	848.849.851.852.855.858.859.860.862.865.868.869.670	Emprunt d'un document par un client.		
Sommet	**Valeur symbolique du client**	**Valeur symbolique du document**	**Valeur symbolique du fichier d'emprunt**	**Condition**
848	Nom = “nom” Prenom = “prenom” Code = “code”			if (debug)
849	Nom = “nom” Prenom = “prenom” Code = “code”			
851	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"			
852	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"			if (client == null)
855	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"			if (!client.peutEmprunter())
858	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"	Document = "doc"		
859	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"	Document = "doc"		if (doc == null)
860	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"	Document = "doc"		
862	Nom = “nom” Prenom = “prenom” Code = “code” Client = "client"	Document = "doc"		if (!doc.estEmpruntable())
865	Nom = “nom” Prenom = “prenom” Code = “code”	Document = "doc"		if (doc.estEmprunte())

ensiie

Tests structurels

chercherEmprunt()

Graphe flot de contrôle/données



Tests structurels

chercherEmprunt()

All-paths : 4 chemins possibles

Numéro	All-paths	Faisable
1	894.895.896	Oui
2	894.895.898.899.900	Oui
3	894.895.898.902.907	Oui
4	894.895.898.902.903.904	Oui

All-nodes, All-arcs, All-2-paths

Numéro	All-nodes	Faisable
Ch1	894.895.896	Oui
Ch2	894.895.898.899.900	Oui
Ch3	894.895.898.902.907	Oui
Ch4	894.895.898.902.903.904	Oui

Numéro	All-arcs	Faisable
1	S.(894 - 895).(895 - 896).E	Oui
2	S.(894 - 895).(895 - 898).(898 - 899).(899 - 900).E	Oui
3	S.(894 - 895).(895 - 898).(898 - 899).(899 - 902).(902 - 903).(903 - 904).E	Oui
4	S.(894 - 895).(895 - 898).(898 - 899).(899 - 902).(902 - 903).(902 - 907).E	Oui

Numéro	All-2-paths	Faisable
1	894.895.896	Oui
2	894.895.898.899.900	Oui
3	894.895.898.902.907	Oui
4	894.895.898.902.903.904	Oui
5	894.895.898.902.903.902.907	Oui
6	894.895.898.902.903.902.904	Oui

Tests structurels

chercherEmprunt()

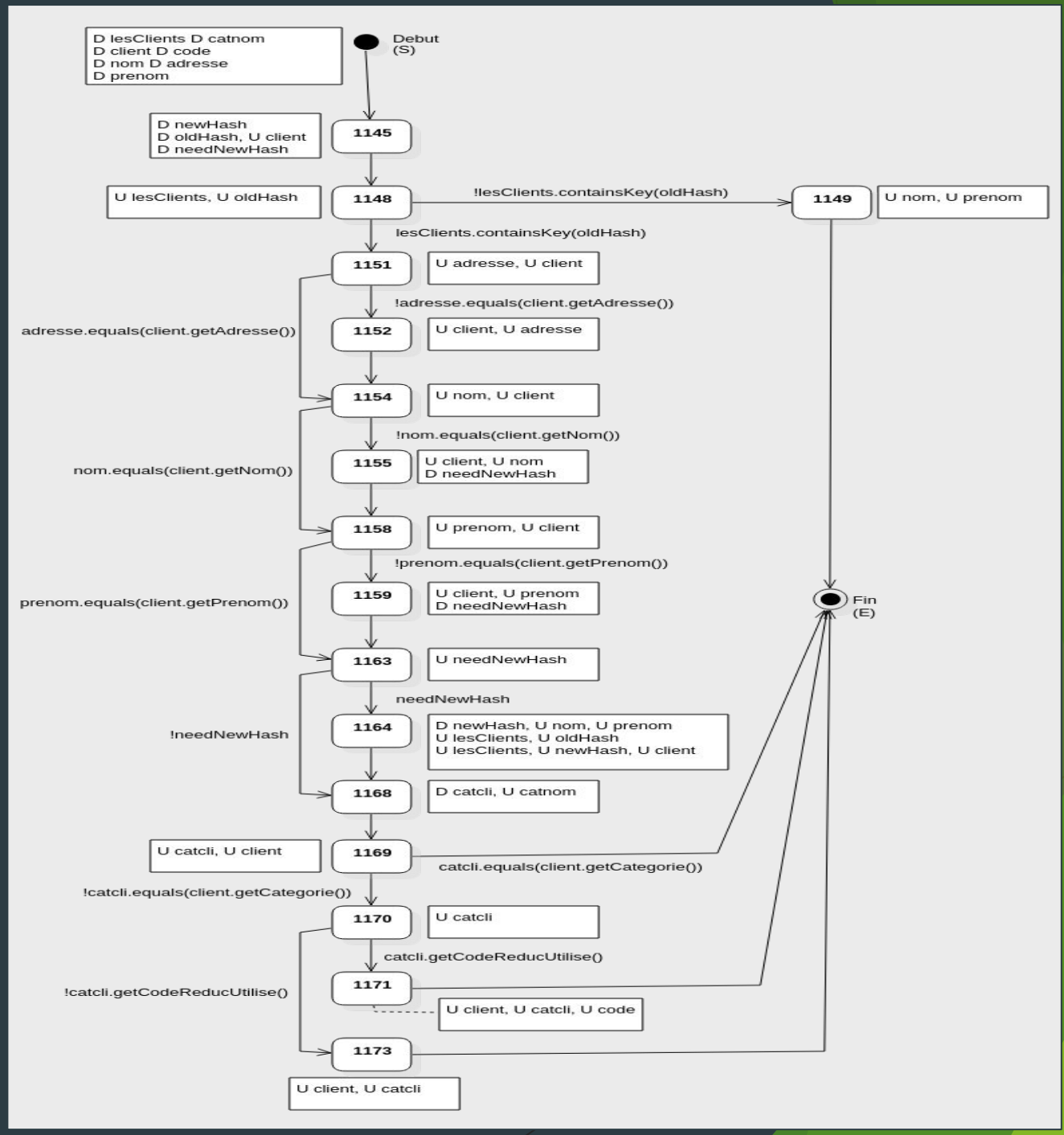
Prédicats chemins

Numéro	Chemin	Description
1	Ch1	Chercher l'emprunt d'un client, avec un client non défini (null)
Sommet	**Valeurs symbolique**	**Condition**
894	nom = nom0 prenom = prenom0 code = code0 client = null	
895	nom = nom0 prenom = prenom0 code = code0 client = null	client == null
896	nom = nom0 prenom = prenom0 code = code0 client = null	

Tests structurels

modifierClient()

Graphe flot de contrôle



Tests structurels

modifierClient()

Chemin d'exécution en expressions régulières
S.1145.1148(1149.E+ε).1151(1152+ε).1154(1155+ε).1158.(1159+ε).1163(1164+ε).1168.1169(E+1170(1171+1173))

All-paths

48 chemins possibles

20	1145.1148.1151.1152.1154.1155.1158.1163.1164.1168.1169.1170.1171	Oui
21	1145.1148.1151.1152.1154.1155.1158.1163.1164.1168.1169.1170.1173	Oui
22	1145.1148.1151.1152.1154.1158.1159.1163.1164.1168.1169.1170.1171	Oui
23	1145.1148.1151.1152.1154.1158.1159.1163.1164.1168.1169.1170.1173	Oui
24	1145.1148.1151.1152.1154.1155.1158.1159.1163.1164.1168.1169.1170.1171	Oui
25	1145.1148.1151.1152.1154.1155.1158.1159.1163.1164.1168.1169.1170.1173	Oui
26	1145.1148.1151.1154.1158.1163.1164.1168.1169	Non (needNewHash jamais à true)
27	1145.1148.1151.1152.1154.1158.1163.1164.1168.1169	Non (needNewHash jamais à true)
28	1145.1148.1151.1154.1155.1158.1163.1168.1169	Non (needNewHash forcément à true)
29	1145.1148.1151.1154.1158.1159.1163.1168.1169	Non (needNewHash forcément à true)
30	1145.1148.1151.1154.1155.1158.1159.1163.1168.1169	Non (needNewHash forcément à true)

Tests structurels

modifierClient()

All-nodes, All-arcs, All-k-paths

Numéro	All-nodes	Faisable
ch1	1145.1148.1149	Oui
ch2	1145.1148.1151.1154.1158.1163.1168.1169.1170.1173	Oui
ch3	1145.1148.1151.1152.1154.1155.1158.1159.1163.1164.1168.1169.1170.1171	Oui

Numéro	All-arcs	Faisable
1	1145.1148.1149.E	Oui
2	1145.1148.1151.1152.1154.1155.1158.1159.1163.1164.1168.1169.1170.1171.E	Oui
3	1145.1148.1151.1154.1158.1163.1168.1169.1170.1173.E	Oui

Numéro	All-2-paths	Faisable
Pas de boucles		

Tests structurels

modifierClient()

Prédicats chemins

Numéro	All-nodes	Faisable
ch1	1145.1148.1149	Oui

Numéro	Chemin	Description	
1	ch1	Modification d'un client non présent dans la médiathèque	
Sommet	**Valeur symbolique du client**	**Valeur symbolique de needNewHash**	**Condition**
1145	Nom = nom0 Prenom = prenom0 Adresse = adresse0 Catnom = categorie0 Code = code0	false	
1148	Nom = nom0 Prenom = prenom0 Adresse = adresse0 Catnom = categorie0 Code = code0	false	client != null
1149	Nom = nom0 Prenom = prenom0 Adresse = adresse0 Catnom = categorie0 Code = code0	false	client != null ^ (client.nom != nomOK v client.prenom != prenomOK)

Tests structurels

modifierClient()

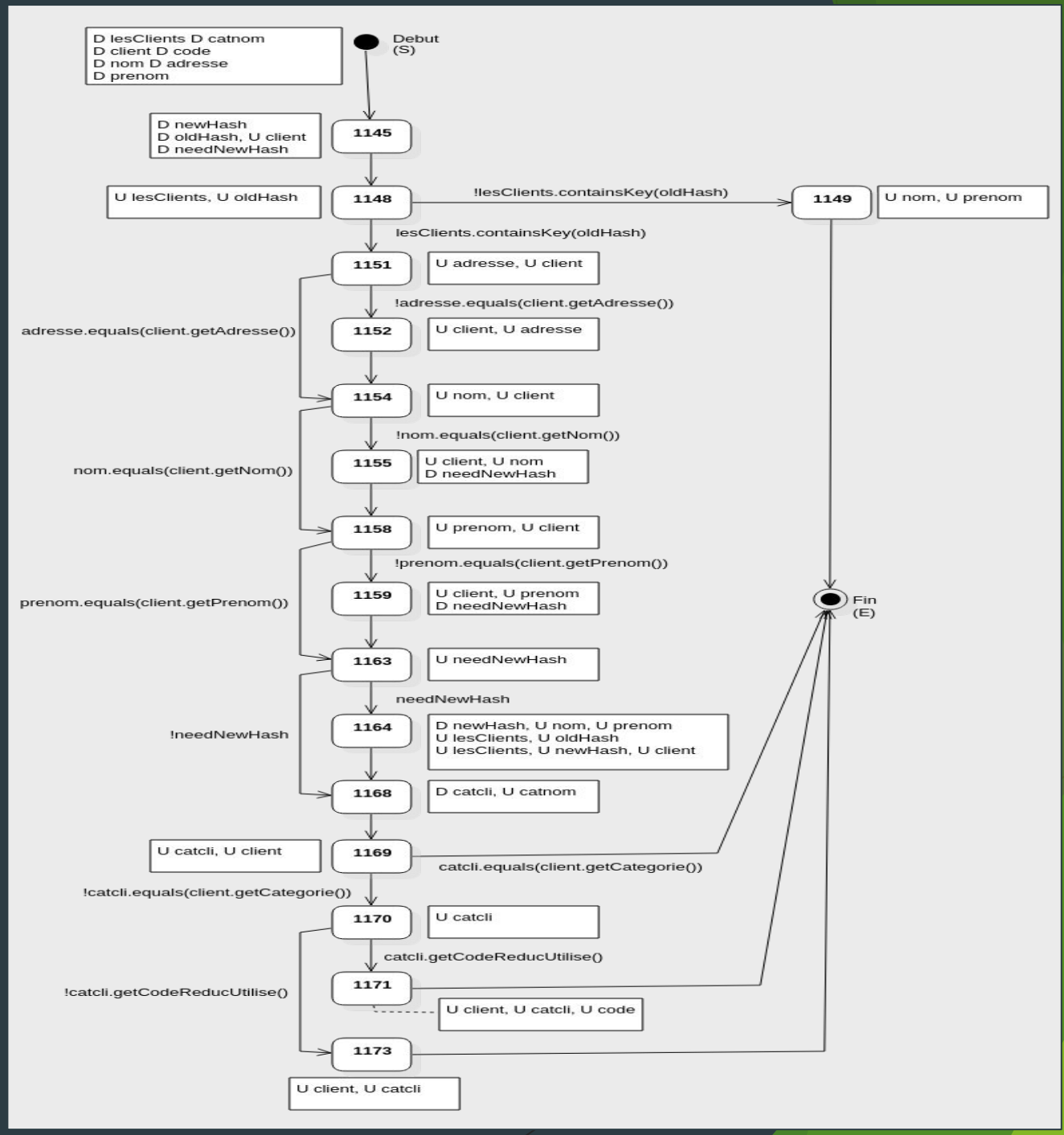
Cas de tests

	Nom	Valeurs dans objet	
	Client(1)	adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », codeActif = false	
	Client(2)	adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », codeActif = true	
Numéro	**Classe d'équivalence**	**Arguments**	**Résultat attendu**
1	**Client**		
1.1	Client vaut null	Client = null, adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », code = « »	Exception NullPointerException
1.2	Couple nom/prénom non présents dans la médiathèque	Client = Client(1), adresse = « adresse0 », nom = « », prénom = « », catnom = « categorie0 », code = « »	Exception OperationImpossible
2	**Changements**		
2.1	Changement adresse	Client = Client(1), adresse = « adresse1 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », code = « »	Adresse client = « adresse1 »
2.2	Changement nom	Client = Client(1), adresse = « adresse0 », nom = « nom1 », prénom = « prenom0 », catnom = « categorie0 », code = « »	Nom client = « nom1 »
2.3	Changement prénom	Client = Client(1), adresse = « adresse0 », nom = « nom0 », prénom = « prenom1 », catnom = « categorie0 », code = « »	Prénom client = « prenom1 »
3	**Changement catégorie (sans code actif)**		
3.1	Changement catégorie (sans code actif)	Client = Client(1), adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie1 », code = « »	Catégorie client = « categorie1 » et code actif pour la catégorie client = false
3.2	Ajout code avec la même catégorie client	Client = Client(1), adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », code = « code0 »	Catégorie client = « categorie0 » et code actif pour la catégorie client = false
4	**Changement catégorie (avec code actif)**		
4.1	Changement catégorie (avec code actif)	Client = Client(2), adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », code = « »	Catégorie client = « categorie1 » et code actif pour la catégorie client = true
4,2	Catégorie non présente dans la médiathèque	Client = Client(2), adresse = « adresse0 », nom = « nom0 », prénom = « prenom0 », catnom = « categorie0 », code = « »	Exception NullPointerException

Tests structurels

modifierClient()

Graphe flot de données



Tests structurels

modifierClient()

Tous les du-chemins

Numéro	Tous-les-du-chemins	Couple	Variable
	S 1145 1148 1151 1154 1158 1163 1168 1169 E	(S, 1148)	lesClients
	S 1145 1148 1151 1154 1155 1158 1163 1164 1168 1169 E	(S, 1164)	lesClients
	S 1145 1148 1151 1154 1155 1158 1163 1164 1168 1169 E	(S, 1145)	client
	S 1145 1148 1151 1152 1154 1155 1158 1159 1163 1164 1168 1169 1170 1171 E	(S, 1151)	client
	S 1145 1148 1151 1152 1154 1155 1158 1159 1163 1164 1168 1169 1170 1171 E	(S, 1152)	client

Aucun chemin de convient	(1145, 1164)	newHash
S 1145 1148 1151 1152 1154 1155 1158 1159 1163 1164 1168 1169 1170 1171 E	(1164, 1164)	newHash

```
1145        ClefClient newHash;
1164        newHash = new ClefClient(nom, prenom);
1165        lesClients.remove(oldHash);
1166        lesClients.put(newHash, client);
```

Définition

Définition

Utilisation

Tests mutationnels

Emprunter()

Classe de test	Fonction de test	Changement dans le mutant
TestMutationnel.ja	testMutantsTues1()	if (client != null) { ligne 1409
TestMutationnel.ja	testMutantsTues2()	if (client.peutEmprunter()) { ligne 1447
TestMutationnel.ja	testMutantsTues3()	if (doc != null) { ligne 1486
TestMutationnel.ja	testMutantsTues4()	if (doc.estEmpruntable()) { ligne 1524
TestMutationnel.ja	testMutantsTues5()	if (!doc.estEmprunte()) { ligne 1562
		Le non mutant
		if (client == null) {
		If (!client.peutEmprunter()) {
		if (doc == null) {
		If (!doc.estEmpruntable()) {
		if (doc.estEmprunte()) {

Tests mutationnels

Emprunter()

Exemple: client != null

```
public void emprunter1(final String nom, final String prenom, final String code)
    if (debug) {
        System.out.println("MediathequeTestMutationnel: emprunter le " + Datutil
    }
    Client client = chercherClient(nom, prenom);
    if (client != null) {
        throw new OperationImpossible("Client " + nom + " " + prenom + " inexist
    }
    if (!client.peutEmprunter()) {
        throw new OperationImpossible("Client " + client.getNom() + " non autori
    }
    Document doc = chercherDocument(code);
    if (doc == null) {
        throw new OperationImpossible("Document " + code + " inexistant");
    }
    if (!doc.estEmpruntable()) {
        throw new OperationImpossible("Document " + doc.getCode() + " non emprun
    }
    if (doc.estEmprunte()) {
        throw new OperationImpossible("Document " + doc.getCode() + " deja empru
    }
    FicheEmprunt emprunt = new FicheEmprunt(client, doc);
    ajouterDansCollection(emprunt);
    return;
}
```

Tests mutationnels

Emprunter()

Analyse: Lorsqu'il y a un client on donne l'exception

```
public void emprunter1(final String nom, final String prenom, final String code)
    if (debug) {
        System.out.println("MediathequeTestMutationnel: emprunter le " + Datutil
    }
    Client client = chercherClient(nom, prenom);
    if (client != null) {
        throw new OperationImpossible("Client " + nom + " " + prenom + " inexist
    }
    if (!client.peutEmprunter()) {
        throw new OperationImpossible("Client " + client.getNom() + " non autori
    }
    Document doc = chercherDocument(code);
    if (doc == null) {
        throw new OperationImpossible("Document " + code + " inexistant");
    }
    if (!doc.estEmpruntable()) {
        throw new OperationImpossible("Document " + doc.getCode() + " non emprun
    }
    if (doc.estEmprunte()) {
        throw new OperationImpossible("Document " + doc.getCode() + " deja empru
    }
    FicheEmprunt emprunt = new FicheEmprunt(client, doc);
    ajouterDansCollection(emprunt);
    return;
}
```

Tests mutationnels

Emprunter()

Message terminal	Analyse Résultat
fr.ensiie.itic.mediatheque.util.OperationImpossible: Client Descroix Guillaume inexistant	L'erreur qu'il n'y a pas de client est levée alors qu'il y en a bien
fr.ensiie.itic.mediatheque.util.OperationImpossible: Client Descroix non autorise a emprunter	L'erreur que le client peut emprunter est levée alors qu'il peut
fr.ensiie.itic.mediatheque.util.OperationImpossible: Document 1 inexistant	L'erreur que le document est inexistant est levée alors qu'il existe
fr.ensiie.itic.mediatheque.util.OperationImpossible: Document 1 non empruntable	L'erreur que le document est non empruntable est levée alors qu'il est empruntable
fr.ensiie.itic.mediatheque.util.OperationImpossible: Deja Emprunte[Audio] "1" Homework Daft	L'erreur que le document a été emprunté est levée alors qu'il n'a pas été emprunté

Questions ?