Système GestionSerrures pour la gestion des serrures programmables

Denis Conan

Janvier 2020

Table des matières

Pour tester notre logiciel GestionClefsHôtel, nous avons besoin de simuler des serrures reprogrammables ; c'est le rôle du système GestionSerrures, que nous présentons maintenant. Le système GestionSerrures est entièrement développé ; le code est disponible dans le projet fourni au départ (cf. le paquetage eu.telecomsudparis.csc4102.gestionserrures).

NB : cette section n'est pas à lire dès le début du module, mais au fur et à mesure de l'avancée dans le Sprint 1. Plus précisément, nous détaillons :

- pour la séance 2, la spécification avec les fonctionnalités de simulation disponibles avec leurs préconditions et postconditions (dans la section 1),
- pour la séance 2, la préparation des tests de validation sous la forme de tables de décision (dans la section 2),
- pour la séance 3, les éléments de conception permettant de comprendre et d'utiliser GestionSerrures (dans la section 3),
- pour les séances 5 et 6, dans le code, la mise en œuvre du système ainsi que de ses tests unitaires et de validation.

1 Spécification

Les fonctionnalités sont décrites dans un diagramme de cas d'utilisation dans la section 1.1, et leurs préconditions et postconditions dans la section 1.2.

1.1 Diagramme de cas d'utilisation

Le système GestionSerrures possède un seul acteur car c'est un système de simulation, donc sans rôle particulier. L'utilisateur peut créer, lister et supprimer les serrures. Il peut aussi ré-initialiser la serrure en fournissant une nouvelle graine pour la génération des clefs. Enfin, il peut simuler la présentation d'un badge pour essayer d'ouvrir la porte. Le diagramme de cas d'utilisation est présenté dans la figure qui suit.

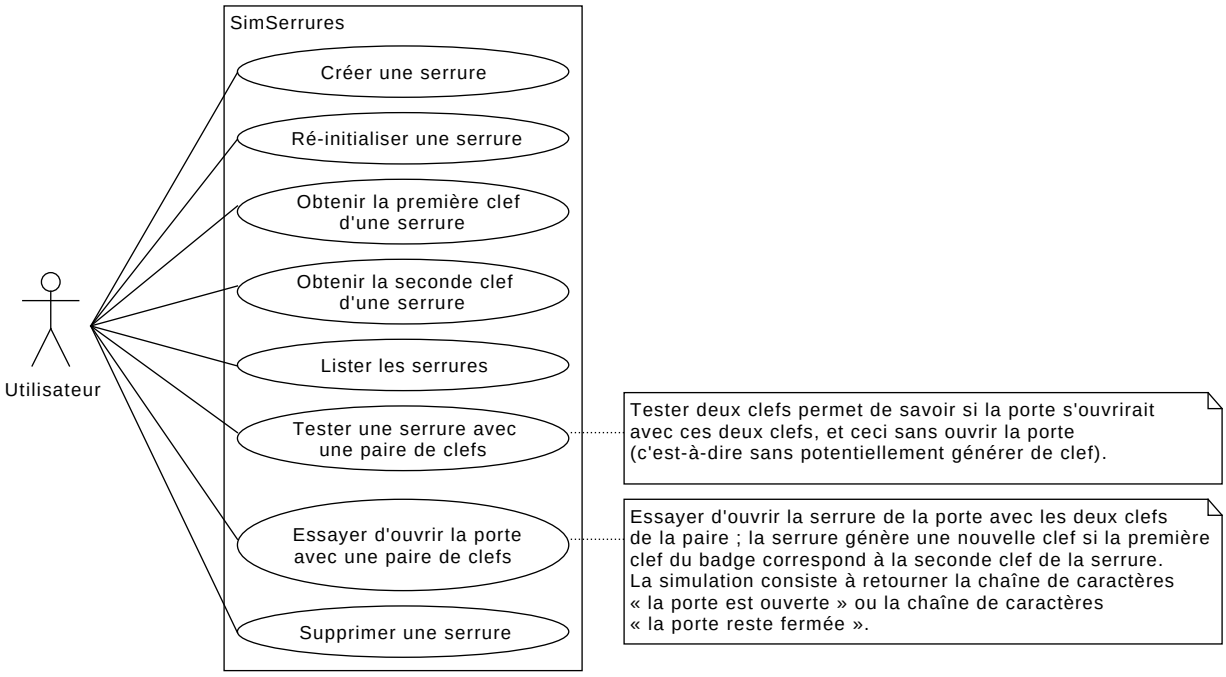

FIGURE 1 – Diagramme de cas d'utilisation du système GESTIONSERRURES

1.2 Priorités, préconditions et postconditions des cas d'utilisation

Nous donnons les préconditions et postconditions pour les cas d'utilisation de priorité « Haute ». Pour les autres, nous indiquons uniquement leur niveau de priorité.

HAUTE n° 1
- Créer une serrure
  - précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant inexistante
  - postcondition : serrure avec cet identifiant existante $\wedge$ les deux clefs sont les deux premières valeurs générées à partir de la graine et du sel

HAUTE n° 2
- Ré-initialiser les clefs d'une serrure
  - précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ nouvelle graine de la serrure bien formée (non null et non vide) $\wedge$ nouvelle graine ou nouveau sel différents
  - postcondition : une nouvelle graine $\vee$ un nouveau sel
    $\wedge$ première clef de la nouvelle paire $\neq$ première clef de l'ancienne paire
    $\wedge$ première clef de la nouvelle paire $\neq$ seconde clef de l'ancienne paire
    $\wedge$ seconde clef de la nouvelle paire $\neq$ première clef de l'ancienne paire
    $\wedge$ seconde clef de la nouvelle paire $\neq$ seconde clef de l'ancienne paire

HAUTE n° 3
- Ré-initialiser les clefs d'une serrure
  - précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ graine de la serrure bien formé (non null et non vide)

— postcondition : serrure avec ce identifiant existante $\wedge$ les deux clefs sont les deux premières valeurs générées à partir de la graine

Moyenne — Obtenir la première clef d'une serrure

Moyenne — Obtenir la seconde clef d'une serrure

Moyenne — Lister les serrures

Haute n° 4 — Tester une serrure avec deux clefs

— précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ valeurs fournies pour les clefs K_1 et K_2 bien formées (non null et non vides)

— postcondition :
$\Big[K_1$ et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure $\vee$ K_1 fournie est identique à la seconde clef de la serrure$\Big]$

Haute n° 5 — Essayer d'ouvrir la porte avec un badge

— précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ valeur des clefs K_1 et K_2 bien formées (non null et non vides) $\wedge$
$\Big[K_1$ et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure $\vee K_1$ fournie était identique à la seconde clef de la serrure$\Big]$

— postcondition :
$\big(K_1$ et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure $\implies$ la porte est ouverte et les clefs de la serrure reste inchangées$\big)$
$\vee\big(K_1$ fournie était identique à la seconde clef de la serrure $\implies$ la première clef de la serrure devient K_1 et une seconde clef pour la serrure$\big)$
$\vee$ la porte ne s'ouvre pas

basse — Supprimer une serrure

2 Préparation des tests de validation

Pour les cas d'utilisation de priorité haute et les plus « intéressants » car les plus sensibles ou risqués, nous préparons des tests de validation.

Numéro de test	1	2	3	4
Identifiant de la serrure bien formé (non null, non vide)	F	T	T	T
Graine de la serrure bien formé (non null, non vide)		F	T	T
Serrure avec cet identifiant inexistante			F	T
Création acceptée	F	F	F	T
Nombre de jeux de test	2	2	1	1

Table 1 – Cas d'utilisation « créer une serrure »

Pour le calcul des jeux de test des tests 4 et 5 du cas d'utilisation « tester une serrure avec deux clefs », considérons tout d'abord l'invariant suivant qui se déduit du texte du brevet : les

Numéro de test	1	2	3	4	5
Identifiant/code de la serrure bien formé (non null et non vide)	F	T	T	T	T
Serrure existante avec ce code		F	T	T	T
Nouvelle graine pour la génération des clefs de la chambre bien formée (non null et non vide)			F	T	T
Nouvelle graine ou nouveau sel différents				F	T
Génération de la nouvelle paire de clefs acceptée	F	F	F	F	T
Nouvelle graine $\neq$ ancienne graine $\vee$ nouveau sel $\neq$ ancien sel					T
Première clef de la nouvelle paire $\neq$ première clef de l'ancienne paire $\wedge$ première clef de la nouvelle paire $\neq$ seconde clef de l'ancienne paire					T
Seconde clef de la nouvelle paire $\neq$ première clef de l'ancienne paire $\wedge$ seconde clef de la nouvelle paire $\neq$ seconde clef de l'ancienne paire					T
Nombre de jeux de test	2	1	2	1	3

TABLE 2 – Cas d'utilisation « ré-initialiser les clefs d'une serrure ». Le jeu de tests du test 5 comprend trois tests : (1) graine différente, (2) sel différent, et (3) graine et sel différents.

Numéro de test	1	2	3	4	5
Identifiant de la serrure bien formé (non null, non vide)	F	T	T	T	T
Serrure avec cet identifiant existante		F	T	T	T
Deux clefs bien formées (non null, non vides)			F	T	T
(K_1 et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure) $\vee$ (K_1 fournie est identique à la seconde clef de la serrure)				F	T
La porte peut s'ouvrir	F	F	F	F	T
Nombre de jeux de test	2	1	4	5	3

TABLE 3 – Cas d'utilisation « tester une serrure avec deux clefs »

deux clefs de la serrure ne sont pas identiques. En considérant que l'invariant est vérifié (c.-à-d., nous nous engageons dans la suite à vérifier par assertion cet invariant dans le code), la table 3 présente toutes les combinaisons possibles entre les clefs fournies (K_1 et K_2) et celles de la serrure (KS_1 et KS_2).

#	**Conditions**				**Résultat ou raison d'impossibilité du test**
	$K_1 = KS_1$	$K_2 = KS_2$	$K_1 = KS_2$	$K_2 = KS_1$	Impossible car non-respect de l'invariant de la serrure
	"	"	"	$K_2 \neq KS_1$	Impossible car contradiction ($K_2 = KS_2 = K_1 = KS_1 \neq K_2$)
	"	"	$K_1 \neq KS_2$	$K_2 = KS_1$	Impossible car contradiction ($K_1 = KS_1 = K_2 = KS_2 \neq K_1$)
	"	"	"	$K_2 \neq KS_1$	Impossible car non-respect de l'invariant de la serrure
	"	$K_2 \neq KS_2$	$K_1 = KS_2$	$K_2 = KS_1$	Impossible car contradiction ($K_2 = KS_1 = K_1 = KS_2 \neq K_2$)
	"	"	"	$K_2 \neq KS_1$	Impossible car non-respect de l'invariant de la serrure
F1	"	"	$K_1 \neq KS_2$	$K_2 = KS_1$	*false*
F2	"	"	"	$K_2 \neq KS_1$	*false*
	$K_1 \neq KS_1$	$K_2 = KS_2$	$K_1 = KS_2$	$K_2 = KS_1$	Impossible car contradiction ($K_1 = KS_2 = K_2 = KS_1 \neq K_1$)
T1	"	"	"	$K_2 \neq KS_1$	*true*
	"	"	$K_1 \neq KS_2$	$K_2 = KS_1$	Impossible car non-respect de l'invariant de la serrure
F3	"	"	"	$K_2 \neq KS_1$	*false*
T2	"	$K_2 \neq KS_2$	$K_1 = KS_2$	$K_2 = KS_1$	*true*
T3	"	"	"	$K_2 \neq KS_1$	*true*
F4	"	"	$K_1 \neq KS_2$	$K_2 = KS_1$	*false*
F5	"	"	"	$K_2 \neq KS_1$	*false*

TABLE 4 – Jeux de test pour l'exhaustivité des test des cas d'utilisation « tester une serrure avec deux clefs » et « essayer d'ouvrir la porte avec un badge » (« FX » et « TX » servent à numéroter les tests ; « " » signifie « *idem* »)

Numéro de test	1	2	3	4	5
Identifiant de la serrure bien formé (non null, non vide)	F	T	T	T	T
Serrure avec cet identifiant existante		F	T	T	T
Deux clefs bien formées (non null, non vides)			F	T	T
(K_1 et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure) $\vee$ (K_1 fournie est identique à la seconde clef de la serrure)				F	T
$\big(K_1$ et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure $\implies$ la porte est ouverte et les clefs de la serrure reste inchangées$\big)$ $\vee\big(K_1$ fournie était identique à la seconde clef de la serrure $\implies$ la première clef de la serrure devient K_1 et une seconde clef est générée pour la serrure$\big)$ $\vee$ la porte ne s'ouvre pas	F	F	F	F	T
Nombre de jeux de test	2	1	2	5	3

TABLE 5 – Cas d'utilisation « essayer d'ouvrir la porte avec un badge »

3 Conception

3.1 Diagrammes de séquence

Le diagramme de classes de la figure qui suit contient la façade du système : la classe GestionSerrures, ainsi que la classe Serrures qui réalise le concept de serrure programmable avec deux clefs.

FIGURE 2 – Diagramme de classes

3.2 Diagrammes de séquence

Voici la description textuelle du cas d'utilisation « créer une serrure » :

— arguments en entrée : identifiant de la serrure, la graine de la serrure pour la génération des clefs, le sel de la serrure pour la génération des clefs

— rappel de la précondition : identifiant de la serrure bien formé (non null et non vide) ∧ graine bien formée (non null et non vide) ∧ serrure avec cet identifiant inexistante

— algorithme :

1. vérifier les arguments
2. vérifier que la serrure est inexistante
3. instancier la serrure
   — générer deux nouvelles clefs
4. ajouter la serrure dans la collection des serrures

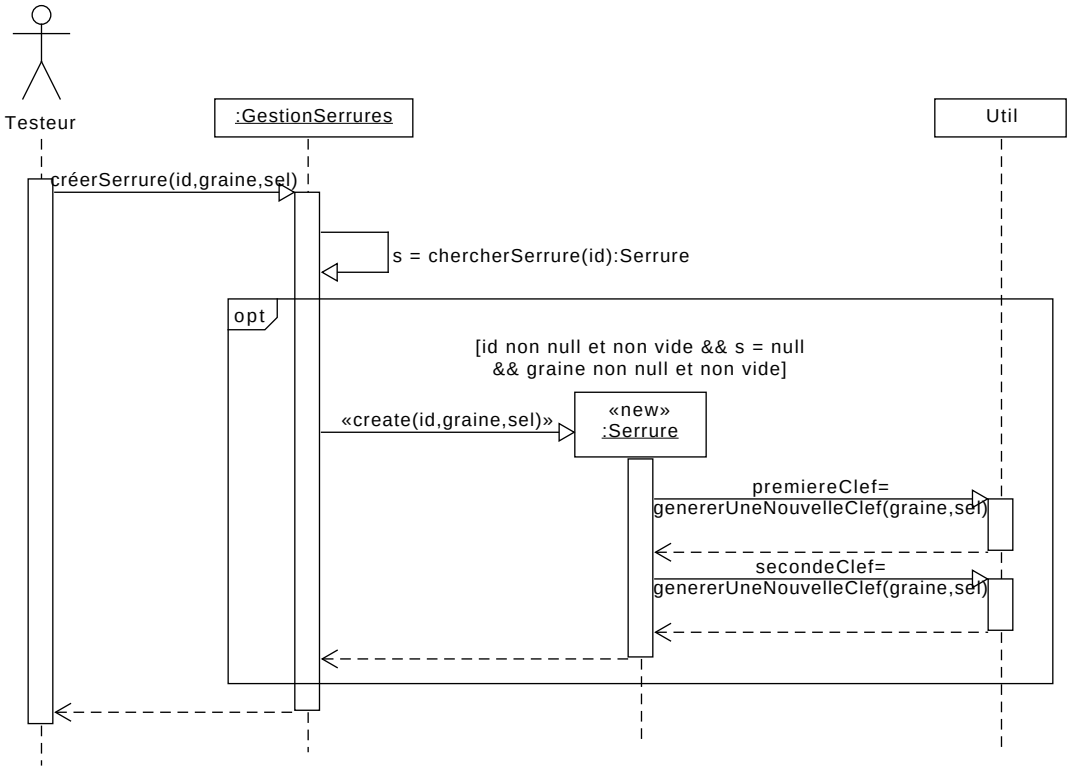

FIGURE 3 – Diagramme de séquence du cas d'utilisation « créer une serrure »

Voici la description textuelle du cas d'utilisation « ré-initialiser une serrure » :

— arguments en entrée : identifiant de la serrure, nouvelle graine, nouveau sel
— rappel de la précondition : identifiant de la serrure bien formé (non null et non vide) ∧ nouvelle graine bien formée (non null et non vide) ∧ serrure avec cet identifiant existante
— algorithme :
  1. vérifier les arguments
  2. vérifier que la serrure existe
  3. ré-initialiser la serrure
     — générer deux nouvelles clefs

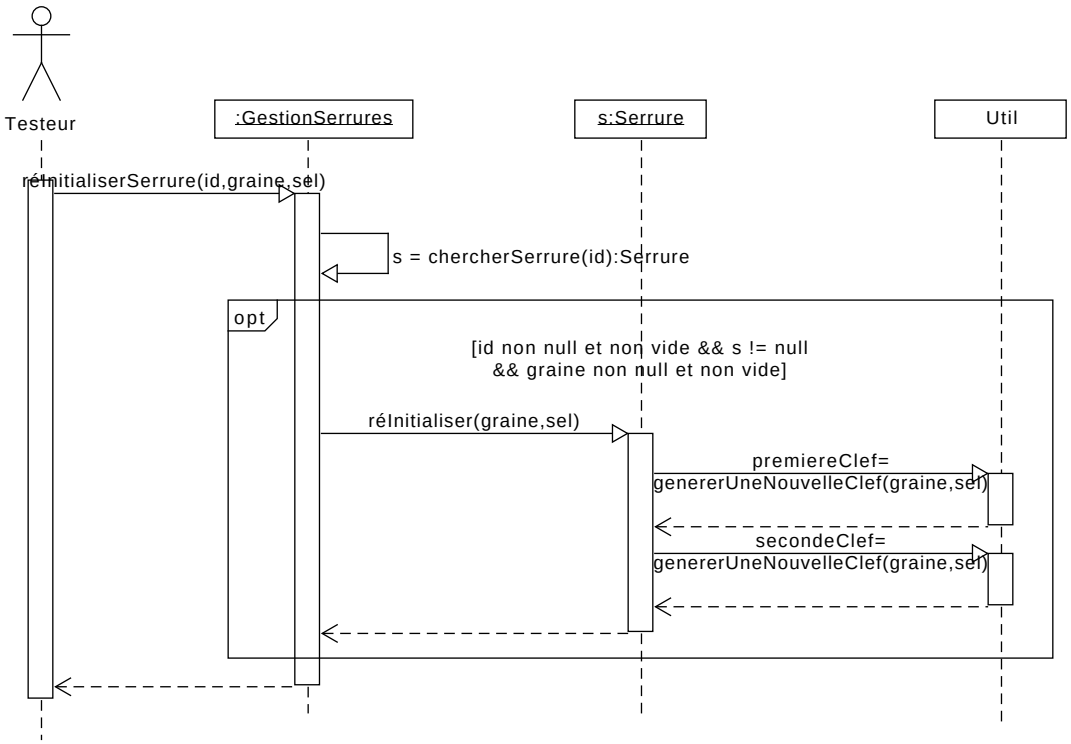

FIGURE 4 – Diagramme de séquence du cas d'utilisation « ré-initialiser une serrure »

Voici la description textuelle du cas d'utilisation « tester une serrure avec deux clefs » :

— arguments en entrée : identifiant de la serrure, première clef, seconde clef
— rappel de la précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ valeurs fournies pour les clefs K_1 et K_2 bien formées (non null et non vides)
— algorithme :
  1. vérifier les arguments
  2. vérifier que la serrure existe
  3. tester la serrure

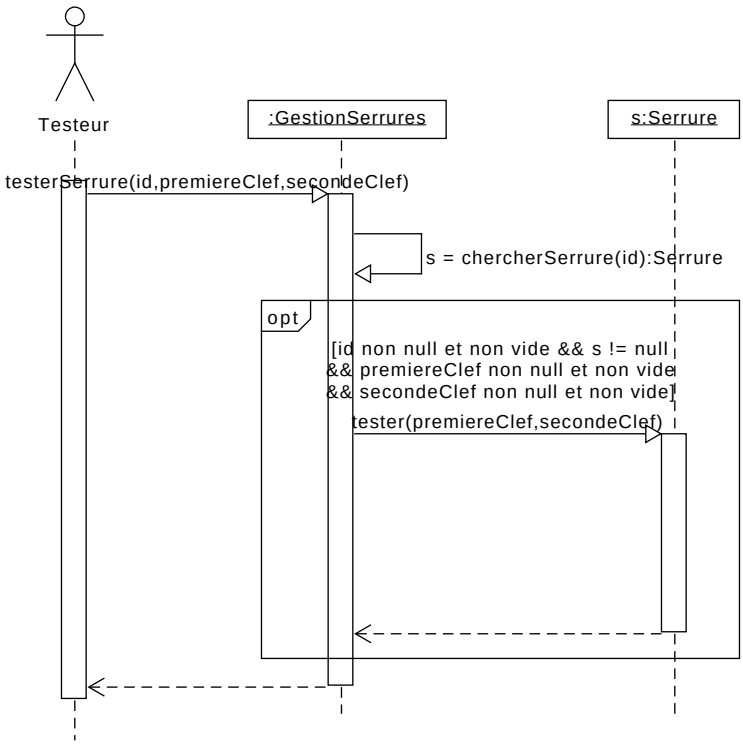

FIGURE 5 – Diagramme de séquence du cas d'utilisation « tester une serrure avec deux clefs »

Voici la description textuelle du cas d'utilisation « essayer d'ouvrir une serrure avec deux clefs » :

- arguments en entrée : identifiant de la serrure, première clef, seconde clef
- rappel de la précondition : identifiant de la serrure bien formé (non null et non vide) $\wedge$ serrure avec cet identifiant existante $\wedge$ valeurs fournies pour les clefs K_1 et K_2 bien formées (non null et non vides)
- algorithme :
  1. vérifier les arguments
  2. vérifier que la serrure existe
  3. essayer d'ouvrir la serrure
     - générer une nouvelle clef

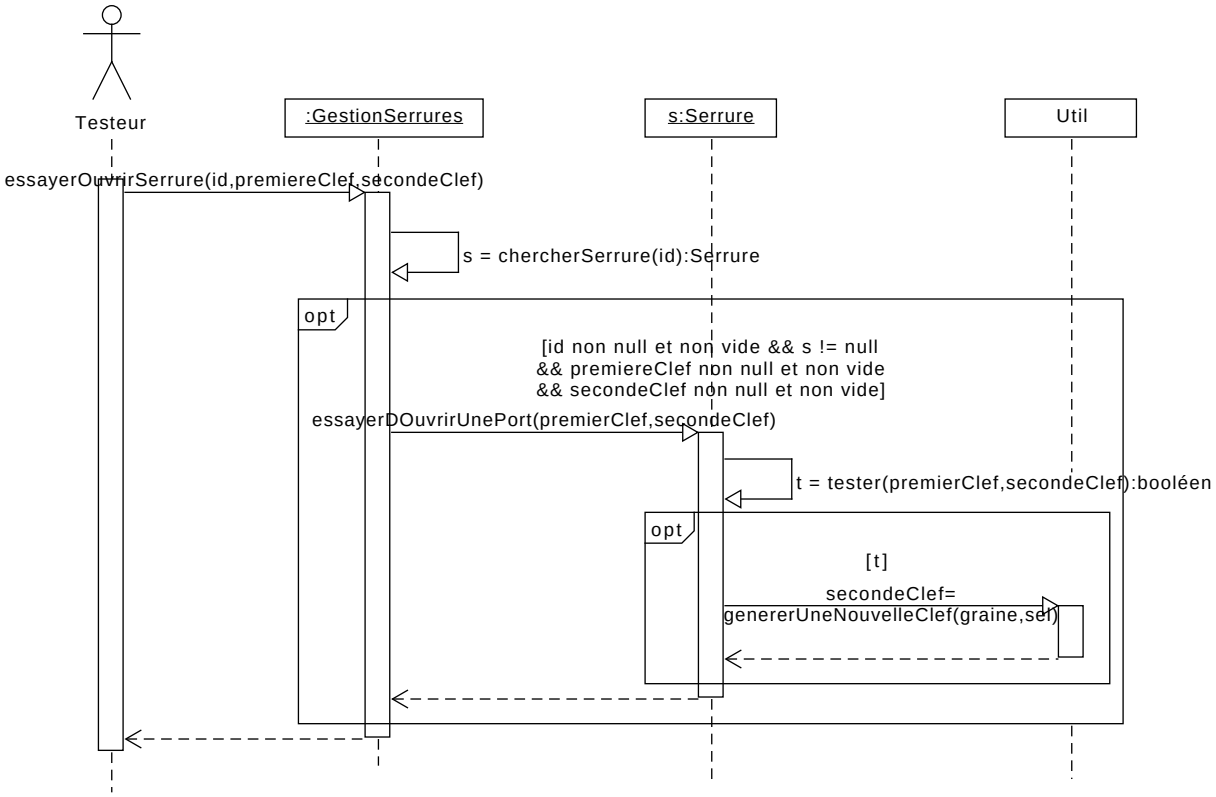

FIGURE 6 – Diagramme de séquence du cas d'utilisation « essayer d'ouvrir une serrure avec deux clefs »

3.3 Fiches des classes

3.3.1 Classe GestionSerrures

GestionSerrures
+ MESSAGE_PORTE_OUVERT : String = “la porte est ouverte” + MESSAGE_PORTE_FERMÉE : String = “la porte reste fermée” <– **attributs « association »** –> – serrures : collection de @Serrure
<– **constructeur** –> + GestionSerrures () + invariant() :booléen <– **operations « cas d’utilisation »** –> + créerSerrure(String identifiant, String graine, int sel) + essayerDOuvrirUnePorte(String identifiant, byte[] premièreClef, byte[] secondeClef) : String + invariant() : booléen + listeSerrures() : String[] + obtenirPremiereClefSerrure(String identifiant) : byte[] + obtenirSecondeClefSerrure(String identifiant) : byte[] + tester(String identifiant, byte[] premièreClef, byte[] secondeClef) :booléen – chercherSerrure(String identifiant) : Serrure

3.3.2 Classe Serrure

Serrure
– identifiant : String – graine : String – sel : int – premiereClef : byte[] – secondeClef : byte[]
<– **constructeur** –> + Serrure(String identifiant, String graine, int sel) + invariant() :booléen + essayerDOuvrirUnePorte(byte[] premièreClef, byte[] secondeClef) : String + invariant() : booléen + getPremiereClef() : byte[] + getSecondeClef() : byte[] + tester(byte[] premièreClef, byte[] secondeClef) :booléen

4 Préparation des tests unitaires

4.1 Classe Serrure

Numéro de test	1	2	3
`identifiant` $\neq$ null $\wedge \neg$vide	F	T	T
`graine` $\neq$ null $\wedge \neg$vide		F	T
identifiant$'$ = identifiant			T
graine$'$ = graine			T
sel$'$ = sel			T
premiereClef$'$ $\neq$ null$\neg$vide			T
secondeClef$'$ $\neq$ null$\neg$vide			T
premiereClef$'$ $\neq$ secondeClef			T
invariant			T
Levée d'une exception	OUI	OUI	NON
Objet créé	F	F	T
Nombre de jeux de test	2	2	1

TABLE 8 – Méthode `constructeurSerrure` de la classe `Serrure` »

Numéro de test	1	2	3
Deux clefs bien formées (non null, non vides)	F	T	T
(K_1 et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure) $\vee$ (K_1 fournie est identique à la seconde clef de la serrure)		F	T
premiereClef$'$ = premiereClef			T
secondeClef$'$ = secondeClef			T
invariant			T
Levée d'une exception	OUI	OUI	NON
La porte peut s'ouvrire	F	F	T
Nombre de jeux de test	4	5	3

TABLE 9 – Méthode `tester` de la classe `Serrure` ». Nous utilisons la même table pour calculer le nombre de tests dans les jeux de test des tests 2 et 3 : 5 combinaisons possibles pour false et 3 combinaisons possibles pour true.

Numéro de test	1	2	3
Deux clefs bien formées (non null, non vides)	F	T	T
(K_1 et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure) $\vee$ (K_1 fournie est identique à la seconde clef de la serrure)		F	T
$\big($$K_1$ et K_2 fournies identiques respectivement aux première et seconde clefs de la serrure $\implies$ la porte est ouverte et les clefs de la serrure reste inchangées$\big)$ $\vee\big($$K_1$ fournie était identique à la seconde clef de la serrure $\implies$ la première clef de la serrure devient K_1 et une seconde clef est générée pour la serrure$\big)$ $\vee$ la porte ne s'ouvre pas	F	F	T
Nombre de jeux de test	2	5	3

TABLE 10 – Méthode `essayerDOuvrirLaPorte` de la classe `Serrure` ». Nous utilisons la même table pour calculer le nombre de tests dans les jeux de test des tests 2 et 3 : 5 combinaisons possibles pour false et 3 combinaisons possibles pour true.

Numéro de test	1	2	3
`graine` $\neq$ null $\wedge \neg$vide	F	T	T
`graine` ou `sel` différents		F	T
identifiant$'$ = identifiant		F	T
graine$'$ $\neq$ graine $\vee$ sel$'$ $\neq$ sel			T
premiereClef$'$ $\neq$ premiereClef			T
premiereClef$'$ $\neq$ secondeClef			T
secondeClef$'$ $\neq$ premiereClef			T
secondeClef$'$ $\neq$ secondeClef			T
premiereClef$'$ $\neq$ secondeClef$'$			T
invariant			T
Levée d'une exception	OUI	OUI	NON
Clefs ré-initialisées	F	F	T
Nombre de jeux de test	2	1	3

TABLE 11 – Méthode `réInitialiser` de la classe `Serrure` ». Le test 3 comporte un jeu de trois tests selon que la graine ou le sel sont différents.