Devoir de spécification et validation du logiciel

Durée totale : **3h**. Tous documents autorisés. Nombre de pages : **3**.
Barème indicatif : moitié test, moitié model-checking.

Exercice 1 : Test

On reprend le jeu du non-merci vu en projet. Le fonctionnement du jeu est rappelé plus bas. Vous pouvez répondre aux questions dans l'ordre qui vous convient. À vous d'écrire les tests les plus simples possibles en réfléchissant notamment aux objets du domaine, à la répartition des responsabilités et à l'architecture de votre application. Tous les tests demandés sont des **tests unitaires en isolation écrits en JUnit4**, qui utilisent Mockito quand c'est nécessaire. **La présentation devra être soignée**.

Q 1.1 : Donner, à l'exclusion de toute autre chose :

- des cas de tests pour les fonctionnalités ou scénarios suivants, regroupés dans les classes de test qui vous semblent pertinentes au regard de votre conception (les classes sont à donner) :
  1. fonctionnalité : calcul du nombre de points associé à un tas de cartes ;
  2. fonctionnalité : tirage aléatoire du joueur qui commence ;
  3. scénario : le joueur courant prend la carte ;
  4. scénario : le joueur courant ne prend pas la carte.
- un diagramme UML de votre application listant les signatures des méthodes pour chaque classe ou interface Java utilisée **dans vos tests** (optique TDD). **Ce diagramme a uniquement pour but de vous aider à structurer vos idées. L'évaluation porte sur les tests.**

□

Idée du jeu D'après `http://www.gigamic.com/images/Anleitung_Geschenkt_GB.pdf` Ce jeu de cartes pour 3 à 5 joueurs se joue avec 33 cartes numérotées de 3 à 35 et 55 jetons.

Une carte est au centre de la table, face visible. Quand c'est à son tour de jouer, le joueur doit décider :

- soit de prendre la carte et de la poser devant lui — sachant que cette carte lui apportera des points négatifs ;
- soit de dire "non-merci" pour refuser de prendre la carte, auquel cas il doit payer avec un de ses jetons qu'il pose à côté de la carte — sachant que posséder un jeton rapporte un point. Le joueur suivant dans le sens des aiguilles d'une montre devra alors prendre une décision similaire : soit prendre la carte (avec le(s) jeton(s) associé(s)), soit dire non-merci et ajouter un jeton au tas qui se forme à côté de la carte.

À l'issue de la partie, le joueur qui a le moins de points négatifs gagne (celui qui a le plus de points, donc).

Valeur des cartes et des jetons Prise isolément, une carte apporte sa valeur en points négatifs (par exemple, la carte 20 apporte -20 points au joueur qui la possède). Mais les cartes peuvent être ordonnées en suites continues, auquel cas une suite apporte la valeur de sa plus petite carte en points négatifs (par exemple la suite 17, 18, 19 apporte -17 points, la suite 6, 7 apporte -6 points).
Chaque jeton possédé par le joueur lui apporte un point (positif).

Préparation Chaque joueur prend 11 jetons (les jetons qui restent éventuellement ne seront pas utilisés au cours de la partie). Ensuite on bat les cartes et on en extrait un tas de 24 cartes face cachées, posé au centre de la table. Les 9 cartes restantes ne seront pas utilisées au cours de la partie.

Déroulement du jeu On tire au sort le joueur qui commence. Il retourne la carte en sommet de pioche et la pose à côté, face visible. Il a alors le choix entre prendre la carte et la poser devant lui, face visible, ou de refuser la carte en payant avec un de ses jetons. Si le joueur refuse la carte, le tour passe au joueur suivant dans le sens des aiguilles d'une montre. Chaque joueur à son tour a le choix entre prendre la carte et tous les jetons qui sont à côté, ou la refuser en payant avec un de ses jetons.

Quand finalement un joueur décide de prendre la carte et tous les jetons qui sont à côté, il pose la carte face visible à côté de lui. Il retourne alors le sommet de la pioche et pose cette nouvelle carte face visible à côté du tas. Il a maintenant le choix entre la prendre ou la refuser, et garde la main tant qu'il décide de prendre.

Fin du jeu et scores Le jeu termine quand la pioche est vide. On compte alors les points associés aux suites de cartes et aux jetons. Le joueur qui a le plus de points a gagné (-12 gagne sur -32).

Exercice 2 : Model-checking

On s'intéresse à un protocole de transmission de données, dans 3 versions différentes ($protocole_1$, $protocole_2$, $protocole_3$). Le fonctionnement du protocole est, en boucle : initialisation de la donnée courante, tentative de transmission de cette donnée, puis, selon que la donnée a bien été transmise ou non, action propre à chaque protocole.

On utilisera les variables propositionnelles de l'ensemble $\mathcal{P} = \{$ `init`, `essai`, `ok` $\}$. `init` est vrai uniquement dans l'état d'initialisation de la donnée courante. `essai` est vrai uniquement dans l'état dans lequel le protocole tente de transmettre la donnée courante. `ok` est vrai uniquement quand la transmission a réussi (`ok` est donc faux tant que la transmission n'a pas encore été testée ou quand le test de transmission indique que la transmission de la donnée courante a échoué).

Les protocoles à étudier sont les suivants :
- $protocole_1$: ce protocole fonctionne dans un environnement idéal dans lequel toute tentative de transmission réussit. Le protocole enchaîne donc en boucle la phase d'initialisation, la tentative de transmission, et la finalisation de la transmission qui vérifie que la donnée a bien été transmise.
- $protocole_2$: même fonctionnement que $protocole_1$ mais la transmission peut échouer. Ce protocole fonctionne aussi en boucle, mais après la phase d'initialisation et la tentive de transmission, le protocole fait un choix : soit la donnée a bien été transmise et la transmission est finalisée, soit elle ne l'a pas été et alors on abandonne sa transmission ; dans les 2 cas le protocole retourne en phase d'initialisation.
- $protocole_3$: même fonctionnement que $protocole_2$ mais, si la donnée n'a pas été transmise avec succès, alors le protocole essaie à nouveau de la transmettre au lieu de d'abandonner la transmission.

Dans les modèles de protocole, vous nommerez chaque état, et l'annoterez avec l'ensemble des variables propositionnelles de $\mathcal{P}$ vraies en cet état, les autres étant par convention fausses dans cet état. Ainsi, si par exemple dans un état *fail* les variables `init`, `essai` et `ok` sont fausses, l'état *fail* n'a aucune étiquette. Si dans un état *success* seule la variable `ok` est vraie, alors *success* est étiqueté avec `ok` uniquement.

Les questions 2.2, 2.3 et 2.5 dépendent de la question 2.1.

Q 2.1 : Donner trois automates $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3$ qui modélisent respectivement les comportements des protocoles $protocole_1$, $protocole_2$ et $protocole_3$. □

Q 2.2 : Donner un processus Spin qui a le même comportement que $\mathcal{A}_1$. □

Q 2.3 : Donner sous la forme d'ω-expressions régulières décrivant des exécutions infinies, et en utilisant les noms d'état que vous avez choisi :
- l'unique exécution de $\mathcal{A}_1$;
- deux exécutions de $\mathcal{A}_2$ qui ne sont pas possibles pour le $protocole_1$;
- deux exécutions de $\mathcal{A}_3$ qui ne sont possibles ni pour le $protocole_1$, ni pour le $protocole_2$.

□

Q 2.4 : En utilisant les variables de $\mathcal{P}$, exprimer en LTL les propriétés suivantes :
- ϕ_1 : le protocole démarre en phase d'initialisation ;
- ϕ_2 : à chaque fois que la transmission a réussi, le protocole sera en phase d'initialisation l'instant d'après ;
- ϕ_3 : la transmission réussira un jour ;
- ϕ_4 : ϕ_3 est fausse ;
- ϕ_5 : la phase d'initialisation a lieu infiniment souvent.
- ϕ_6 : la phase d'initialisation et de tentative de transmission n'ont jamais lieu en même temps.
- ϕ_7 : la transmission réussira un jour, et jusqu'à ce qu'elle réussisse le protocole sera en phase d'initialisation ou de transmission ;
- ϕ_8 : à partir d'un certain instant dans le futur, le protocole ne sera plus jamais en état de transmission réussie.

□

Q 2. 5 : Pour chaque formule donnée à la question 2.4 et chaque modèle de protocole, dites si l'automate satisfait la formule et justifier **brièvement** votre réponse, soit par un contre-exemple, soit par une explication rigoureusement basée sur l'automate. □

Q 2. 6 : Donner une formule CTL* ϕ_9 qui exprime : "il existe une exécution dans laquelle jamais aucune transmission ne réussit". Dire pour chaque modèle de protocole si cette formule est vérifiée, en justifiant votre réponse. □

Q 2. 7 : Parmi les formules ϕ_1 à ϕ_9, donner :
- une formule de sûreté ;
- une formule de vivacité ;
- une formule d'équité.

□