IUT Sénart-Fontainebleau, UPEC
DEV 2.3
IUT Info BUT1

Florent Madelaine
Luc Hernandez
2023-24

Programmation défensive et tests unitaires

Notions importantes

- boîte noire
- boîte blanche
- programmation par contrat
- précondition
- postcondition
- invariant
- assertion
- (vers les) tests unitaires

Exercices

Exercice 1. On se place dans le cadre du *jeu de memory* (decription de wikipedia reproduite ci-dessous).

> Le jeu se compose de paires de cartes portant des illustrations identiques. L'ensemble des cartes est mélangé, puis étalé face contre table. À son tour, chaque joueur retourne deux cartes de son choix. S'il découvre deux cartes identiques, il les ramasse et les conserve, ce qui lui permet de rejouer. Si les cartes ne sont pas identiques, il les retourne faces cachées à leur emplacement de départ.
>
> Le jeu se termine quand toutes les paires de cartes ont été découvertes et ramassées. Le gagnant est le joueur qui possède le plus de paires.

1. On se place dans le cadre d'une modélisation du M de MVC, en particulier les aspects gestion de la sauvegarde de l'état du jeu. Réfléchissez rapidement aux classes possibles et à leur rôle (Si vous ne trouvez pas de nom clair pour une classe c'est qu'elle essaye de faire trop de choses).
2. Quelles sont les préconditions et postconditions de la méthode correspondant à l'action de retourner 2 cartes d'un joueur ? (on suppose que cette méthode est très bête elle ne fait que retourner les cartes. Une autre méthode se chargera de vérifier si les cartes sont égales et de faire un déplacement, probablement dans une autre classe que celle des cartes).
3. Même question pour cacher 2 cartes
4. Quelles propriétés sont invariantes par l'une et l'autre de ces méthodes ?
5. En supposant qu'on ait une classe Carte, quel invariant envisagez vous pour cette classe ?

1. Il faut une classe carte qui sont cachées ou visibles. (un type énuméré peut servir pour spécifier ces deux états). Une carte possède une valeur (par exemple un entier, qui permet de retrouver le nom du fichier d'illustration). On peut comparer deux cartes pour savoir si elles sont égales (de même valeur).
   Il faut aussi une classe permettant de contenir des cartes (probablement abstraite).
   Il y aura deux classes concrètes en héritant : la classe table et la classe (ensemble de carte d'un) joueur. On peut révéler ou cacher une carte sur la table (au plus deux cartes révélées). On peut connaître le score d'un joueur (nombre de paires).
   Il faut un mécanisme pour déplacer les cartes d'un ensemble de carte à un autre (en pratique de la table vers un joueur pendant la partie et inversement en fin de partie).
   Pour l'instant pour simplifier on ne gère pas la modélisation physique du jeu (par exemple disposition sous forme de tableau). On pourra supposer que c'est la partie vue qui s'en charge.
2. La précondition de la méthode révéler est que la carte est cachée et que au plus 1 carte est déjà révélée. La postcondition est qu'au plus 2 cartes sont révélées et que la carte est maintenant bien visible.

3. La méthode cacher est similaire. La carte doit être visible (précondition). Elle devient cachée (postcondition). Normalement il y a au plus deux cartes cachées avant (précondition) et au plus une carte cachée après (postcondition).
4. Le nombre de cartes visibles est 0, 1 ou 2.
5. On devrait avoir un nombre pair de cartes, et pour chaque valeur un nombre paire de cartes de cette valeur.

Exercice 2. On considère un algorithme de tri opérant sur un tableau d'entiers.

1. Quelle est la post-condition de l'algorithme ?
2. On suppose que l'algorithme utilisé est le suivant.

```
procédure tri-selection(tableau t)
      n := longueur(t)
      pour i de 0 à n - 2
          min := i
          pour j de i + 1 à n - 1
              si t[j] < t[min], alors min := j
          fin pour
          si min =/= i, alors échanger t[i] et t[min]
      fin pour
  fin procédure
```

   Proposez un invariant pour la boucle externe (celle de paramètre i).

1. postcondition : le tableau est trié. Pour tout index i strictement avant le dernier index, $t[i]$ est plus petit que l'entier suivant $t[i+1]$.
2. Invariant pour la boucle externe (de paramètre i) : le début du tableau est trié jusqu'à l'index i (NB. c'est par récurrence sur cette propriété qu'on démontre que l'algorithme est correct).

Exercice 3. On considère l'algorithme de recherche binaire d'un entier dans un tableau.

1. Expliquez en français l'idée de l'algorithme.
2. Donnez le pseudo-code
3. Quelle est la précondition de cet algorithme ?
4. Quelle propriété est vraie à chaque étape de l'algorithme ? (dans une boucle si vous donnez un algo avec une boucle, dans un appel récursif si vous utilisez cette alternative).