TD5

Exercice 1 :

Considérons une machine à café à modéliser (insérer pièce, si pas assez de monnaie alors message d'erreur, choisir café, donner café ou erreur). On suppose que la programmation est événementielle (un thread qui fonctionne, reçoit des événements et produit des observations). La machine est modélisable par une seule classe, composée de plusieurs méthodes et doit être au moins composée d'un paramètre EAU (TRUE FALSE) et d'un constructeur permettant de définir ce paramètre.

1. Définir les actions puis donner le diagramme d'état UML (en mettant l'accent sur la partie condition) et enfin le diagramme de classe.
2. Selon vous la machine, telle qu'elle est décrite, est-elle testable ? pourquoi ? Développons la question, avec l'étude de la testabilité.

"Software testability is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document) supports testing in a given test context. If the testability of the software artifact is high, then finding faults in the system (if it has any) by means of testing is easier." [wiki]

Un système est testable s'il est observable et contrôlable [Freedman 91]. D'autres propriétés peuvent aussi être considérées (propriété d'isolation, propriété temporelle, etc.)[salva01].

— Observability : The degree to which it is possible to observe (intermediate and final) test results.

Observabilité (pour boite noire avec entrée/sortie) : Pour toute entrée, il existe une sortie unique

— Controllability : The degree to which it is possible to control the state of the component under test (CUT) as required for testing.

Contrôlabilité (pour boite noire avec entrée/sortie) : Pour toute sortie, il existe une entrée unique qui donne cette sortie

Entrée : action fournie au système, modélisé avec ? -> ?insérerpièce() sortie : action observée depuis le système, modélisé avec ! -> !afficher

3. Modifiez votre machine avec ? et !. Avez-vous d'autres actions que " ?" et " !" ? Si oui comment tester ?
4. Votre machine est-elle observable ? A montrer pour chaque entrée. si non, modifiez votre machine.
5. Votre machine est-elle contrôlable ? A montrer pour chaque sortie. si non, modifiez votre machine.
6. Une autre propriété importante est le déterminisme : depuis un état, il n'existe pas 2 transitions avec la même action vers 2 états différents. Votre machine est-elle déterministe ? Si non, pourquoi n'est pas testable?

Voilà, vous avez une machine testable. Écrivons quelques tests en JUNIT.

Vous pouvez tester par exemple que : "si - de 0,5e sont donnés alors un message d'erreur est affiché", "si plus de 0,5e (par exemple 1e) sont donnés alors un café est mis en préparation et la monnaie est rendue", "si monnaie donnée et s'il n'y a plus d'eau alors il n'y a pas de café en préparation (message d'erreur)".

7. Donnez des exemples de test de robustesse à effectuer en JUNIT.
8. Donnez des exemples de test de robustesse à effectuer en JUNIT. Peut-on faire des tests de sécurité ? (quelques critères : availability, integrity, confidentiality, authorization, authentication)