



L'objectif de ce TD est de reprendre sur un exemple l'utilisation d'objets factices pour réaliser des tests d'intégration.

Le package *temp* sert à convertir des degrés Celsius en degrés Fahrenheit et vice-versa. Il comprend 2 classes :

aTester : la classe à vérifier qui contient en particulier

1. un attribut privé de type *Conversion*
2. une méthode *Double convertit(Double temperature, String sens)* qui prend une température, une chaîne qui est F2C ou C2F et retourne la conversion de température en Celsius ou Fahrenheit.

Conversion : la classe de service pour *aTester* qui contient

- une méthode *convF2C* qui convertit une température en Fahrenheit en Celsius (formule : $\text{temp}-32.0)*5.0/9.0$),
- une méthode *convC2F* qui convertit une température en Celsius en Fahrenheit (formule : $\text{temp}*9.0/5.0+32.0$).

Exercice 1 Réalisation des tests.

1. Donner une classe de test *atesterTest* pour *aTester* (quelques idées : $0^{\circ}\text{C}=32^{\circ}\text{F}$, $100^{\circ}\text{C}=212^{\circ}\text{F}$, $37^{\circ}\text{C}=98.6^{\circ}\text{F}$, $-40^{\circ}\text{C}=-40^{\circ}\text{F}$).
2. L'équipe chargée de l'écriture de *Conversion* prévient qu'elle n'est pas sûre des formules qu'elle a utilisées. Il est nécessaire de remplacer la classe *Conversion* par une autre classe qui permet de faire fonctionner les tests. Expliquer comment faire et donner la classe *MockConversion*. Donner le schéma d'héritage et de dépendance entre classes.

Exercice 2 Inversion du contrôle.

La solution précédente n'est pas satisfaisante. On va utiliser la méthode d'inversion du contrôle pour obtenir une approche plus générale.

1. Créer une Interface *ICConversion* qui permet de donner les fonctionnalités attendues de la classe *Conversion*. Que faut-il réécrire dans la classe *aTester* et dans la classe *Conversion*.
2. Donner une classe factice *MockConversion* qui permet d'implémenter l'interface *ICConversion* et de simuler le comportement attendu pour les tests. Donner le schéma d'héritage et de dépendance entre classes.
3. L'approche précédente nécessite d'écrire la classe *MockConversion*. Utiliser l'outil de génération de classes JMock et expliquer comment on peut écrire une classe de test sans avoir à générer explicitement la classe *MockConversion*. Donner la class *aTesterTest* correspondante.

Le package *robot* sert à contrôler le déplacement d'un robot dans une zone de surveillance identifiée avec le plan euclidien. Un GPS embarqué sur un capteur permet mesurer les coordonnées cartésiennes (abscisse, ordonnée) du robot dans le plan. Les coordonnées polaires correspondant aux coordonnées cartésiennes sont définies par un angle et un module. Le package comprend 2 classes :

Pilote : la classe à vérifier qui contient en particulier

1. un attribut privé de type *Capteur*,
2. deux attributs privés *thetamax* et *modmax* de type double,
3. une méthode *ModPol(double abs, double ord)* qui calcule le module correspondant à *abs* et *ord*,
4. une méthode *AngPol(double abs, double ord)* qui calcule l'angle correspondant à *abs* et *ord*,
5. une méthode *Boolean horsZone(Double module, String angle)* qui signale que le robot est hors zone si dans le système de coordonnées polaire l'angle est supérieur en valeur absolue à *thetamax* ou son module supérieur à *modmax*.

Capteur : la classe qui gère un capteur pour *Pilote* qui contient

- un attribut privé *horloge* de type double qui donne le temps courant *t*,
- une méthode *double getAbs()* qui calcule l'abscisse à l'instant *t*,
- une méthode *double getOrd()* qui calcule l'ordonnée à l'instant *t*.

Exercice 3 Réalisation des tests.

1. Donner une classe de test *PiloteTest* pour *Pilote* en supposant que la classe *Capteur* a été écrite.
2. L'équipe chargée de la réalisation de *Capteur* prévient qu'elle doit remplacer le modèle de capteur et réécrire la classe. Il est nécessaire de remplacer la classe *Capteur* par une autre classe qui permet de faire fonctionner les tests. Expliquer comment faire et donner la classe *MockCapteur*. Donner le schéma d'héritage et de dépendance entre classes.

Exercice 4 Inversion du contrôle.

La solution précédente n'est pas satisfaisante. On va utiliser la méthode d'inversion du contrôle pour obtenir une approche plus générale.

1. Créer une Interface *ICapteur* qui permet de donner les fonctionnalités attendues de la classe *Capteur*. Que faut-il réécrire dans la classe *Pilote* et dans la classe *Capteur*.
2. Donner une classe factice *MockCapteur* qui permet d'implémenter l'interface *ICapteur* et de simuler le comportement attendu pour les tests. Donner le schéma d'héritage et de dépendance entre classes.

3. L'approche précédente nécessite d'écrire la classe *MockCapteur*. Utiliser l'outil de génération de classes JMock et expliquer comment on peut écrire une classe de test sans avoir à générer explicitement la classe *MockCapteur*. Donner la classe *PiloteTest* correspondante.

Exercice 5 On change le comportement du capteur dont les méthodes retournent maintenant une exception *IllegalArgumentException* en cas de données incohérentes.

1. Modifier les tests du premier exercice pour tenir compte de ce changement.
2. Que faut-il modifier dans la classe factice de l'exercice 2.