IUT Nantes
Pôle Sciences et technologie

Nantes Université

R2.03 - Qualité de développement 1
Automatisation des tests

CM7 - Implémentation des tests de levées d'exceptions

Les exceptions

- Mécanisme permettant d'écrire des tests vérifiant la levée des exceptions
  - La levée des exceptions est positive : elle protège
  - Si un bug empêche la levée d'une exception, le programme risque
    - de planter,
    - de renvoyer de mauvais résultats.

- Le test des levées d'exception change à chaque version de Junit : ce sont des tests importants mais pas simples.

Test d'une méthode levant une exception

```
/**
 * Age humain d'un chien
 * @param theDog : Chien
 * @return ageHumain : flottant
 * @throws ArithmeticException
 */
fun ageHumain(theDog: Chien): int {
    if (theDog.age <= 0) {
        throw ArithmeticException("Un age est positif ou nul")
    }else {
        return theDog.age/12*7 //potentiellement faux
    }
}
```

La version basique
Calquée le test d'exception sur leur utilisation

```
@Test
fun testAgeHumainFail(){
    var lassie = Chien(nom:"Lassie", race:"yorkshire", mois: -4)
    try {
        chenil. ageHumain(lassie) //instancié précédemment
        //executed if exception not raised => test fails
        fail()
    }catch (e: ArithmeticException){
        //executed if exception raised as expected => test passes
    }
}
```

La version Junit 5 utilisant une assertion

```
import org.junit.jupiter.api.Test
import org.junit.jupiter.api.assertThrows

@Test
fun testAgeHumainAssert(){
    var lassie = Chien(nom:"Lassie", race:"yorkshire", mois: -4)
    assertThrows<ArithmeticException> {
            chenil. ageHumain(lassie) //instancié précédemment
    }
}
```

Attention aux levées de mauvaises d'exception

Si votre test vérifie la bonne levée d'une exception particulière, il ne faut pas qu'une autre soit lancée à la place.

Ce test est invalide, il vérifie à tort une mauvaise levée d'exception :

```
@Test
fun testAgeHumainAutreExcept(){
    var lassie = Chien(nom:"Lassie", race:"yorkshire", mois: -4)
    assertThrows<NomInapproprieException> {
            chenil. ageHumain(lassie) //instancié précédemment
    }
}
```

Attention à ne pas attendre de levées d'exception à tort (influencé par l'IDE qui le suggère)

```
@Test // ce test est invalide : on n'attend pas de levée d'exception
fun ageHumainFailBis(){
   var lassie = Chien(nom:"Lassie", race:"yorkshire", mois:24)
   try {
      chenil. ageHumain(lassie) //instancié précédemment
      //executed if exception not raised => test fails
      fail()
   }catch (e: ArithmeticException){
      //executed if exception raised as expected => test passes
   }
}

@Test // ce test nominal est valide : on n'attend pas de levée d'exception
fun ageHumainCorrect(){ // ce test est valide en étant nominal
   var lassie = Chien(nom:"Lassie", race:"yorkshire", mois:24)
   assertEquals(14, chenil. ageHumain(lassie))
}
```