



TP 2 : réaliser une campagne de tests automatisés

Printemps 2024





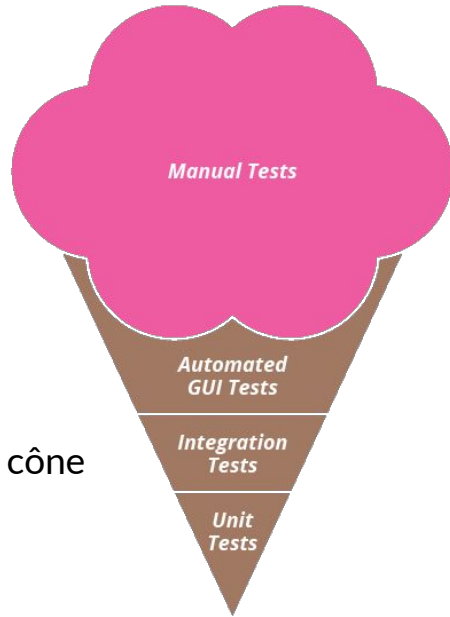
La phase de prototypage a été réalisée.

Le domaine a été développé.

Les contrôleurs ont été développés, puis intégrés aux boundaries.

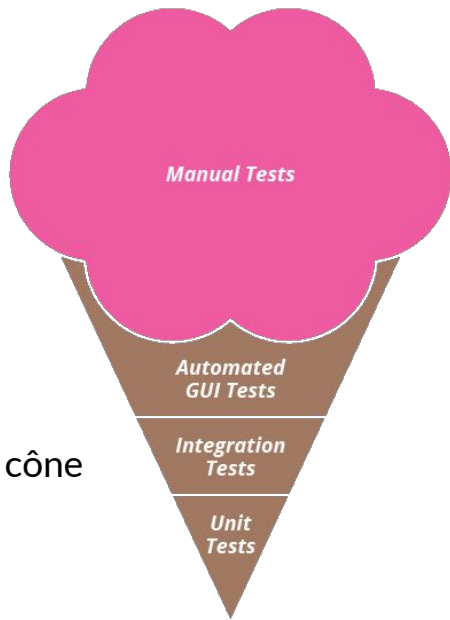
On a même lancé une campagne de ***spécification*** et de tests ***fonctionnels***.

Mais est-ce qu'on aurait pas (encore une fois) oublié quelque chose... ?

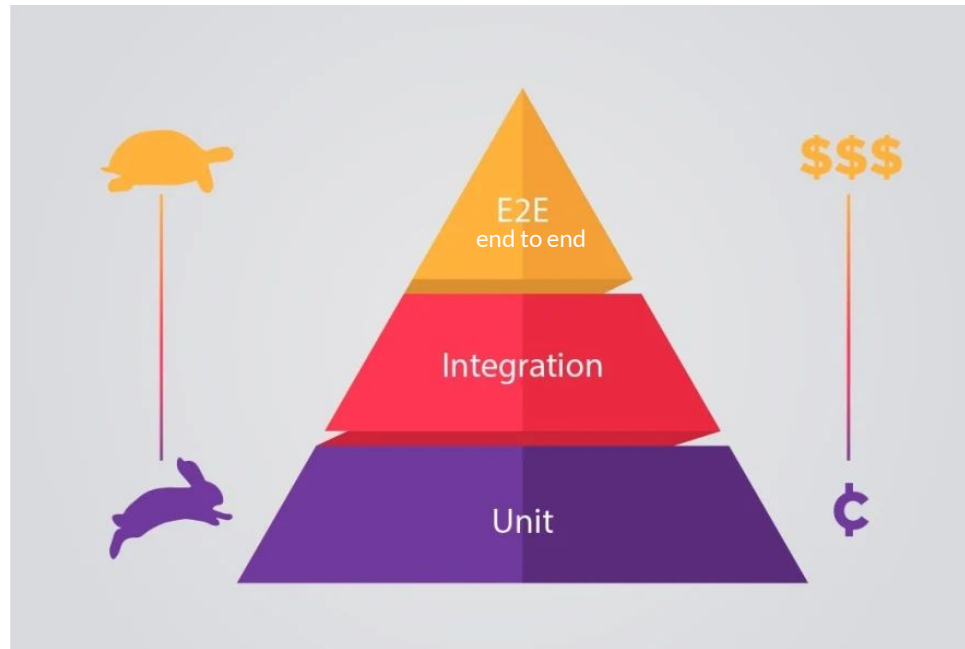


L'anti-pattern du cône de glace...

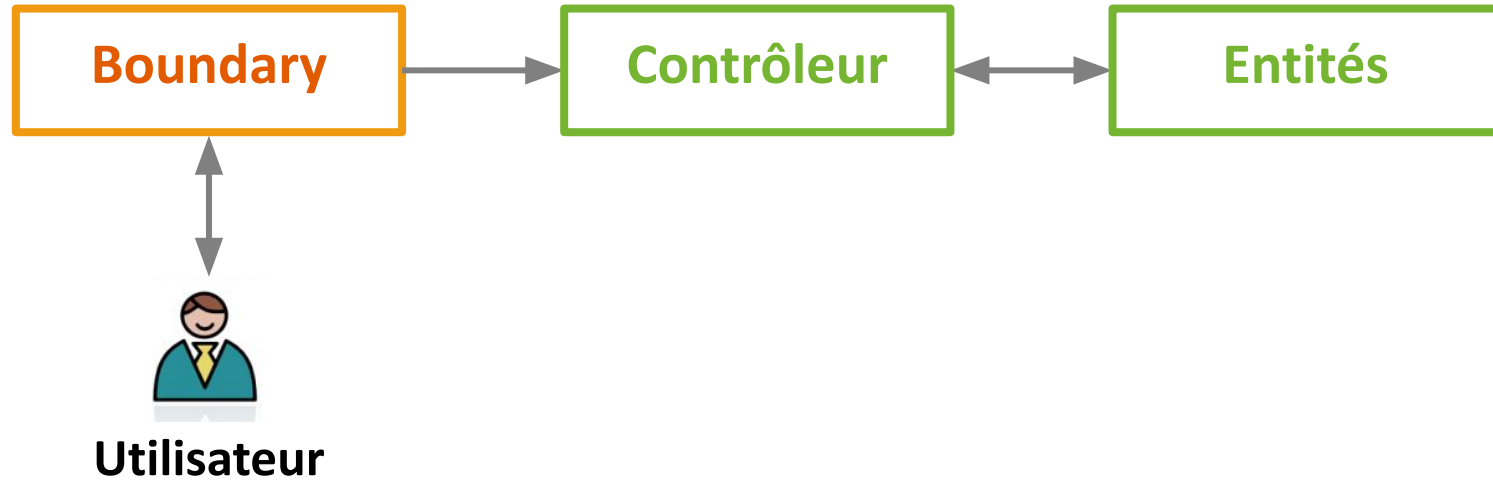
Mais est-ce qu'on aurait pas (encore une fois) oublié quelque chose... ?



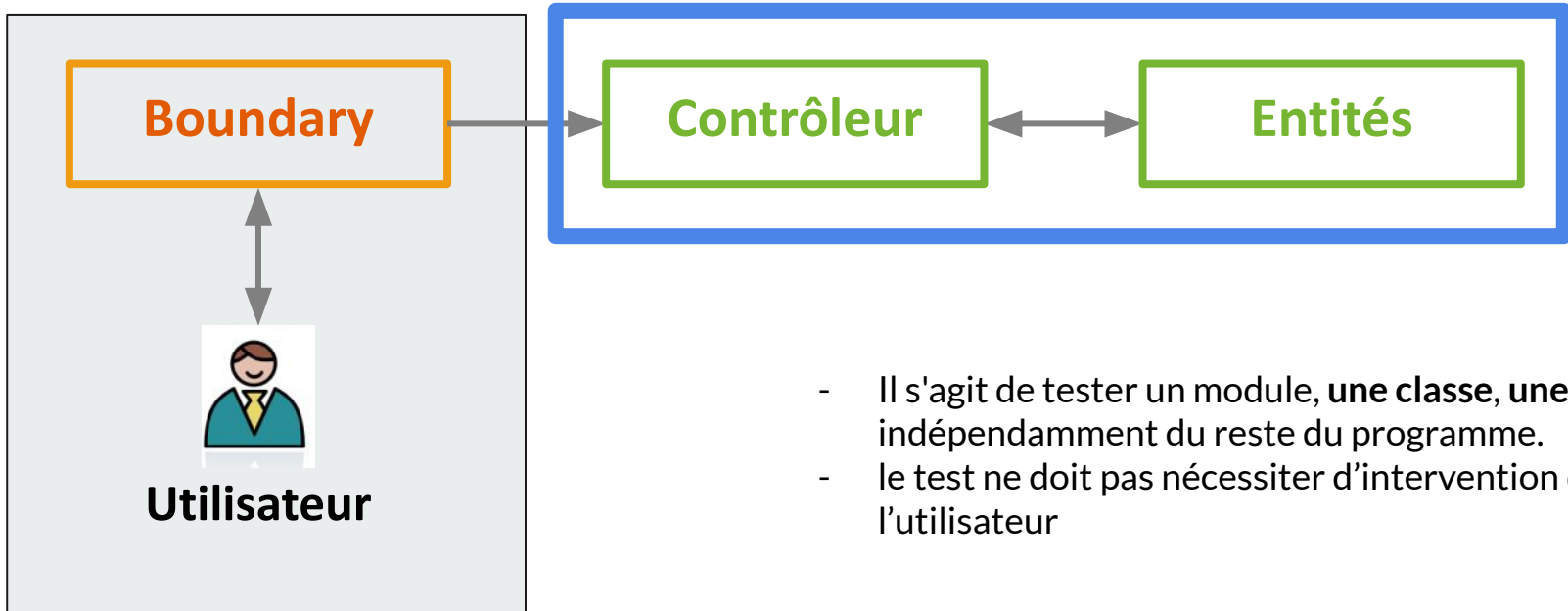
L'anti-pattern du cône de glace...



Notre architecture :



Que va-t-on tester unitairement ?



Retour sur le TP2 ILU2-POO



Chaque use case est testé...par du test fonctionnel

Bienvenue dans le village des irréductibles dirigé par le chef Abraracourcix.

Ce village possède un joli marché avec 5 étals mis à la disposition des villageois afin qu'ils puissent vendre leurs produits. Pour l'instant, le chef est bien seul dans son village.

Qui êtes-vous ?

- 1 - un voyageur
- 2 - un marchand
- 3 - un client du marché
- 4 - quitter l'application

1

Quel est votre nom ?

Bonemine

- 1 - je souhaite que vous me présentiez votre village.
- 2 - je voudrais emménager dans votre village.
- 3 - quitter l'application.

2

Êtes-vous :

- 1 - un druide.
- 2 - un gaulois.

2

Bienvenu villageois Bonemine

Quelle est votre force ?

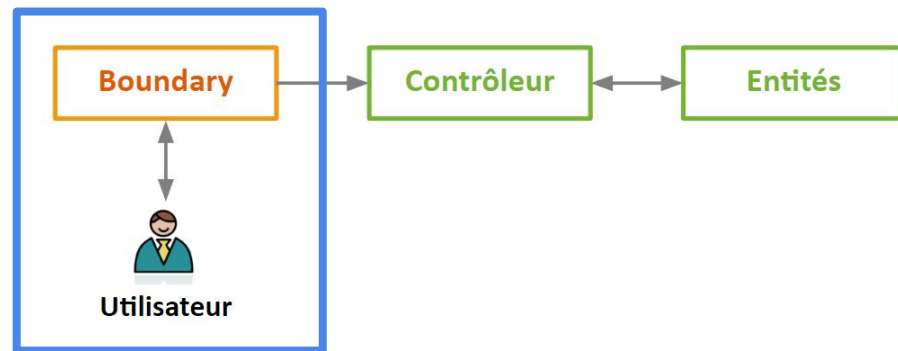
3

- 1 - je souhaite que vous me présentiez votre village.
- 2 - je voudrais emménager dans votre village.
- 3 - quitter l'application.

3

Au revoir voyageur Bonemine

Retour sur les pdf du TP2



Notre objectif est de transformer ce test fonctionnel en tests automatisés...

Etape 1 : mise en place du projet : 3 solutions

- Repartir du projet TP2
- Créer un nouveau projet sous github
- Copier / coller les sources du projet initial dans le nouveau
- Repartir du projet TP2
- Poser un tag pour identifier l'état initial
- Vos modifications s'ajoutent dans la branche "main" de votre projet initial
- Repartir du projet TP2, et faire une nouvelle branche dédiée à la mise en place des tests



solution 2 > Tag de l'état initial

Go to file Add file <> Code

8da7a5f 9 minutes ago 2 commits

9 minutes ago

9 minutes ago

9 minutes ago

14 minutes ago

About

No description, website,

Readme

0 stars

1 watching

1 fork

Releases

No releases published

[Create a new release](#)

Releases

Tags

Choose a tag

Target: main

Choose a tag

v1.0

+ Create new tag: v1.0 on publish

H B I <> @

Describe this release

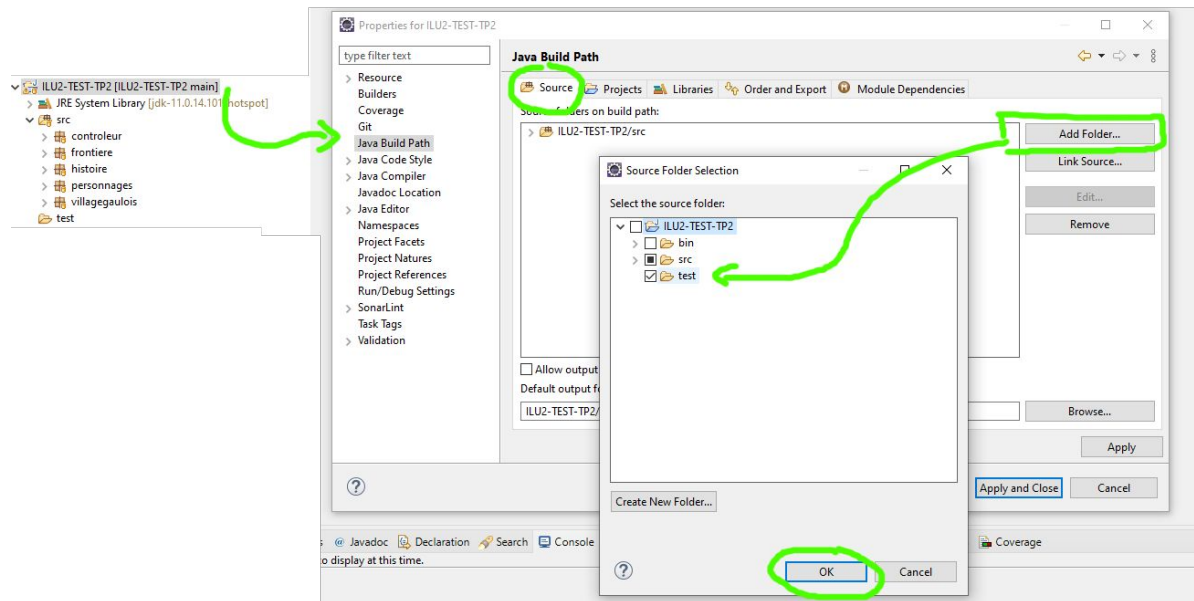
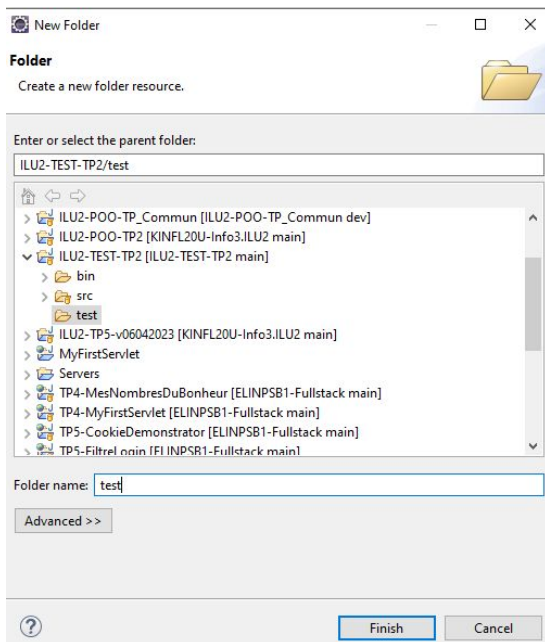
Si vous êtes sous Eclipse...

Félicitations, vous savez respecter une consigne.

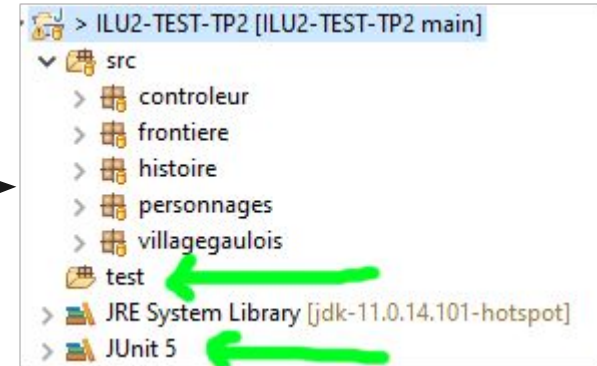
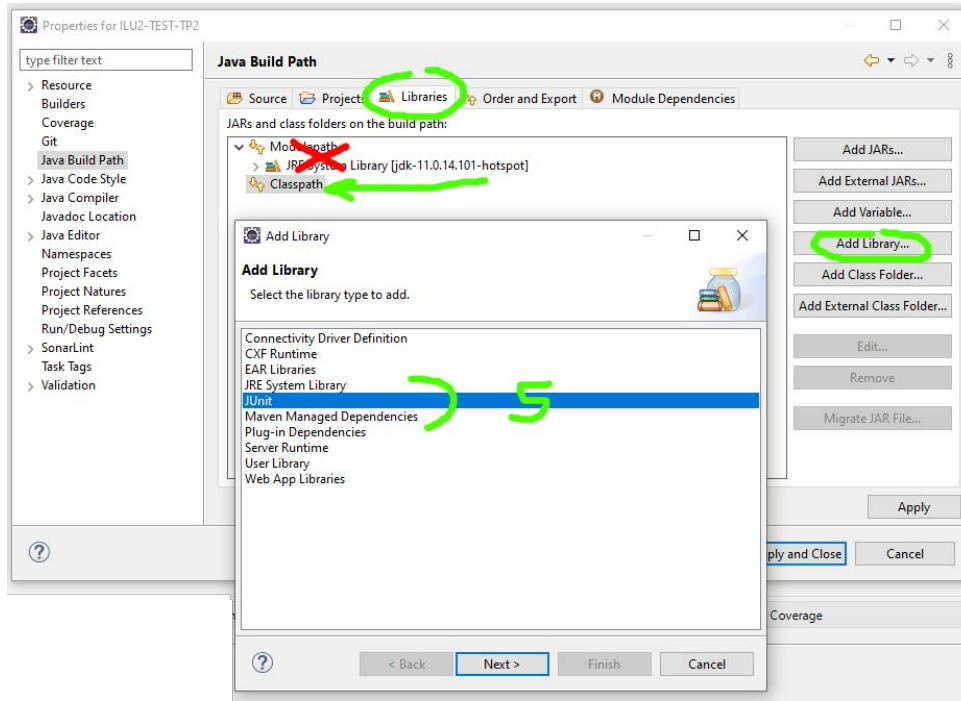


Ajouter un répertoire pour les tests

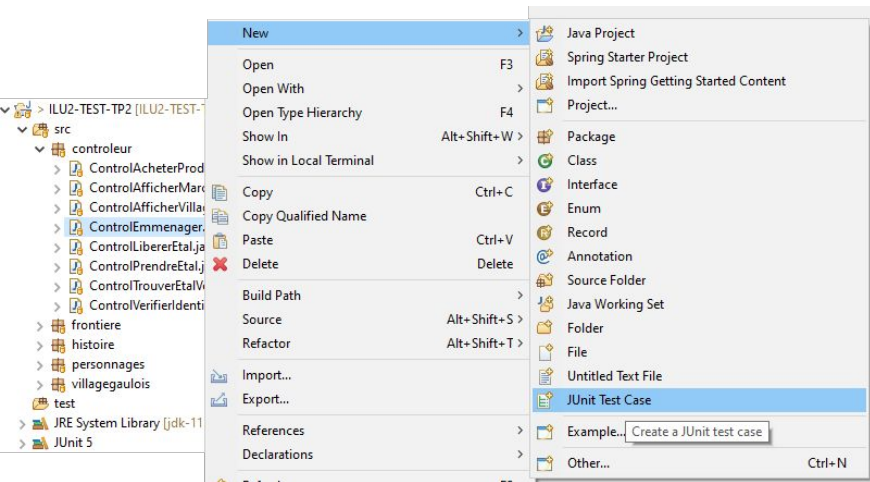
```
PROJECT_ROOT
+--- src/
+----test/
```



Ajouter JUnit 5 au projet

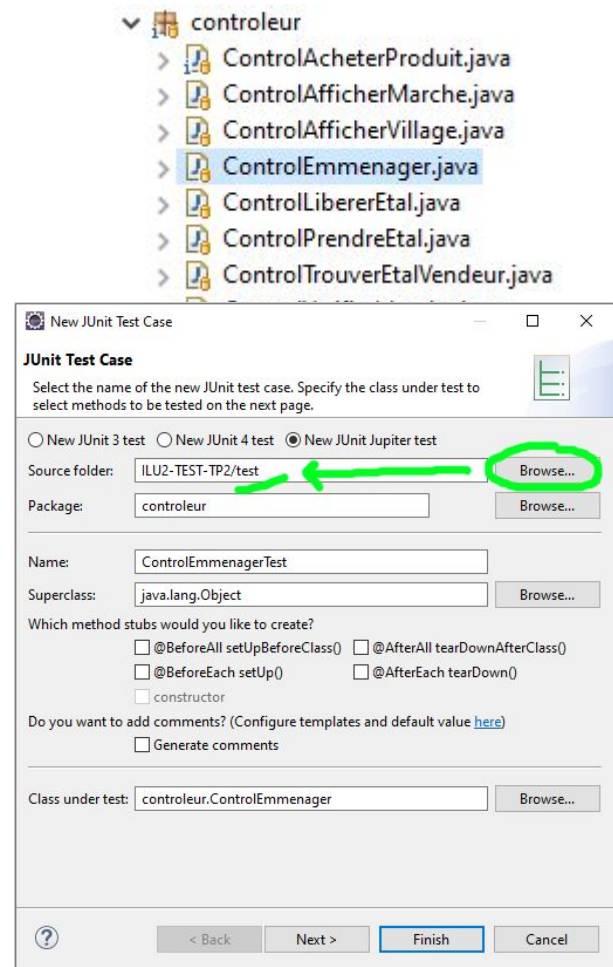


Choisir un contrôleur... (ControlEmmenager)



Click droit : ajouter un test...JUNIT 5
(modifier le répertoire !)

Next > indiquer les méthodes que vous souhaitez tester (toutes, sauf Object).



ILU2-TEST-TP2 [ILU2-TEST-TP2 main]

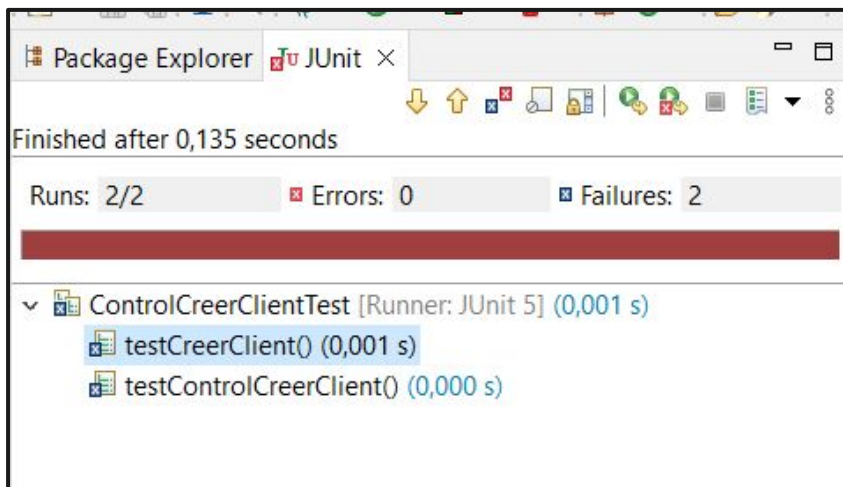
- src
 - controleur
 - ControlAcheterProduit.java
 - ControlAfficherMarche.java
 - ControlAfficherVillage.java
 - ControlEmmenager.java
 - ControlLibererEtal.java
 - ControlPrendreEtal.java
 - ControlTrouverEtalVendeur.java
 - ControlVerifierIdentite.java
 - frontiere
 - histoire
 - personnages
 - villagegaulois
- test
 - controleur
 - ControlEmmenagerTest.java
- JRE System Library [jdk-11.0.14.101-hotspot]
- JUnit 5

ControlEmmenagerTest.java

ILU2-TEST-TP2 ▶ test ▶ controleur ▶ ControlEmmenager

```
1 package controleur;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class ControlEmmenagerTest {
8
9     @Test
10     void testControlEmmenager() {
11         fail("Not yet implemented");
12     }
13
14     @Test
15     void testIsHabitant() {
16         fail("Not yet implemented");
17     }
18
19     @Test
20     void testAjouterDruide() {
21         fail("Not yet implemented");
22     }
23
24     @Test
25     void testAjouterGaulois() {
26         fail("Not yet implemented");
27     }
28
29 }
```


Lancer le test



Naturellement...

On a rien fait dans le test !

Coder les tests



- identifier ce que l'on souhaite observer et comment on souhaite le tester
 - Un constructeur doit retourner une instance (par opposition avec une levée d'exception)
 - un traitement fonctionnel doit permettre de vérifier qu'il a effectué ce qu'on attendait de lui...

Bienvenue dans le village des irréductibles dirigé par le chef Abraracourcix.

Ce village possède un joli marché avec 5 étals mis à la disposition des villageois afin qu'ils puissent vendre leurs produits.

Pour l'instant, le chef est bien seul dans son village.

Qui êtes-vous ?

- 1 - un voyageur
- 2 - un marchand
- 3 - un client du marché
- 4 - quitter l'application

1
Quel est votre nom ?

Bonemine

1 - je souhaite que vous me présentiez votre village.
2 - je voudrais emménager dans votre village.

3 - quitter l'application.

2

Êtes-vous :

- 1 - un druide.
- 2 - un gaulois.

2

Bienvenu villageois Bonemine

Quelle est votre force ?

3

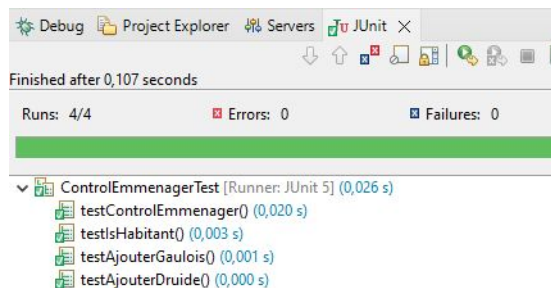
1 - je souhaite que vous me présentiez votre village.

2 - je voudrais emménager dans votre village.

3 - quitter l'application.

3

Au revoir voyageur Bonemine



```
class ControlEmmenagerTest {  
    private Village village;  
    private Chef abraracourcix;  
  
    @BeforeEach  
    public void initialiserSituation() {  
        System.out.println("Initialisation...");  
        village = new Village("le village des irréductibles", 10, 5);  
        abraracourcix = new Chef("Abraracourcix", 10, village);  
        village.setChef(abraracourcix);  
    }  
  
    @Test  
    void testControlEmmenager() {  
        ControlEmmenager controlEmmenager = new ControlEmmenager(village);  
        assertNotNull(controlEmmenager, "Constructeur ne renvoie pas null");  
    }  
  
    @Test  
    void testIsHabitant() {  
        ControlEmmenager controlEmmenager = new ControlEmmenager(village);  
        controlEmmenager.ajouterGaulois("Bonemine", 10);  
        assertTrue(controlEmmenager.isHabitant("Bonemine"));  
        assertFalse(controlEmmenager.isHabitant("Existe pas"));  
        controlEmmenager.ajouterDruide("Panoramix", 10, 1, 5);  
        assertTrue(controlEmmenager.isHabitant("Panoramix"));  
    }  
  
    @Test  
    void testAjouterDruide() {  
        ControlEmmenager controlEmmenager = new ControlEmmenager(village);  
        controlEmmenager.ajouterDruide("Panoramix", 10, 1, 5);  
    }  
}
```

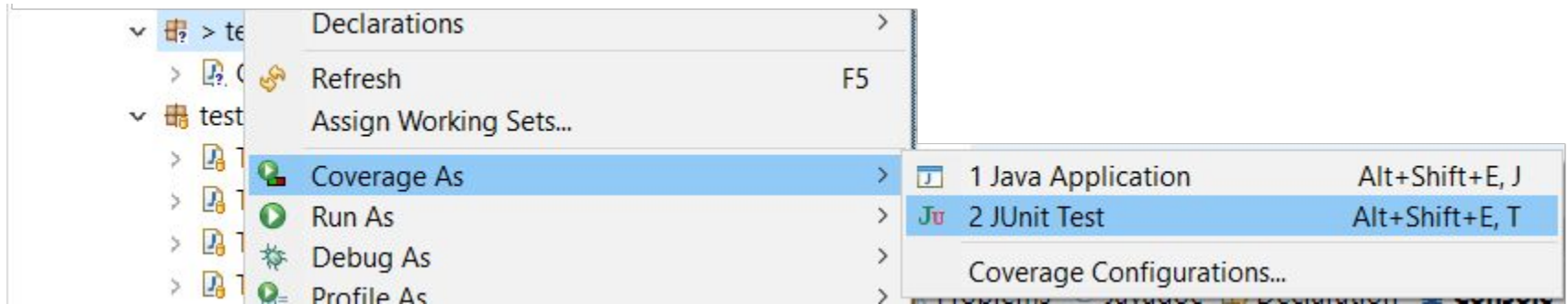


Couverture du code par les tests

- La **couverture de code** est une mesure utilisée pour décrire le taux de code source exécuté d'un programme quand une suite de test est lancée.
- Un programme avec une haute couverture de code, mesurée en pourcentage, a davantage de code exécuté durant les tests ce qui laisse à penser qu'il a moins de chance de contenir de bugs logiciels non détectés

Obtenir la couverture de code par les tests

- La couverture **de code** est une mesure utilisée pour décrire le taux de code source exécuté d'un programme quand une suite de test est lancée.
- Un programme avec une haute couverture de code, mesurée en pourcentage, a davantage de code exécuté durant les tests ce qui laisse à penser qu'il a moins de chance de contenir de bugs logiciels non détectés



ILU2-TEST-TP2 [ILU2-TEST-TP2]

src

test

controleur

ControlEmmenager

JRE System Library [jdk-11.0.12]

JUnit 5

README.md

Refresh F5

Close Project

Close Unrelated Projects

References

Declarations

Coverage As

Run As

Debug As

Ju 1 JUnit Test Alt+Shift+E, T

Coverage Configurations...

```
1 package controleur;
2
3 import static org.junit.Assert.assertTrue;
4 import static org.junit.jupiter.api.Assertions.assertFalse;
5 import static org.junit.jupiter.api.Assertions.assertNotNull;
6
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.Test;
9
10 import personnages.Chef;
11 import villagegaulois.Village;
12
13 class ControlEmmenagerTest {
14     private Village village;
15     private Chef abraracourcix;
16
17     @BeforeEach
18     public void initialiserSituation() {
19         System.out.println("Initialisation...");
20         village = new Village("le village des irréductibles", 10, 5);
21         abraracourcix = new Chef("Abraracourcix", 10, village);
22         village.setChef(abraracourcix);
23     }
24
25     @Test
26     void testControlEmmenager() {
27         ControlEmmenager controleur = new ControlEmmenager(village);
28         controleur.ajouterGaulois("Bonemine", 10);
29         controleur.ajouterDruide("Panoramix", 10, 1, 5);
30     }
31
32     @Test
33     void testIsHabitant() {
34         ControlEmmenager controleur = new ControlEmmenager(village);
35         controleur.ajouterGaulois("Bonemine", 10);
36         assertTrue(controlleur.isHabitant("Bonemine"));
37         assertFalse(controlleur.isHabitant("Existe pas"));
38         controleur.ajouterDruide("Panoramix", 10, 1, 5);
39         assertTrue(controlleur.isHabitant("Panoramix"));
40     }
41
42     @Test
43     void testAjouterDruide() {
44         ControlEmmenager controleur = new ControlEmmenager(village);
45         controleur.ajouterDruide("Panoramix", 10, 1, 5);
46     }
47 }
```

Finished after 0,18 seconds

Runs: 4/4 Errors: 0 Failures: 0

ControlEmmenagerTest [Runner: JUnit 5] (0,015 s)

testControlEmmenager() (0,006 s)

testIsHabitant() (0,005 s)

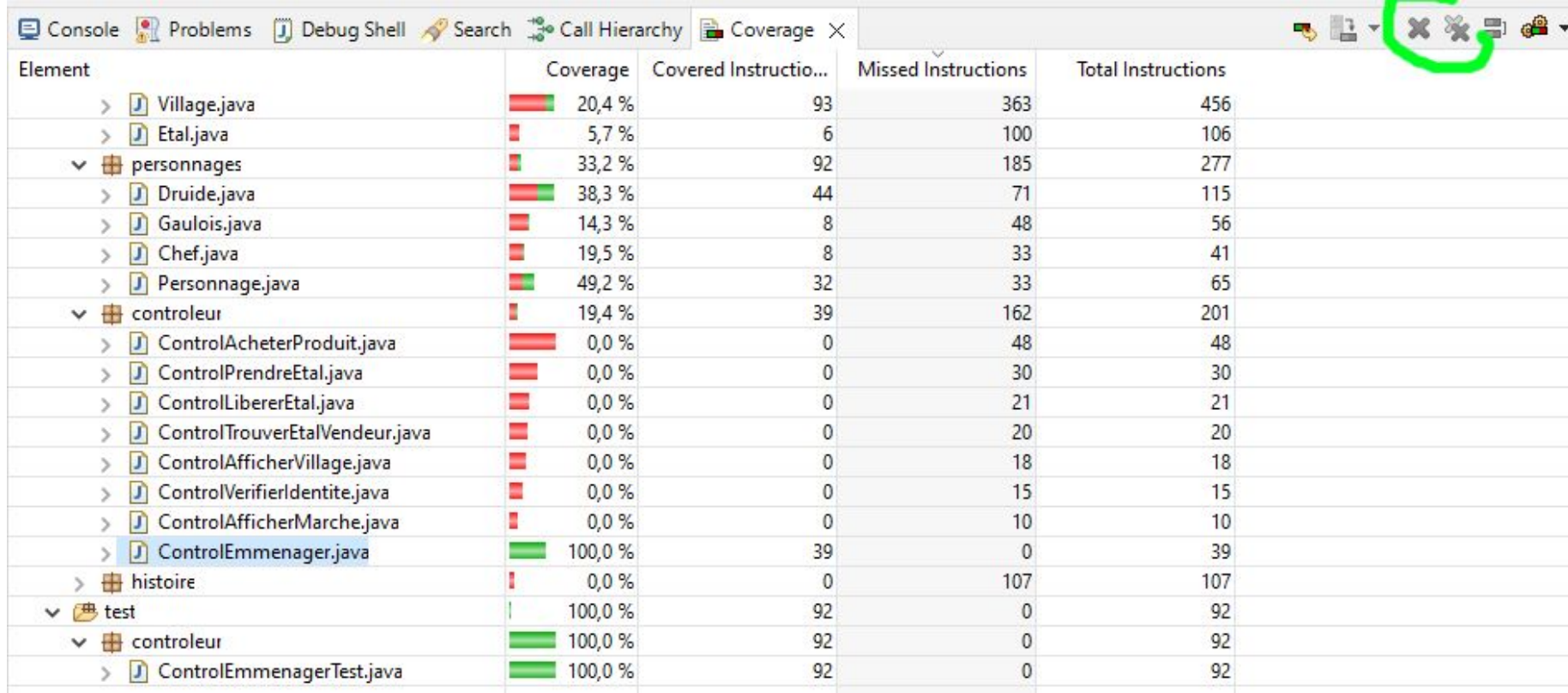
testAjouterGaulois() (0,001 s)

testAjouterDruide() (0,001 s)

Console Problems Debug Shell Search Call Hierarchy Coverage

Element	Coverage	Covered Instruction...	Missed Instructions	Total Instructions
ILU2-TEST-TP2	14,7 %	322	1 873	2 195
src	10,9 %	230	1 873	2 103
frontiere	0,0 %	0	956	956
villagegaulois	17,6 %	99	463	562
personnages	33,2 %	92	185	277
controleur	19,4 %	39	162	201
histoire	0,0 %	0	107	107
test	100,0 %	92	0	92
controleur	100,0 %	92	0	92
ControlEmmenagerTest.java	100,0 %	92	0	92

Obtenir la couverture de code par les tests



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
> Village.java	20,4 %	93	363	456
> Etal.java	5,7 %	6	100	106
▼ personnages	33,2 %	92	185	277
> Druide.java	38,3 %	44	71	115
> Gaulois.java	14,3 %	8	48	56
> Chef.java	19,5 %	8	33	41
> Personnage.java	49,2 %	32	33	65
▼ controleur	19,4 %	39	162	201
> ControlAcheterProduit.java	0,0 %	0	48	48
> ControlPrendreEtal.java	0,0 %	0	30	30
> ControlLibererEtal.java	0,0 %	0	21	21
> ControlTrouverEtalVendeur.java	0,0 %	0	20	20
> ControlAfficherVillage.java	0,0 %	0	18	18
> ControlVerifierIdentite.java	0,0 %	0	15	15
> ControlAfficherMarche.java	0,0 %	0	10	10
> ControlEmmenager.java	100,0 %	39	0	39
> histoire	0,0 %	0	107	107
▼ test	100,0 %	92	0	92
▼ controleur	100,0 %	92	0	92
> ControlEmmenagerTest.java	100,0 %	92	0	92

ControlEmmenagerTest.java Scenario.java Assertions.class ControlEmmenager.java Druid.java ×

```

1 package personnages;
2
3 import java.util.Random;
4
5 public class Druid extends Gaulois {
6     private int effetPotionMin;
7     private int effetPotionMax;
8     private int forcePotion = 1;
9
10    public Druid(String nom, int force, int effetPotionMin,
11        int effetPotionMax) {
12        super(nom, force);
13        this.effetPotionMin = effetPotionMin;
14        this.effetPotionMax = effetPotionMax;
15        parler("Bonjour, je suis le druide " + nom
16            + " et ma potion peut aller d'une force " + effetPotionMin
17            + " à " + effetPotionMax + ".");
18    }
19
20    public void preparerPotion() {
21        Random random = new Random();
22        forcePotion = (random.nextInt(effetPotionMax - effetPotionMin)
23            + effetPotionMin);
24        String texte = "";
25        All 2 branches missed, potion > 7) {
26            texte += "J'ai préparé une super potion ";
27        } else {
28            texte += "Je n'ai pas trouvé tous les ingrédients, ma potion est seulement ";
29        }
30        parler(texte + "de force " + forcePotion + ".");
31    }
32
33    public void booster(Gaulois gaulois) {
34        if (gaulois.getNom().equals("Obélix")) {
35            parler("Non, Obélix !... Tu n'auras pas de potion magique !");
36        } else {
37            gaulois.boirePotion(forcePotion);
38        }
39    }
40
41    @Override
42    protected String prendreParole() {
43        return "Le druide " + nom + " : ";
44    }

```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
ILU2-TEST-TP2	14,7 %	322	1 873	2 195
src	10,9 %	230	1 873	2 103
frontiere	0,0 %	0	956	956
villagegaulois	17,6 %	99	463	562
personnages	33,2 %	92	185	277
Druid.java	38,3 %	44	71	115
Gaulois.java	14,3 %	8	48	56
Chef.java	19,5 %	8	33	41
Personnage.java	49,2 %	32	33	65
controleur	19,4 %	39	162	201

La vérification de la couverture de code est fine :

- est-ce qu'on passe par toutes les branches des if / then / else ?
- est-ce qu'on rentre dans les boucles ?
- est-ce qu'on rentre dans les try/catch ?





A vous de jouer...

Ajouter des tests unitaires:

- obtenir la couverture des contrôleurs
- obtenir la couverture du modèle



La recherche de la couverture du code par les tests est aussi l'occasion d'effectuer une relecture critique du code.

Est-ce que le code se protège des arguments invalides ?

Est-ce qu'une stratégie de gestion d'erreur a été implémentée ?

Concrètement

Les tests font partie du code source d'un projet.

Ils doivent être **commit**, et rester d'une qualité suffisante pour être maintenus dans le temps.

Ils doivent être maintenus à jour au fur-et-à-mesure des évolutions apportées par le code.

Ils ont vocation à être enrichis lorsque des bugs sont détectés, pour fournir un premier niveau de tests de non régression.

Il faut rester pragmatique, **le coût des tests unitaires doit rester faible**, pour maximiser le retour sur investissement :

- les tests unitaires n'ont pas vocation à être eux-même testés, ils doivent être simples
- les tests unitaires ont vocation à être exécutés souvent, ils doivent rester légers
- les tests unitaires ne doivent pas avoir d'effet de bord sur l'environnement de développement

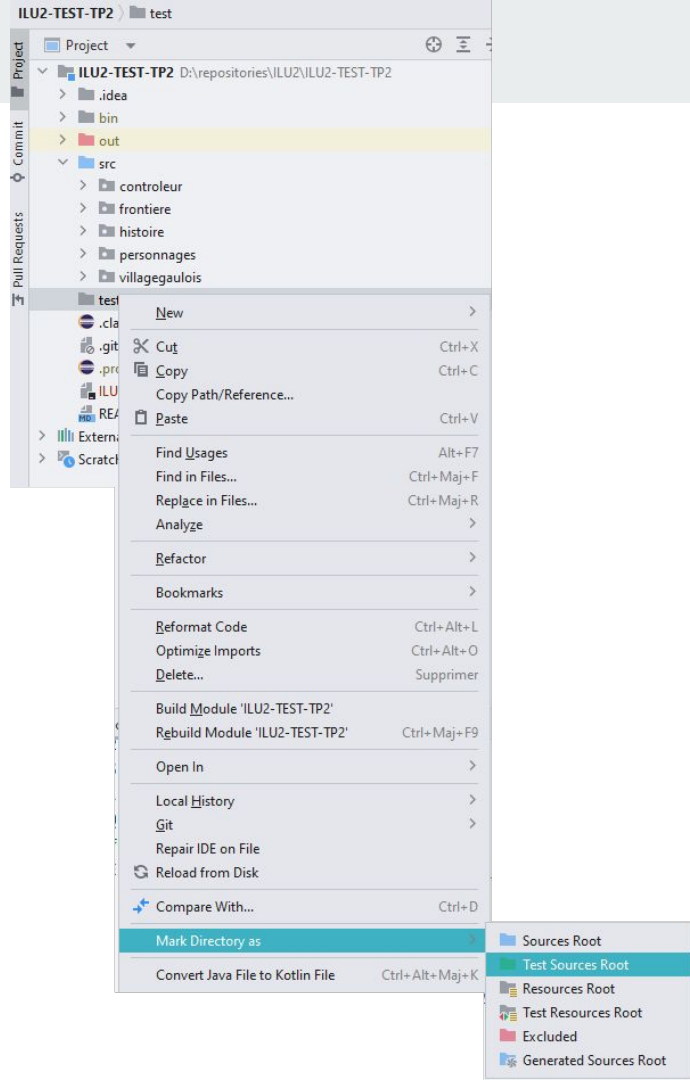


Si vous êtes sous IntelliJ...

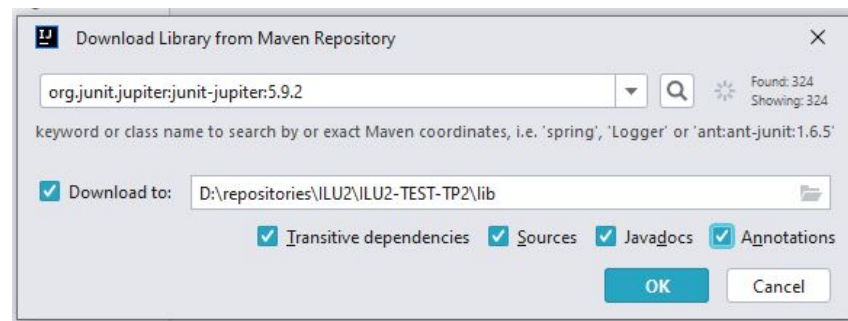
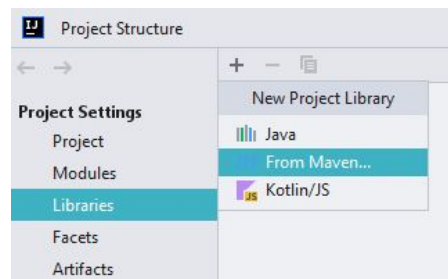
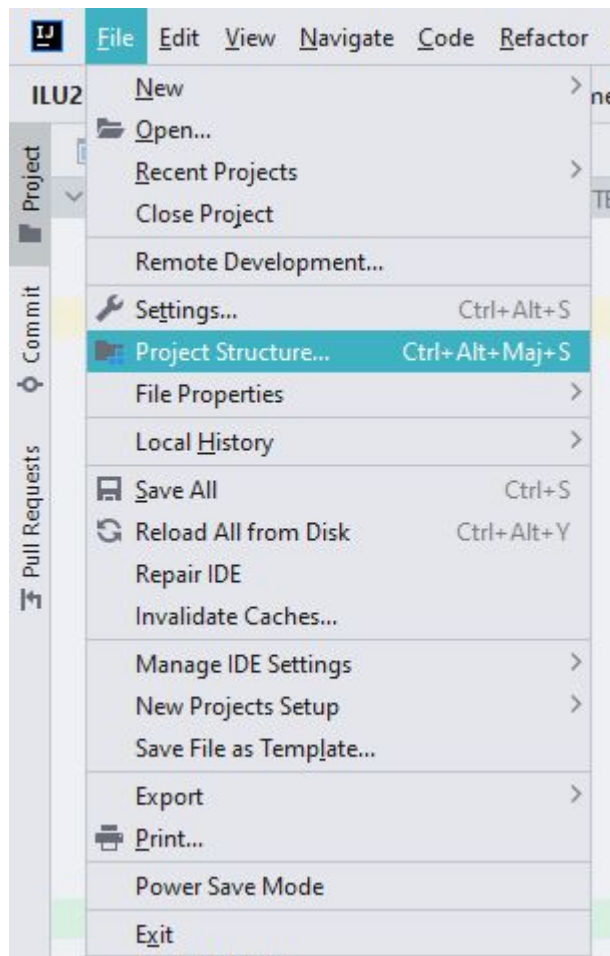


Ajouter un répertoire pour les tests

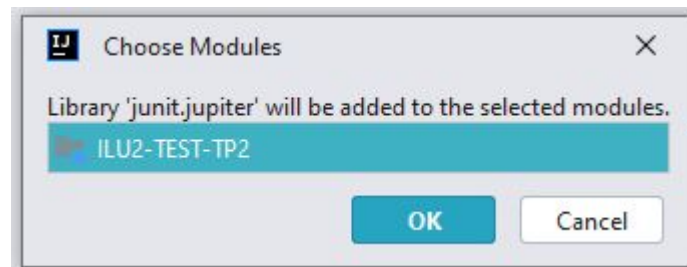
```
PROJECT_ROOT
+--- src/
+----test/
```



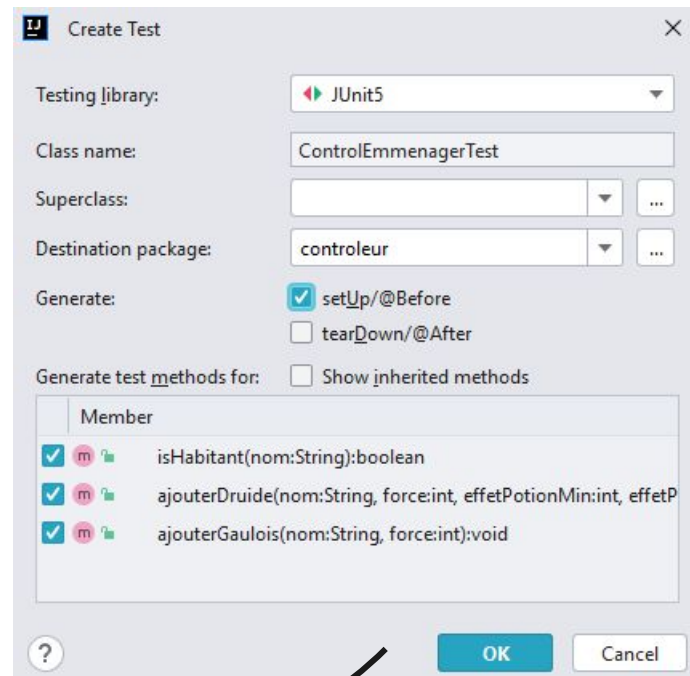
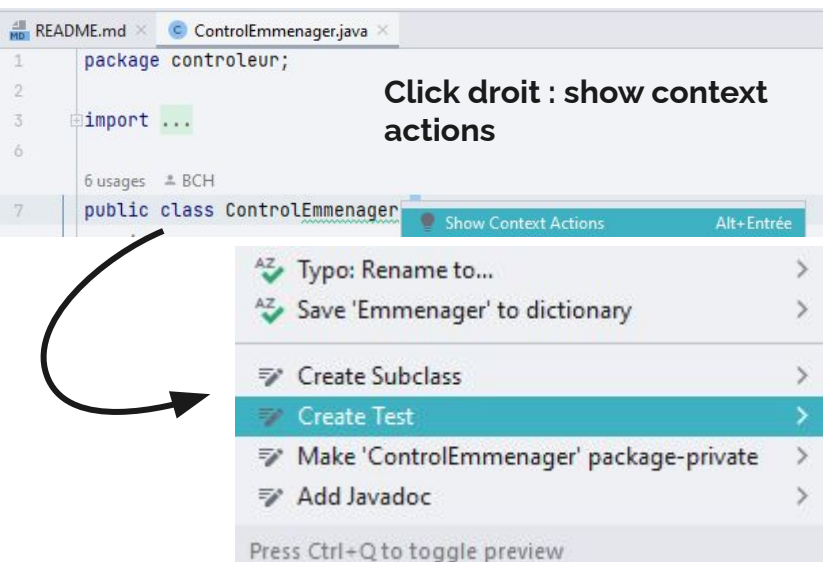
Ajouter JUnit 5 au projet



Et attendre que ça
télécharge...



Choisir un contrôleur...(*ControlEmmenager*)



ILU2-TEST-TP2 > test > controleur > ControlEmmenagerTest

Project

- ILU2-TEST-TP2 D:\repositories\ILU2\ILU2-TEST-TP2
 - .idea
 - bin
 - lib
 - out
 - src
 - controleur
 - ControlAcheterProduit
 - ControlAfficherMarche
 - ControlAfficherVillage
 - ControlEmmenager
 - ControlLibererEtal
 - ControlPrendreEtal
 - ControlTrouverEtalVendeur
 - ControlVerifierIdentite
 - frontiere
 - histoire
 - personnages
 - villagegaulois
 - test
 - controleur
 - ControlEmmenagerTest
 - .classpath
 - .gitignore
 - .project
 - ILU2-TEST-TP2.iml
 - README.md
- External Libraries

Run: ControlEmmenagerTest

Tests passed: 3 of 3 tests – 22 ms

ControlEmmenagerTest (controleur)	22 ms
ajouterGaulois()	20 ms
isHabitant()	1 ms
ajouterDruide()	1 ms

D:\Dev\Java\jdk-11.0.14.9-hotspot\bin\java.exe ...

Process finished with exit code 0

```
package controleur;

import ...

no usages
class ControlEmmenagerTest {

    no usages
    @BeforeEach
    void setUp() {
    }

    no usages
    @Test
    void isHabitant() {
    }

    no usages
    @Test
    void ajouterDruide() {
    }

    no usages
    @Test
    void ajouterGaulois() {
    }
```

Lancer le test

clic droit sur le test :

- run



Coder les tests

- identifier ce que l'on souhaite observer et comment on souhaite le tester
 - Un constructeur doit retourner une instance (par opposition avec une levée d'exception)
 - un traitement fonctionnel doit permettre de vérifier qu'il a effectué ce qu'on attendait de lui...

Bienvenue dans le village des irréductibles dirigé par le chef Abraracourcix.

Ce village possède un joli marché avec 5 étals mis à la disposition des villageois afin qu'ils puissent vendre leurs produits.

Pour l'instant, le chef est bien seul dans son village.

Qui êtes-vous ?

- 1 - un voyageur
- 2 - un marchand
- 3 - un client du marché
- 4 - quitter l'application

1
Quel est votre nom ?
Bonemine

- 1 - je souhaite que vous me présentiez votre village.
- 2 - je voudrais emménager dans votre village.
- 3 - quitter l'application.

2

Êtes-vous :

- 1 - un druide.
- 2 - un gaulois.

2

Bienvenu villageois Bonemine

Quelle est votre force ?

3

- 1 - je souhaite que vous me présentiez votre village.
- 2 - je voudrais emménager dans votre village.

3 - quitter l'application.

3

Au revoir voyageur Bonemine

no usages

@BeforeEach

void setUp() {

System.out.println("Initialisation...");

village = new Village(nom: "le village des irréductibles", nbVillageoisMaximum: 10, nbEtal: 5);

abraracourcix = new Chef(nom: "Abraracourcix", force: 10, village);

village.setChef(abraracourcix);

}

no usages

@Test

void isHabitant() {

ControlEmmenager controlEmmenager = new ControlEmmenager(village);

controlEmmenager.ajouterGaulois(nom: "Bonemine", force: 10);

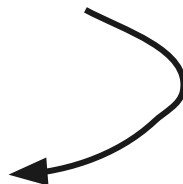
assertTrue(controlEmmenager.isHabitant(nom: "Bonemine"));

assertFalse(controlEmmenager.isHabitant(nom: "Existe pas"));

controlEmmenager.ajouterDruide(nom: "Panoramix", force: 10, effetPotionMin: 1, effetPotionMax: 5);

assertTrue(controlEmmenager.isHabitant(nom: "Panoramix"));

}



Run: ControlEmmenagerTest x

Tests passed: 3 of 3 tests - 53 ms

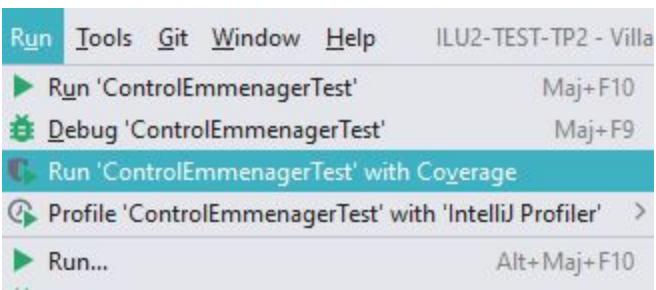
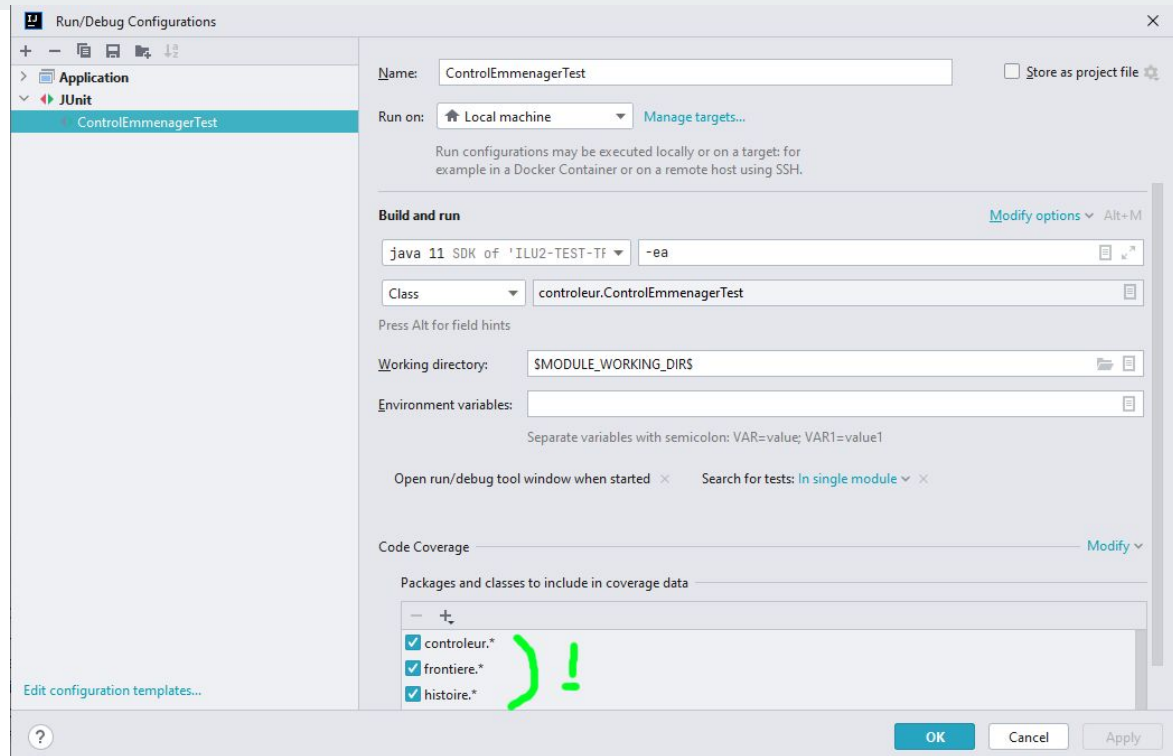
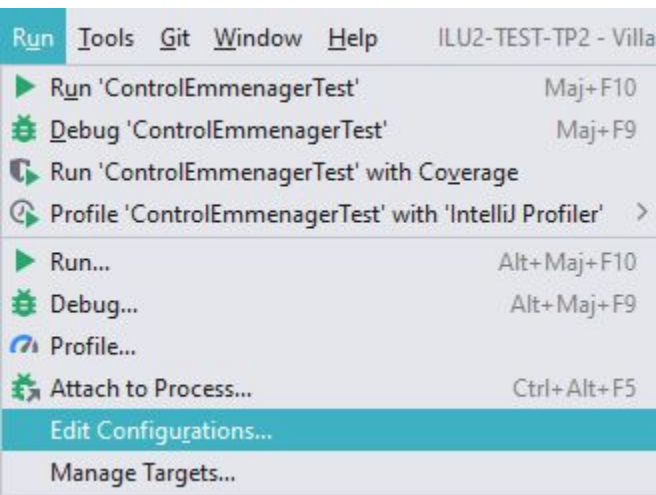
ControlEmmenagerTest (controleur)	53 ms	Initialisation...
ajouterGaulois()	28 ms	Le druide Panoramix : « Bo
isHabitant()	24 ms	
ajouterDruide()	1 ms	



Couverture du code par les tests

- La **couverture de code** est une mesure utilisée pour décrire le taux de code source exécuté d'un programme quand une suite de test est lancée.
- Un programme avec une haute couverture de code, mesurée en pourcentage, a davantage de code exécuté durant les tests ce qui laisse à penser qu'il a moins de chance de contenir de bugs logiciels non détectés

Configurer la couverture des tests



README.mdControlEmmenager.javaChef.javaVillage.javaControlAfficherVillage.java

```
11 private Gaulois[] villageois;
12     6 usages
13 private int nbVillageois = 0;
14     9 usages
15 private Marche marche;
16
17 BCH
18 public Village(String nom, int nbVillageoisMaximum, int nbEtal) {
19     this.nom = nom;
20     villageois = new Gaulois[nbVillageoisMaximum];
21     marche = new Marche(nbEtal);
22 }
23
24 BCH
25 public String getNom() { return nom; }
26
27     2 usages BCH
28 public void setChef(Chef chef) { this.chef = chef; }
29
30     2 usages BCH
31 public void ajouterHabitant(Gaulois gaulois) {
32     if (nbVillageois < villageois.length) {
33         villageois[nbVillageois] = gaulois;
34         nbVillageois++;
35     }
36 }
```

Coverage: ControlEmmenagerTest

Element	Class, %	Method, %	Line, %
all	100% (3/3)	16% (5/30)	17% (22/123)
controleur	12% (1/8)	16% (4/24)	14% (7/47)
ControlAcheterProduit	0% (0/1)	0% (0/3)	0% (0/12)
ControlAfficherMarche	0% (0/1)	0% (0/2)	0% (0/3)
ControlAfficherVillage	0% (0/1)	0% (0/4)	0% (0/5)
ControlEmmenager	100% (1/1)	100% (4/4)	100% (7/7)
ControlLibererEtal	0% (0/1)	0% (0/3)	0% (0/4)
ControlPrendreEtal	0% (0/1)	0% (0/4)	0% (0/7)
ControlTrouverEtalVendeur	0% (0/1)	0% (0/2)	0% (0/6)
ControlVerifierIdentite	0% (0/1)	0% (0/2)	0% (0/3)
frontiere	0% (0/8)	0% (0/23)	0% (0/214)
histoire	0% (0/1)	0% (0/1)	0% (0/19)
personnages	100% (4/4)	44% (8/18)	44% (17/38)
villagegaulois	100% (3/3)	16% (5/30)	17% (22/123)
Etal	100% (1/1)	0% (0/9)	7% (2/28)
Village	100% (2/2)	23% (5/21)	21% (20/95)