EXAMEN TEST UNITAIRE

Lien du GITHUB : https://github.com/Meidi-Agd/EXAMEN_Test_Unitaire.git

Lien du Word : https://1drv.ms/w/s!ArRMvTF90USVhcdh1i9Qko93d4jdlA?e=Ar6wGo

Exercice 1

A) La classe Employee

```java
package EmployeeManager;

public class Employee {  1 usage
    private String nom;  2 usages
    private String prenom;  2 usages
    private int anneeExp;  2 usages
    private String niveau;  2 usages

    public Employee(String nom, String prenom, int anneeExp, String niveau) {  no usages
        this.nom = nom;
        this.prenom = prenom;
        this.anneeExp = anneeExp;
        this.niveau = niveau;
    }

    public String getNom() {  no usages
        return nom;
    }

    public String getPrenom(){  no usages
        return prenom;
    }

    public int getAnneeExp(){  no usages
        return anneeExp;
    }

    public String getNiveau(){  no usages
        return niveau;
    }
}
```

B) La méthode calculateSalary

```java
public int calculateSalary(Employee employee) {  no usages
    String niveau = employee.getNiveau();

    switch (niveau) {
        case "Junior":
            return 20000;
        case "Intermédiaire":
            return 40000;
        case "Senior":
            return 60000;
        default:
            return 0;
    }
}
```

C) La méthode de calcul du coefficient

```java
public double calculateExperienceMultiplier(int anneesExperience)  no usages  new *
{
    double coefficient = 1.00;
    for (int i = 0; i < anneesExperience; i++)
    {
        coefficient += 0.05;
    }

    return coefficient;
}
```

D) Méthode de test pour vérifier son salaire d'ancienneté d'un Junior

```java
public class testEmployeeManager {  new *
    private final EmployeeManager employeeManager = new EmployeeManager();  2 usages

    @Test // Méthode exercice 1 D  new *
    public void testEmployeeNiveauJuniorUnAn() {
        Employee emp = new Employee( nom: "Son",  prenom: "John",  anneeExp: 1,  niveau: "Junior");

        double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
        assertEquals( expected: 21000, result);
    }
}
```

E) Méthode de test pour vérifier son salaire d'ancienneté d'un Intermédiaire

```java
    @Test // Méthode exercice 1 E  new *
    public void testEmployeeNiveauInterCinqAn() {
        Employee emp = new Employee( nom: "Son",  prenom: "John",  anneeExp: 5,  niveau: "Intermédiaire");

        double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
        int intResult = (int) result;
        assertEquals( expected: 50000, intResult);
    }
```

F) Méthode de test pour vérifier son salaire d'ancienneté d'un Sénior

```java
    @Test // Méthode exercice 1 E  new *
    public void testEmployeeNiveauSeniorDixAn() {
        Employee emp = new Employee( nom: "Son",  prenom: "John",  anneeExp: 10,  niveau: "Senior");

        double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
        int intResult = (int) result;
        assertEquals( expected: 90000, intResult);
    }
```

G) Méthode de test pour vérifier son salaire d'ancienneté d'un Sénior

```java
@Test // Méthode exercice 1 E  new *
public void testEmployeeNiveauSeniorVingtAn() {
    Employee emp = new Employee( nom: "Son",  prenom: "John",  anneeExp: 20,  niveau: "Senior");

    double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
    int intResult = (int) result;
    assertEquals( expected: 120000, intResult);
}
```

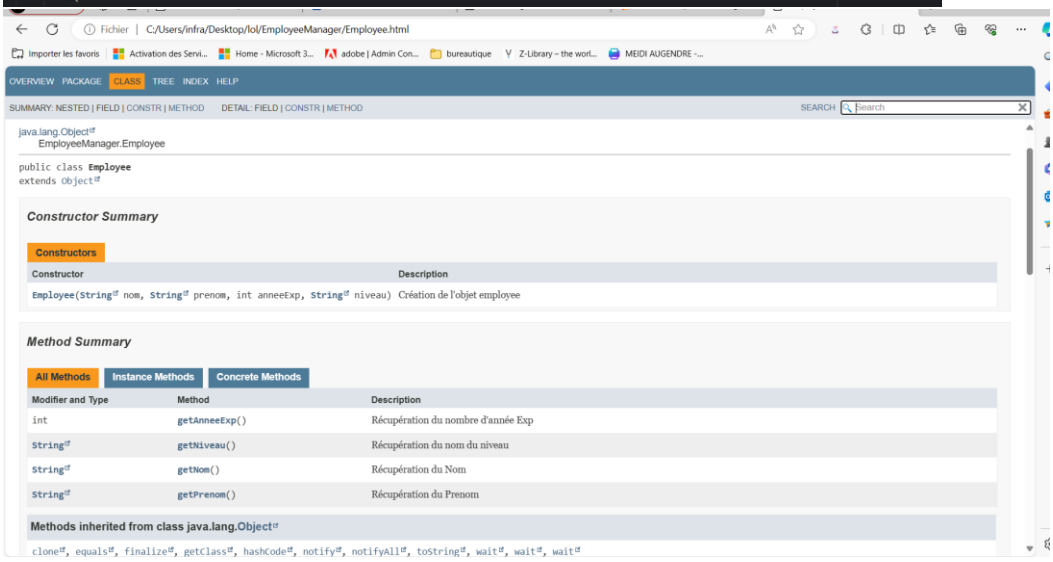
H) Méthode de test pour vérifier son salaire d'ancienneté d'un Sénior

```java
@Test // Méthode exercice 1 H  new *
public void testEmployeeNiveauJuniorQuinzeAn() {
    Employee emp = new Employee( nom: "Son", prenom: "John", anneeExp: 15, niveau: "Junior");

    double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
    int intResult = (int) result;
    assertEquals( expected: 35000, intResult);
}
```

```java
@Test // Méthode exercice 1 H  new *
public void testEmployeeNiveauInterQuinzeAn() {
    Employee emp = new Employee( nom: "Son", prenom: "John", anneeExp: 15, niveau: "Intermédiaire");

    double result = employeeManager.calculateSalary(emp) * employeeManager.calculateExperienceMultiplier(emp.getAnneeExp());
    int intResult = (int) result;
    assertEquals( expected: 70000, intResult);
}
```

I) Documentation de son code et JavaDoc

```java
/**
 * Calcule du salaire de l'employee via son niveau
 * @param employee
 * @return salaire en int
 */
public int calculateSalary(Employee employee) {  6 usages  new *
    String niveau = employee.getNiveau();

    switch (niveau) {
        case "Junior":
            return 20000;
        case "Intermédiaire":
            return 40000;
        case "Senior":
            return 60000;
        default:
            return 0;
    }
}

/**
 * Calcule de coefficient par le nombre d'année
 * @param anneesExperience
 * @return le coefficient multiplicateur en double
 */
public double calculateExperienceMultiplier(int anneesExperience)  6 usages  new *
{
    double coefficient = 1.00;
    for (int i = 0; i < anneesExperience; i++)
    {
```



Exercice 2

1) Création de la classe InventoryManager

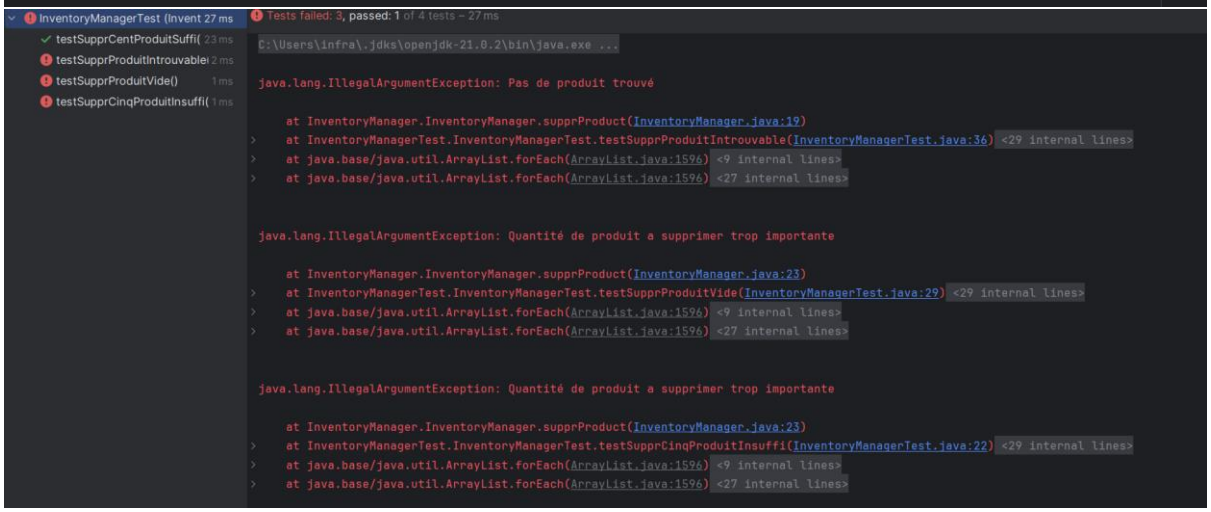
```java
public class InventoryManager {  3 usages  new *
    private Map<String, Integer> inventory;  6 usages

    public InventoryManager() {  1 usage  new *
        inventory = new HashMap<>();
    }

    public void ajoutProduit(String produit, int quantite) {  5 usages  new *
        this.inventory.put(produit, quantite);
    }

    public void supprProduct(String produit, int quantite) throws IllegalArgumentException {  4 usages  new *
        if (!this.inventory.containsKey(produit)) {
            throw new IllegalArgumentException("Pas de produit trouvé");
        }
        int currentQuantity = this.inventory.get(produit);
        if (quantite > currentQuantity) {
            throw new IllegalArgumentException("Quantité de produit a supprimer trop importante");
        }
        else
            this.inventory.put(produit, currentQuantity - quantite);
    }

    Rename usages
    public int recupStock(String product) {  new *  1 related problem
        return this.inventory.getOrDefault(product,  defaultValue: 0);
    }
}
```

2, 3, 4) Classe de test InventoryManagerTest

```java
package InventoryManagerTest;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class InventoryManagerTest {  2 usages  new *
    private final InventoryManagerTest inventoryManager = new InventoryManagerTest();  no usages

}
```

5) Les tests

```java
@Test //Cas où le stock est suffisant (par exemple, 100 unités disponibles pour un produit donné)  new *
public void testSupprCentProduitSuffi() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 100);

    inventoryManager.supprProduct( produit: "Huawei P60",  quantite: 10);
}

@Test //Cas où le stock est insuffisant (par exemple, 5 unités disponibles pour un produit donné).  new *
public void testSupprCinqProduitInsuffi() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 5);

    inventoryManager.supprProduct( produit: "Huawei P60",  quantite: 10);
}

@Test //Cas où le stock est épuisé (par exemple, 0 unité disponible pour un produit donné).  new *
public void testSupprProduitVide() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 0);

    inventoryManager.supprProduct( produit: "Huawei P60",  quantite: 10);
}

@Test //Cas où le produit n'existe pas dans l'inventaire.  new *
public void testSupprProduitIntrouvable() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 0);

    inventoryManager.supprProduct( produit: "Huawei P50",  quantite: 10);
}
```



6) Autres tests

```java
@Test //Cas où le produit n'existe pas dans l'inventaire.  new *
public void getStockAvailabilityInexistant() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 667);

    int nbStock = inventoryManager.recupStock( product: "Huawei P50");
    System.out.println("Il y en a " + nbStock + " en stock");
}
```

```
InventoryManagerTest (Invent 27 ms
  getStockAvailabilityInexiste 27 ms
Tests passed: 1 of 1 test – 27 ms
C:\Users\infra\.jdks\openjdk-21.0.2\bin\java.exe ...
Il y en a 0 en stock

Process finished with exit code 0
```

7) La méthode getStockAvailability

```java
@Test //Cas où le produit n'existe pas dans l'inventaire.  new *
public void getStockAvailability() {
    inventoryManager.ajoutProduit( produit: "Huawei P60",  quantite: 667);

    int nbStock = inventoryManager.recupStock( product: "Huawei P60");
    System.out.println("Il y en a " + nbStock + " en stock");
}
}
```

```
InventoryManagerTest (Invent 26 ms
  getStockAvailability() 26 ms
Tests passed: 1 of 1 test – 26 ms
C:\Users\infra\.jdks\openjdk-21.0.2\bin\java.exe ...
Il y en a 667 en stock

Process finished with exit code 0
```

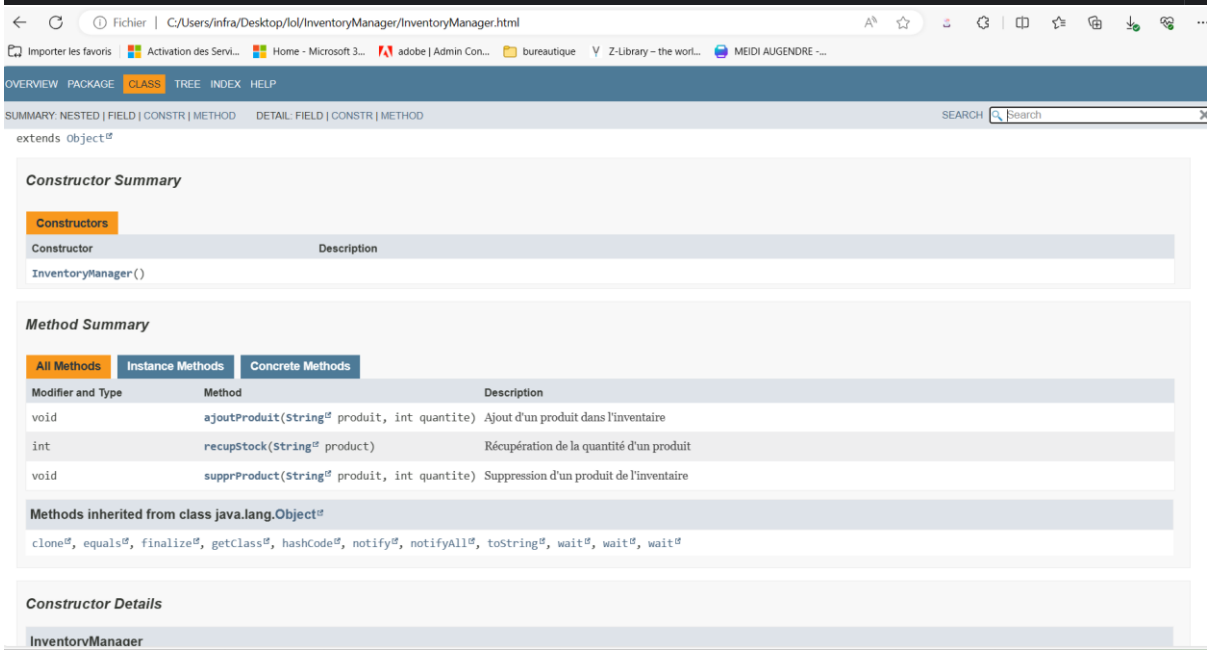
8, 9) Résultat des Tests

```
InventoryManagerTest (Invent 39 ms
  getStockAvailabilityInexiste 33 ms
  testSupprCentProduitSuffi() 1 ms
  testSupprProduitIntrouvable 2 ms
  testSupprProduitVide() 1 ms
  testSupprCinqProduitInsuffi 1 ms
  getStockAvailability() 1 ms
Tests failed: 3, passed: 3 of 6 tests – 39 ms
Il y en a 0 en stock

java.lang.IllegalArgumentException: Pas de produit trouvé

    at InventoryManager.InventoryManager.supprProduct(InventoryManager.java:20)
    at InventoryManagerTest.InventoryManagerTest.testSupprProduitIntrouvable(InventoryManagerTest.java:36) <29 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <27 internal lines>


java.lang.IllegalArgumentException: Quantité de produit a supprimer trop importante

    at InventoryManager.InventoryManager.supprProduct(InventoryManager.java:24)
    at InventoryManagerTest.InventoryManagerTest.testSupprProduitVide(InventoryManagerTest.java:29) <29 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <27 internal lines>


java.lang.IllegalArgumentException: Quantité de produit a supprimer trop importante

    at InventoryManager.InventoryManager.supprProduct(InventoryManager.java:24)
    at InventoryManagerTest.InventoryManagerTest.testSupprCinqProduitInsuffi(InventoryManagerTest.java:22) <29 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <27 internal lines>

Il y en a 667 en stock
```

10) Documentation de son code et JavaDoc

```java
/**
 * Ajout d'un produit dans l'inventaire
 * @param produit
 * @param quantite
 */
public void ajoutProduit(String produit, int quantite) {  6 usages  new *
    this.inventory.put(produit, quantite);
}

/**
 * Suppression d'un produit de l'inventaire
 * @param produit
 * @param quantite
 * @throws IllegalArgumentException
 */
public void supprProduct(String produit, int quantite) throws IllegalArgumentException {  4 usages  new *
    if (!this.inventory.containsKey(produit)) {
        throw new IllegalArgumentException("Pas de produit trouvé");
    }
    int currentQuantity = this.inventory.get(produit);
    if (quantite > currentQuantity) {
        throw new IllegalArgumentException("Quantité de produit a supprimer trop importante");
    }
    else
        this.inventory.put(produit, currentQuantity - quantite);
}
```

```java
/**
 * Récupération de la quantité d'un produit
 * @param product
 * @return
 */
public int recupStock(String product) {  2 usages  new *
    return this.inventory.getOrDefault(product,  defaultValue: 0);
}
```



Fichier | C:/Users/infra/Desktop/lol/InventoryManager/InventoryManager.html

Importer les favoris | Activation des Servi... | Home - Microsoft 3... | adobe | Admin Con... | bureautique | Z-Library – the worl... | MEIDI AUGENDRE -...

OVERVIEW PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH Search

extends Object

Constructor Summary

Constructors

Constructor	Description
InventoryManager()	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	ajoutProduit(String produit, int quantite)	Ajout d'un produit dans l'inventaire
int	recupStock(String product)	Récupération de la quantité d'un produit
void	supprProduct(String produit, int quantite)	Suppression d'un produit de l'inventaire

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

InventoryManager

Exercice 3

Étape 1 : Création des classes de base

1) Création de la classe Book

```java
package Bibliotheque;

public class Book {  no usages  new *
    /**
     * Déclaration des variables en private
     */
    private String titre;  2 usages
    private String author;  2 usages

    /**
     * Création de l'objet book
     * @param titre
     * @param author
     */
    public Book(String titre, String author) {  no usages  new *
        this.titre = titre;
        this.author = author;
    }

    /**
     * Récupération du titre
     * @return
     */
    public String getTitre() {  no usages  new *
        return titre;
    }
}
```

```java
/**
 * Récupération de l'auteur
 * @return
 */
public String getAuthor(){  no usages  new *
    return author;
}
```

2) La classe User

```java
package Bibliotheque;

import java.util.List;

public class User {  no usages  new *
    /**
     * Déclaration des variables en private
     */
    private String name;  2 usages
    private List<Book> borrowedBooks;  2 usages

    /**
     * Création de l'objet book
     * @param name
     * @param borrowedBooks
     */
    public User(String name, List<Book> borrowedBooks) {  no usages  new *
        this.name = name;
        this.borrowedBooks = borrowedBooks;
    }

    /**
     * Récupération du titre
     * @return
     */
    public String getName() {  no usages  new *
        return name;
    }
```

EXAMEN

```java
    /**
     * Récupération du titre
     * @return
     */
    public String getName() {  no usages  new *
        return name;
    }

    /**
     * Récupération de l'auteur
     * @return
     */
    public String getborrowedBooks(){  no usages  new *
        return borrowedBooks.toString()
                ;
    }
}
```