

---

# COMPTE RENDU

## TP04 – PROGEFF

---

Christopher Marie-Angélique – Groupe 2B

### Table des matières

Introduction.....	2
Exercice 1.....	2
Code & résultats : .....	2
Signature : .....	2
Exercice 2.....	2
Code & résultats : .....	2
Exercice 3.....	3
Exercice 4.....	4
Conclusion .....	<b>Erreur ! Signet non défini.</b>

# Introduction

Dans ce TP de programmation Efficace, nous manipulerons l'interface Java « Collections » ainsi que ces implémentations.

Pour vous mettre dans le contexte, nous avons une pierre de plusieurs tonnes (créée informatiquement) que nous devons réduire en plusieurs fragments.

## Exercice 1

Code & résultats :

```
1 import stone.Stone;
2
3 public class Exo1 {
4     public static void main(String[] args) {
5         Stone smallStone = Stone.makeSmallStone(); // Instantie une petite stone avec la méthode makeSmallStone()
6         System.out.println(smallStone.toString());
7         smallStone.split(); // Casse smallStone en deux
8         System.out.println(smallStone.toString());
9     }
10 }
11
```

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot
pierre pesant 5,000kg et ayant un diamètre de 16cm
pierre pesant 2,635kg et ayant un diamètre de 15cm
```

Signature :

Function makeSmallStone() -> Stone

## Exercice 2

Code & résultats :

```
public class Exo2 {
    public static void main(String[] args) {
        Stone bigStone = Stone.makeBigStone();
        System.out.println(bigStone.toString());
        bigStone.split();
        System.out.println(bigStone.toString());
        int i = 0;
        while (bigStone.diameter() >= 5){
            i++;
            bigStone.split();
            System.out.println("split n°" + i + " " + bigStone.toString());
        }
    }
}
```

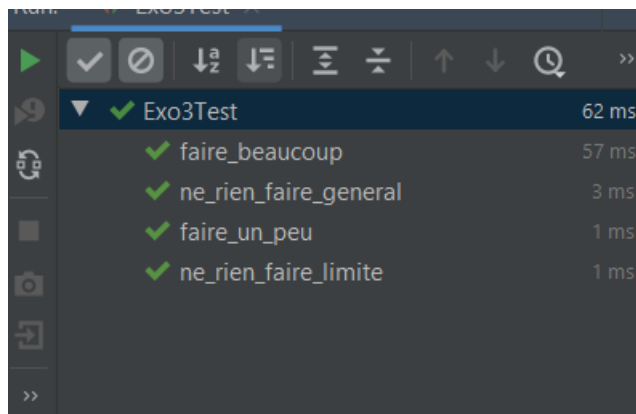
```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" -Did
pierre pesant 1000,000kg et ayant un diamètre de 90cm
pierre pesant 548,008kg et ayant un diamètre de 87cm
split n°1 pierre pesant 280,407kg et ayant un diamètre de 59cm
split n°2 pierre pesant 165,216kg et ayant un diamètre de 58cm
split n°3 pierre pesant 97,716kg et ayant un diamètre de 58cm
split n°4 pierre pesant 54,147kg et ayant un diamètre de 34cm
split n°5 pierre pesant 27,751kg et ayant un diamètre de 32cm
split n°6 pierre pesant 15,654kg et ayant un diamètre de 32cm
split n°7 pierre pesant 9,103kg et ayant un diamètre de 28cm
split n°8 pierre pesant 4,847kg et ayant un diamètre de 22cm
split n°9 pierre pesant 2,755kg et ayant un diamètre de 19cm
split n°10 pierre pesant 1,489kg et ayant un diamètre de 15cm
split n°11 pierre pesant 0,862kg et ayant un diamètre de 9cm
split n°12 pierre pesant 0,445kg et ayant un diamètre de 9cm
split n°13 pierre pesant 0,249kg et ayant un diamètre de 9cm
split n°14 pierre pesant 0,134kg et ayant un diamètre de 5cm
split n°15 pierre pesant 0,075kg et ayant un diamètre de 4cm

Process finished with exit code 0
```

## Exercice 3

Code & résultats :

```
public class Exo3Test extends AbstractGrinderTest {
|
|   @Override
|   protected Grinder makeGrinder() {
|       return new MyGrinder();
|   }
|
| }
}
```



Test Name	Duration
Exo3Test	62 ms
faire_beaucoup	57 ms
ne_rien_faire_general	3 ms
faire_un_peu	1 ms
ne_rien_faire_limite	1 ms

## Exercice 4

Code & résultats :

```
public class Exo4 {
    public static void main(String[] args) {
        Stone s = Stone.makeHugeStone();
        Grinder g = new MyGrinder();
        GrinderBench.benchmark(g, 4, s);
    }
}
```

```
Exo4
"C:\Program Files\Eclipse Adoptium\jdk-17.0.2.8-hotspot\bin\java.exe" -Didea.launcher.port=6428
Pierre initiale : pierre pesant 100000,000kg et ayant un diamètre de 412cm, diamètre visé : 4
Le test de performance commence (class MyGrinder).
Fin du test : 3928894 fragments obtenus en 1509742500 nanoseconds (1509 ms)

Process finished with exit code 0
```