

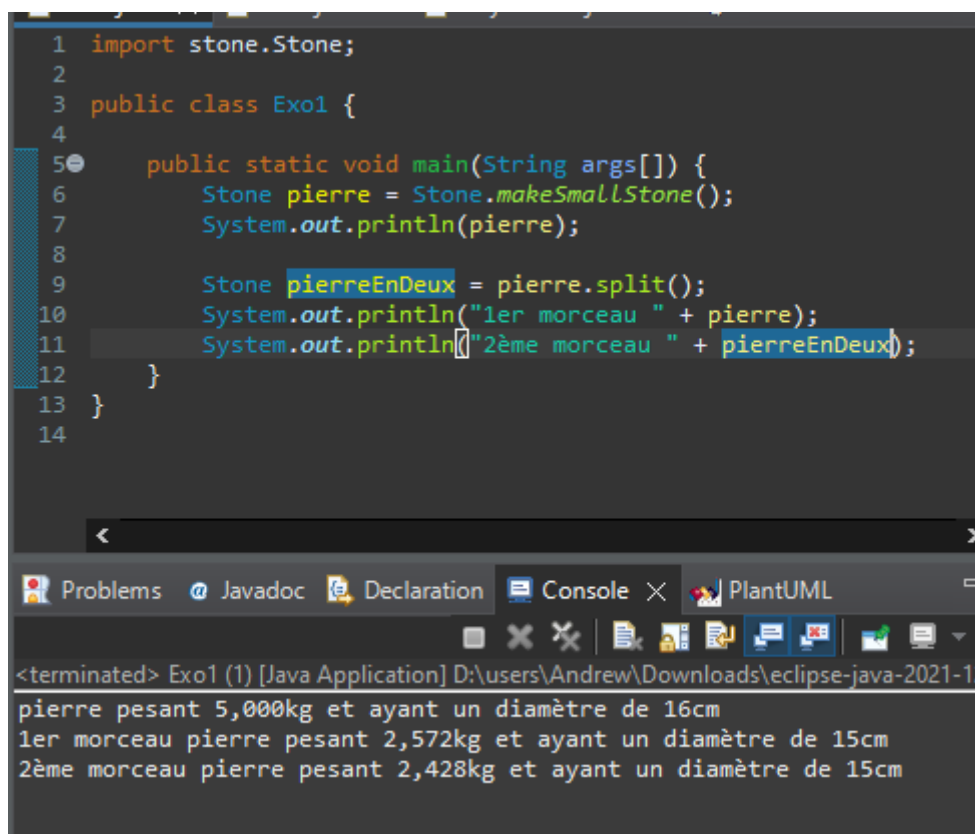
# Prog. Eff

## TP4 - Un TP au pénitencier

### Table des matières

Exercice 1.....	1
Exercice 2.....	2
Exercice 3.....	2
Exercice 4.....	2

### Exercice 1



```
1 import stone.Stone;
2
3 public class Exo1 {
4
5     public static void main(String args[]) {
6         Stone pierre = Stone.makeSmallStone();
7         System.out.println(pierre);
8
9         Stone pierreEnDeux = pierre.split();
10        System.out.println("1er morceau " + pierre);
11        System.out.println("2ème morceau " + pierreEnDeux);
12    }
13 }
14
```

Problems Javadoc Declaration Console × PlantUML

<terminated> Exo1 (1) [Java Application] D:\users\Andrew\Downloads\eclipse-java-2021-1  
pierre pesant 5,000kg et ayant un diamètre de 16cm  
1er morceau pierre pesant 2,572kg et ayant un diamètre de 15cm  
2ème morceau pierre pesant 2,428kg et ayant un diamètre de 15cm

La méthode Java implémentant l'opération a un effet de bord, quand nous toString() les 2 variables, la pierre normale et la pierre2 qu'on attribue pierre.split(), on voit que pierre a reçu l'effet de la méthode.

## Exercice 2

La classe de complexité est  $O(\log n)$ , car lorsque qu'on passe de big à huge pierre, on passe de 15~ opérations de split à 25~ opérations.

## Exercice 3

La classe de complexité de grind() est factorielle car la méthode est une méthode récursif, on a  $n/2^k$

## Exercice 4

```
1
2 import stone.Grinder;
3
4
5
6 public class Exo4Bench {
7
8     public static void main(String[] args) {
9         Stone s = Stone.makeHugeStone();
10        Grinder g = new MyGrinder();
11        GrinderBench.benchmark(g, 4, s);
12    }
13
14 }
```

<

Problems Javadoc Declaration Console X PlantUML

<terminated> Exo4Bench [Java Application] D:\users\Andrew\Downloads\eclipse-java-2021-12-R-win32-x86\_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.

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 : 3920468 fragments obtenus en 5940792300 nanoseconds (5940 ms)

```
7 public class MyGrinder implements Grinder {
8
9     @Override
10    public Collection<Stone> grind(Stone stone, int diameter) {
11        // TODO Auto-generated method stub
12        Collection<Stone> stones = new ArrayList<Stone>();
13        while(stone.diameter() > diameter) {
14            Stone splitStone = stone.split();
15            if(splitStone.diameter() > diameter) {
16                Collection<Stone> collStones = grind(splitStone, diameter);
17                stones.addAll(collStones);
18            }
19            else {
20                stones.add(splitStone);
21            }
22        }
23        stones.add(stone);
24        return stones;
25    }
26
27 }
28
```