JAVA PROBLÈMES TYPES

Calculer le PGCD

Remarque

PGCD est une abréviation pour: Plus grand commun diviseur, en anglais: Greatest common divisor ou GCD.

```
public int calculateGcd(int pA, int pB) {
    int gcd = 0;
    int min;
    if (pA < pB) {
        min = pA;
        gcd = pB;  //pour le cas où a=0 -> gcd=b
    } else {
        min = pB;
        gcd = pA;  //pour le cas où b=0 -> gcd=a
    }

    for (int i = 1; i <= min; i++) {
        if (pA % i == 0 && pB % i == 0) {
            gcd = i;
        }
     }
    return gcd;
 }
```

Algorithme d'Euclide par soustractions

```
 public int calculateGcdEuclidSubtraction() {
    int gA = a;
    int gB = b;
    if (gA == 0) {
```

```
        return gB;
    }

    while (gB != 0) {
        if (gA > gB) {
            gA = gA - gB;
        } else {
             gB = gB - gA;
        }
    }
    return gA;
}
```

Algorithme d'Euclide par divisions

```
public int calculateGcdEuclidDivision() {
    int gA = a;
    int gB = b;
    int t = 0;

    while (gB != 0) {
        t  = gB;
        gB = gA % gB;
        gA = t;
    }

    return gA;
}
```

isPrime()

```
public boolean isPrime(int pNumber) {
    int count = 0;
```

```
    for (int i = 1; i <= pNumber; i++) {
        if (pNumber % i == 0) {
            count++;
        }
    }
        return (count == 2);
}
```

Radpide

```
public boolean isPrime() {
    int i = 2;
    while ((i < number) && (number % i != 0)) {
        i = i + 1;
    }
    return i == number;
}
```

Plus radpide

```
public boolean isPrime() {
    int n = Math.abs(number);
    int i = 2;
    while (i <= Math.sqrt(n) && n % i != 0) {
        i++;
    }
    return n == 2 || n % i != 0 && n > 1;
}
```

Génération d'une suite de nombres aléatoires entiers

```java
public void printRandomNumbers(int pN, int pMin, int pMax){
    for (int i = 1; i <= pN; i++) {
        int currentNumber = (int)(Math.random() * (pMax - pMin +1
        System.out.print(currentNumber+"  ");
        if(i%20==0) {
            System.out.println();
        }
    }
    System.out.println();
}
```

```java
public class NombresAleatoire {

    private ArrayList< Double> alNumbers = new ArrayList<>();

    public void randomFill(int pN, double pMin, double pMax) {
        alNumbers.clear();
        double currentNumber;
        for (int i = 0; i < pN; i++) {
            currentNumber = Math.random() * (pMax - pMin) + pMin;
            alNumbers.add(currentNumber);
        }
    }

    public void print() {
          for (int i = 0; i < alNumbers.size(); i++) {
            System.out.print(alNumbers.get(i)+"  ");
        if((i+1)%5==0) System.out.println();
          }
          System.out.println();
    }

}
```

Trouver un élément dans une liste

```
public int findLast(String pNeedle) {
    int result = -1;
    for (int i = 0; i < alMyList.size(); i++) {
        if (alMyList.get(i).getAttribute().equals(pNeedle)) {
            result = i;
        }
    }
    return result;
}
```

```
public int findFirst(String pNeedle) {
    int result = -1;
    for (int i = alMyList.size() - 1; i >= 0; i--) {
        if (alMyList.get(i).getAttribute().equals(pNeedle)) {
            result = i;
        }
    }
    return result;
}
```

```
 public int findFirst(String pNeedle) {
    int result = -1;
    for (int i = 0; i < alMyList.size(); i++) {
        if (alMyList.get(i).getAttribute().equals(pNeedle) && res
            result = i;
        }
```

```
    }
    return result;
}
```

```
 public int findFirst(String pNeedle) {
    int result = -1;
    int i = 0;
    while (i < alMyList.size() && result == -1) {
        if (alMyList.get(i).getAttribute().equals(pNeedle)) {
            result = i;
        }
        i++;
    }
    return result;
}
```

Trouver le maximum dans une liste

```
public int getMax() {
    if (alListOfNumbers.size() > 0) {
        int max = alListOfNumbers.get(0);
          int current;

            for (int i = 1; i < alListOfNumbers.size(); i++) {
                current = alListOfNumbers.get(i);
                if (max < current) {
                    max = current;
                }
            }

        return max;
    } else {
        return -1;
```

```
    }
}
```

```
public int getMin() {
    if (alListOfNumbers.size() > 0) {
        int min = alListOfNumbers.get(0);
          int current;

            for (int i = 1; i < alListOfNumbers.size(); i++) {
                current = alListOfNumbers.get(i);
                if (min > alListOfNumbers.size current) {
                    min = current;
                }
            }

        return max;
    } else {
        return -1;
    }
}
```

Calculer la somme des nombres d'une Liste

```
public int getSum() {
    int sum = 0;
    for (int i = 0; i < alListOfNumbers.size(); i++) {
        sum = sum + alListOfNumbers.get(i);
    }
    return sum;
}
```

```
public double getAverage() {
    if (alMarks.size()==0)
    //La méthode retourne 0 si la liste est vide
       return 0;
    else
       return (double) getSum() / alListOfNumbers.size();
}
```