

# TP 9 VARI 1

## Démarrer une console/terminal

**Rappel :** `man xyz` affiche le manuel d'une commande `xyz` (remplacer `xyz` par votre vraie commande). Taper `/abc` et `Entrée` pour chercher `abc` dans le manuel. Pour quitter le manuel, taper "q".

**Exercice 1** Taper `man du` pour trouver le manuel de la commande `du`.

- Quel est l'objectif de cette commande `du` ?
- À quoi sert l'option `--max-depth` ?

**Exercice 2** Taper la commande suivante pour afficher l'espace disque utilisé par chaque dossier dans le dossier courant

```
du -h --max-depth=1
```

Que se passe si on remplace `max-depth=1` avec

```
du -h --max-depth=2 ?
```

Quel est le résultat de la commande suivante ?

```
du -h --max-depth=1|du -h
```

**Exercice 3** Taper la commande suivante pour créer un dossier `tp9vari1` pour y mettre vos programmes.

```
mkdir tp9vari1
```

Se placer dans ce dossier :

```
cd tp9vari1
```

Par la suite taper

```
echo "salut"> toto.txt
```

pour écrire « salut » dans un fichier `toto.txt`. Visualiser ce fichier à l'aide d'une commande `cat`. Ouvrir le gestionnaire de fichiers et vérifier que vous trouvez le texte « salut » dans ce fichier.

## Programmation : vous pouvez choisir de faire les exos non graphiques sous Java ou Processing

**Exercice 1** Écrire une fonction `doubler(int)` qui renvoie le double d'un nombre entier. Remplir le code ci-après pour le faire fonctionner ; vous pouvez choisir de travailler soit sous `processing` soit sous `java`.

PROCESSING :

```
... doubler (int x){
    ....
}
void setup(){
    int salaire = 1000;
    int nouveauSalaire = doubler(salaire);
    println("J'ai doublé le salaire et mon nouveau salaire est "+nouveauSalaire);
}
```

JAVA :

```
class Exo1{
    static int doubler (int x){
        ....
    }
    public static void main(String [] args){
        int salaire = 1000;
        int nouveauSalaire = doubler(salaire);
        .... println("J'ai doublé le salaire et mon nouveau salaire est "+
            nouveauSalaire);
    }
}
```

**Exercice 2** Écrire une fonction qui renvoie le cube d'un float. Ce programme devrait afficher 1030.301.

```
... cube (...) {
    ....
}
void setup(){//ou main(..) sous java
    float lecube=cube(10.1);
    println(lecube);
}
```

**Exercice 3** Remplir la fonction `calcIntérêts` ci-dessous pour calculer les intérêts gagnés au taux `t` sur un capital `cap`. Par exemple, `calcIntérêts(1000,0.05)` devrait renvoyer 50. Remplir le programme ci-après.

```
... calcInterets (float cap, float t){
    ....
    return ...
}
void setup(){
    float x ;
    x = calcInterets(1000,0.05);

    println("Capital_final_=" ...);
}
```

**Exercice 4** Soit le programme ci-après. Dire ce qu'il affiche sans le faire tourner. Dans une deuxième étape, taper le code et vérifier votre réponse.

```
int detMax(int a, int b){
    if(a>b)
        return a;
    return b;
}
void setup(){
    int a=9, b=8;
    println(detMax(b,5));
    println(detMax(5,a));
}
```

**Exercice 5** Écrire une fonction `detMaxTab(int[])` qui appelle 2 fois la fonction `detMax(...)` ci-dessus pour trouver le maximum d'un tableau de 3 cases. Écrire un fonction similaire `detMinTab(int[])` qui détermine le minimum. Ajouter une fonction `notesValides(int[])` qui renvoie `true` si les notes d'un tableau de 3 cases sont comprises entre 1 et 20 et `false` sinon. Commencer avec :

```
boolean notesValides(int [] notes){
    if((detMinTab(notes)>0)&& ...)
        return true;
    ... ..
```

**Exercice 6** Le programme ci-après permet de tracer des lignes à des positions aléatoires. **Rappel :** La fonction `draw()` est appelée de manière répétitive en continu. L'appel de fonction `random(600)` renvoie un float aléatoire entre 0 et 600. Modifier le programme pour le faire afficher des ellipses de couleurs différentes (utiliser `fill(random(255),...,...)`). Les tailles des ellipses doivent être inférieures à 300.

```
void setup(){
    size(600,600);
}
void draw(){
    //tracer une ligne du pixel x au pixel y,
    line(random(600),random(600), //pixel x = coordonnées aléatoires <600
        random(600),random(600)); //pixel y = coordonnées aléatoires <600
}
```

**Exercice 7** Écrire une méthode (c.-à.-d. une fonction qui renvoie `void`) pour afficher un nom de 4 lettres. L'argument de la méthode est un tableau avec 4 cases de type `char` (lettre). Corriger une erreur dans le programme ci-après. Remplir le reste du code pour le faire afficher "Toto". **Note :** Utiliser la fonction `print(...)` pour afficher un caractère sans sauter une ligne.

```
.... afficherNomDe4Lettres(char [] nomDe4Lettres){
    ....
}
void setup(){
    char [] nom4Lettres = new char [4]; //tableau de 4 lettres
    nom4Lettres[0] = 'T';
    afficherNomDe4Lettres(nomDe4Lettres); //corriger une erreur ici
}
```