TP 9 VARI 1

1 Commandes linux à taper dans une console/terminal

Exercice 1 Taper une commande dans le terminal pour créer un dossier où vous allez déposer tous vos programmes d'aujourd'hui. Indication : utiliser `mkdir` (un acronyme pour l'anglais `make directory`), par exemple, ainsi :

```
mkdir tp10_22novembre
```

Exercice 2 Taper une commande `cd` (un acronyme pour `change directory`) pour se placer dans le dossier crée au premier exercice.

Exercice 3 Taper les commandes ci-dessous pour créer trois dossiers.

```
cd dossier1
cd dossier2
cd dossier3
```

Exercice 4 Taper une des commandes ci-après pour se placer dans un de ces trois dossiers :

```
mkdir dossier1
mkdir dossier2
mkdir dossier3
```

Exercice 5 Créer plusieurs fichiers dans le dossier choisi, avec des commandes comme par exemple :

```
touch abc
touch abd
touch mmm
```

Exercice 6 Taper `cd ../../` pour revenir dans le dossier personnel. Ensuite, taper la commande suivante pour chercher un fichier nommé `abc` dans votre dossier perso.

```
find . -name abc
```

Exercice 7 La commande ci-dessous permet de chercher tous les fichiers dont le nom commence avec `gimp` dans le dossier `/usr/`. Modifier cette commande pour chercher tous les fichiers dont le nom commence avec `javac`.

```
find /usr -name gimp*
```

Exercice 8 Taper la commande `pwd` pour connaitre/afficher le dossier courant. Essayer de mémoriser cette commande utile, c'est un acronyme pour `print working directory`.

2 Processing

Exercice 1 Soit le code ci-après. Corriger deux erreurs de compilation et une erreur de logique !

```
1  int note1;note2;          //déclaration de variables
2  note1=19;                 //affectation de valeur
3  note2=14;
4  note3=14;
5  int min=note3;            //on commence le calcul de la note minimale
6  if(min>note2)
7      min=note2;
8  if(min>note3)
9      min=note3;
10 if(min<10)
11     println("échec: vous avez au moins une note inférieure à 10");
12 else
13     println("succès: vous avez validé toutes les UEs avec des notes >=10");
14 int note3;
```

Exercice 2 Modifier le code de l'exercice précédent pour le faire utiliser un tableau `notes` avec trois cases. La déclaration et l'initialisation du tableau sont données ci-après, n'hésitez pas à regarder les exemples du cours (7) sur les tableaux. Voici comment déclarer le tableau :

```
int[] notes = new int[3];
notes[0] = 19;
notes[1] = 14;
notes[2] = 14;
```

Exercice 3 Remplir (au début) le code ci-après pour afficher un cercle rouge centré au pixel (50,50).

```
int[] x = new int[..à remplir..];
.....//à remplir
tab[3]=40;
fill(250,0,0);
ellipse(x[0],x[1],x[2],x[3]);
```

Exercice 4 La fonction `quad(...)` permet de tracer un quadrilatère. Remplir le programme ci après pour tracer un quadrilatère avec les sommets positionnés comme indiqués dans le tableau `x`.

```
int x[] ....//valeurs à remplir
fill(250,0,0);
quad(x[0],x[1],x[2],x[3],
     x[4],x[5],x[6],x[7]);
```

Exercice 5 Utiliser `fill(...)` et `stroke()` pour faire le contour du quadrilatère en rouge et le fond en bleu.

Exercice 6 Utiliser deux appels à la fonction `line(...)` pour tracer les diagonales du quadrilatère.

Exercice 7 Écrire une fonction qui prend un argument `x` de type tableau. On dit que l'en-tête ou signature de la fonction est :

void quadAvecDiag(**int**[] x)

L'argument `x` est un tableau de 8 cases. Cette fonction doit tracer un quadrilatère avec les deux diagonales aux coordonnées indiqués par `x`, comme à l'exercice précédent. Testez votre fonction dans `setup()`

Exercice 8 Écrire une fonction `puissance(...)` qui prend deux arguments x et n de type `int` et qui renvoie la valeur x^n. Le résultat sera un `int`. L'en-tête (ou la signature) de la fonction est :

int puissance(**int** x, **int** n)

Attention : si n est inférieur à zéro, il faut renvoyer -1.

Exercice 9 Écrire une fonction d'en-tête `void polygone(int n, int[] x)` qui reçoit comme arguments une valeur `n` et un tableau x. Tracer un polygone avec `n` sommets situés ainsi : le premier sommet est placé à $(x[0], x[1])$, le 2ème à $(x[2], y[3])$, le 3ème à $(x[4], y[5])$, ..., le dernier à $(x[2n-2], x[2n-1])$.

Exercice 10 Écrire une fonction d'en-tête `void polygoneDiags(int n, int[] x)` qui appelle d'abord la fonction `polygone(...)` de l'exercice précédent. Ensuite, il faut tracer en rouge toutes les diagonales du polygone.

Exercice 11 Écrire une fonction `grossir(int[] x)` qui fait élargir le polygone : à chaque itération, chaque sommet $(x[2i], x[2i+1])$ s'éloigne d'un pixel du centre de la toile. Appeler cette fonction dans `draw()` suivie de `polygone(x)` ; fixez le `frameRate` à 3 appels par seconde. Testez votre méthode sur un polygone avec `n=5` pour obtenir un pentagone qui s'élargit. Il faut s'arrêter dès qu'on touche un bord, comme dans la vidéo ci-après : `youtu.be/ukT33LS2JfA`

Exercice 12 Soit les programmes `Java` ci-après. Vous observez que les deux premières lignes sont très similaires. Essayer de déterminer ce que ces programmes affichent sans les exécuter. Trouver une petite erreur/oubli dans chaque programme : dans le premier programme il manque un espace et dans le 2ème la division ne calcule pas la moyenne correctement, car il manque des parenthèses.

```
class Prog1{
   public static void main(String[] args){
      System.out.println("Salut Toto");
      System.out.print("C'est mon premier");
      System.out.println("programme Java.");
   }
}
```

```
class Prog2{
   public static void main(String[] args){
      float x = 10, y=20;
      float moyenne = x+y/2;
      System.out.print("moyenne+"+moyenne);
   }
}
```

Exercice 13 Utiliser un éditeur de texte pour taper ces deux programmes. Dans les salles TP, vous pouvez utiliser `kate` ; il faut taper :

```
kate Prog1.java&
```

Pour exécuter ce programme, il faut se placer dans son dossier via le terminal et taper :

```
javac Prog1.java
java Prog1
```