TP 7 VARI 1

1 Commandes Linux

Démarrer une console/terminal en utilisant une des méthodes indiquée sur le site ci-dessous, comme au TP précédent.
`cedric.cnam.fr/~porumbed/vari1/term/`

Exercice 1 Taper une commande dans le terminal pour créer un dossier. Si le terminal tourne sur votre machine, vous allez pouvoir y déposer tous vos fichiers d'aujourd'hui. Indication : utiliser `mkdir` (un acronyme pour l'anglais `make directory`), n'hésitez pas à regarder les exemples à la fin du Cours sur le systèmes sur le site :
`cedric.cnam.fr/~porumbed/vari1`

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

Exercice 3 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 4 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 dernier en ligne (sur les tableaux).

```
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 de coordonnées (50,50).

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

Exercice 4 La fonction `quad(...)` permet de tracer un quadrilatère. Tester par exemple un programme qui contient une seule ligne : `quad(0,0,100,100,50,90,10,50);`. Remplir le programme ci après pour faire un nouveau quadrilatère, avec les sommets positionnés comme vous le souhaitez.

```
int x[] ....//coordonnées x à remplir
int y[] ....//coordonnées y à remplir
fill(250,0,0);
quad(x[0],y[0],x[1],y[1],
     x[2],y[2],x[3],y[3]);
```

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 Soit deux tableaux x et y de 6 cases initialisés au début du programme, comme dans le code ci-dessous. Tracer un hexagone rempli avec les sommets/coins aux coordonnées indiquées par x et y, c. à. d., le premier sommet est placé à $(x[0], y[0])$, le 2ème à $(x[1], y[1])$, le 3ème à $(x[2], y[2])$. etc.
Indication : utiliser des appels à la fonction `quad(...)`.

```
size(500,500);
int x[] = {100, 200, 400, 500, 400, 200};
int y[] = {273, 100, 100, 275, 446, 446};
```

Exercice 8 On considère une somme d'argent `fortune` déposée sur cinq ans sur un livret rémunéré avec des intérêts. Les taux d'intérêts sur les cinq ans sont donnés dans un tableau d'entiers `taux` avec 5 cases. Par exemple, si `taux`[0]=5 alors le taux d'intérêt sur la première année est de 5%=0.05. Calculer la somme finale générée à la fin des 5 années. Les intérêts sont **capitalisés**, c.à.d., à la fin de chaque année, **les intérêts générés** sont **ajoutés au capital** pour produire de nouveaux intérêts.
Note : Vous pourriez avoir besoin de transformer les entiers en réels. Pour cela on utilise un `cast` : si x est une variable entière et y est réelle, alors `y=(float)x/100` réalise la conversion (ou le `cast`) entier→réel avant de faire le calcul. Si on ne faisait pas ce `cast`, l'affectation `y=x`/100 serait évaluée à 0 pour tout $x < 100$.

Exercice 9 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 parentheses.

```
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 10 Installer `Java` sur votre machine et utiliser un éditeur de texte pour taper ces deux programmes. Si vous utilisez la machine virtuelle Antix que je vous ai donné, `Java` est déjà installé et l'éditeur s'appelle `geany`. Pour le lancer, il faut taper `geany Prog1.java&` dans un terminal. Pour exécuter un programme, il faut sauvegarder le code et revenir dans le terminal pour taper :

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