TP 9 VARI 1 Révisions

Informations techniques PC Suse :

(a) Pour démarrer une session : utilisateur **licencep** et mot de passe **7002n***.

(b) Pour démarrer *Processing* : clic sur `l'icône lézard` en haut à droit → Dévelopemment → Processing.

(c) Pour démarrer un *terminal* : `l'icône lézard` → Terminal → Konsole.

(d) Pour ouvrir un gestionaire/navigateur de fichiers : `l'icône lézard` → Système → Dolphin

(e) Pour modifier un fichier, clic droit sur le fichier → Ouvrir avec Kate

1 Commandes terminal et réseaux

Aller sur le site `cedric.cnam.fr/~porumbed/vari/`, **télécharger le cours 9** et parcourir les diapos, regarder les programmes et essayer de tout comprendre dans le moindre détail.

Exercice 1 Taper la commande suivante dans un terminal (à démarrer grâce aux instructions au point (c) ci-dessus), en faisant attention aux espaces.

```
time  wget cedric.cnam.fr/~porumbed/1giga
```

Elle télécharge un fichier d'1 GB≈1000MB (commande `wget`) et affiche le temps utilisé pour cela, voir rubrique `real` (affiché par la commande `time` devant). **Calculer la vitesse (débit) du réseau** $\frac{1000\text{MB}}{\texttt{temps (sec)}}$, ex., si cela prend 9 secondes on a $\frac{1000}{9}$=111MBs. Vous pouvez faire le calcul avec une calculette.

Exercice 2 Exécuter la commande suivante pour faire une copie du fichier. Remarquer la vitesse du disque dur !

```
time cp 1giga 1gigacopie
```

Exercice 3 Taper la commande suivante.

```
ls  -l  -S
```

Elle permet d'afficher les fichiers du dossier courant. Quel est le rôle de l'option `-S`, elle permet de trier les fichiers selon quel critère ?

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

Exercice 4 Taper la commande suivante pour visualiser les fichiers d'une taille supérieure à 900 MégaOctets. En principe, vous allez trouver au moins les fichiers `1giga` et `1gigacopie`.

```
find . -size +900M
```

Quelle commande permettrait d'afficher tous les fichier de taille supérieure à 100 MégaOctets ?

Exercice 5 Ajouter à la commande de l'exercice précédent l'option `-delete`, pour effacer les fichiers trouvés, attention !

Exercice 6 Taper `ifconfig` pour récupérer l'adresse IP de votre machine, associée à l'interface `br0`. Pouvez vous trouver la même information grâce à la commande `ip addr` .

Exercice 7 Travailler en binôme sur deux machines A et B. La machine A lance "`netcat -l 10000`" pour ouvrir un serveur TCP qui écoute le port 10000 (option "`-l`"=listen). La machine B lance "`netcat` IP_A `10000`", où IP_A est l'adresse IP de A. Vous obtenez une connexion TCP et vous pouvez communiquer comme dans un programme de messagerie instantanée.

Exercice 8 Taper la commande ci-après pour télécharger le fichier source `Toto2.java`.

```
wget  cedric.cnam.fr/~porumbed/Toto2.java
```

Regarder le contenu du fichier `Toto2.java` à l'aide de la commande `cat`, ou avec un éditeur comme `kate`. Pour **compiler** le programme Java, taper :

```
javac   Toto2.java
```

Et pour l'**exécuter :**

```
java   Toto2
```

Exercice 9 Aller sur le site ci-dessus et télécharger la machine virtuelle (fichier vdi) :

`http://cedric.cnam.fr/~porumbed/vari1/vbox/`

Suivre les indications pour faire tourner cette machine virtuelle (VM) ; vous pouvez faire les prochains exercices Processing dans cette VM.

2 Fonctions et boucles

Exercice 1 Écrire une fonction `boolean revenuImposable(float revenuFiscalRef)` qui renvoie `true` si le revenu fiscal de référence `revenuFiscalRef` est supérieur à 9710 euros ou `false` sinon. Il faut donc renvoyer une valeur de type `boolean` qui indique si le revenu est imposable ou pas. Remplir le code ci-après pour le faire fonctionner. Lancer le programme plusieurs fois pour tester plusieurs valeurs aléatoires de revenu, ligne 4.

```
1 boolean revenuImposable(... ... ...)
2     ...
3 void setup(){
4     float revenuNet = random(90000); //le revenu fiscal de référence
5     boolean imposable;
6     imposable = revenuImposable(revenuNet);
7     if(imposable){
8         println("Avec un revenu de "+revenuNet+" vous payez des impôts");
9     }else{
10        println("Avec un revenu de "+revenuNet+" vous ne payez pas d'impôt");
11    }
12 }
```

Exercice 2 Écrire un fonction `float calculerImpots(float revenuFiscalRef)` qui renvoie le montant des impôts à payer pour un revenu fiscal de référence `revenuFiscalRef`. Il s'agit d'un impôt progressif, défini sur 2 tranches :

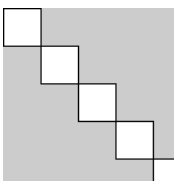
- Pas d'impôt pour la tranche $0 - 9710$ euros.
- 14% pour tout gain au delà de 9710 euros.

Par exemple, pour un revenu de 10000 euros, l'impôt vaut $0.14 \times (10000 - 9710)$.

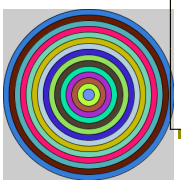
Faire fonctionner le code ci-après.

```
.. calculerImpots(.. revenuFiscalRef){
    ...
}
void setup(){
    float revenuNet = random(90000);
    float impots =
            calculerImpots(revenuNet);
    println("Votre revenu="+revenuNet);
    println("Vos impôts="+impots);
}
```

Exercice 3 Tracer 20 rectangles de taille 30×30 le long de la diagonale d'une toile de taille 600×600. La figure à droite montre le début de la diagonale, il faut faire 20 rectangles.



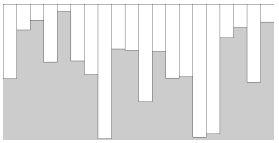
Exercice 4 Tracer 15 cercles de même centre et de rayons $20, 40, 60, \ldots, 300$. Chaque cercle aura une couleur aléatoire. Vous aller avoir besoin de commencer avec $i = 15$ et finir avec $i = 1$; utiliser i- - au lieux de i++.



Exercice 5 Écrire une méthode `tracerBarre(int colonne, float hauteur)` qui permet de tracer un rectangle de largeur 50 et hauteur `hauteur` à la position (50×`colonne`, 0). Plus précisément, le coin en haut à gauche est situé aux coordonnées (50×`colonne`, 0) et la taille du rectangle est 50×`hauteur`. Faire fonctionner le code suivant.

```
void setup(){
    size(1500,500);
    for(int i=0;i<30;i++)
        tracerBarre(i,random(500));
}
```

Résultats final attendu :



Exercice 6 Écrire une méthode `histogramme(float[] vals)` pour tracer un histogramme associé aux valeurs `vals`. On considère que `vals` est un tableau de 30 éléments `float`. La méthode devrait faire 30 fois appel à `tracerBarre(...)`. Faire fonctionner le code ci-après.

```
void setup(){
    size(1500,500);
    float[] valeurs = new float[30];
    for(int i=0;i<30;i++)
        valeurs[i] = i*10;//ou random
    histogramme(valeurs);
}
```

Affichage attendu :