TP 11 noté

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

(b) Pour démarrer *Processing* : clic sur `la tête de coméléon` en haut à droite → Dévelopemment → Processing.

(c) La page VARI1 : `cedric.cnam.fr/~porumbed/vari1/`

(d) Pour ouvrir un gestionaire/navigateur de fichiers : clic sur `la tête de caméléon` → Système → Dolphin.

Exercice 1 **(4 points)** Soit le code ci-après qui a pour objectif de calculer la note maximale. Corriger toutes les erreurs. Sauvegarder le programme corrigé dans un fichier `Exo1.pde`

```
float note1,note2,note3,note4;
note1=random(20);
note2=random(20);
note3=random(20);
note4=random(20);
float min = note1;
if(max<note2)
    max=note2;
if(max<note3)
    max=note3;
if(max<note4)
    min = note4;
printf("la note maximale est :"+max);
```

Exercice 2 **(4 points)** Copier le programme précédent dans un fichier `Exo2.pde` ; modifiez ce nouveau fichier `Exo2.pde` pour le faire utiliser un tableau de 4 cases à la places des variables `note1`, `note2`, `note3`, `note4`.

Exercice 3 **(3 points)** Écrire dans un fichier `Exo3.pde` une fonction `moyenne(float[] tab)` qui prend comme argument un tableau de `float` et qui renvoie la valeur **moyenne. Rappel :** le nombre de cases dans le tableau est `tab.length`. L'en-tête (ou la signature) de la fonction est :

```
float moyenne(float[] tab)
```

Attention : vous pouvez obtenir 2 points sur 3 si vous considérez que le tableau comporte trois cases.

Exercice 4 **(2 points)** Écrire une méthode (fonction qui renvoie `void`) `triangleBleu(int x0,int y0,int x1, int y1, int x2, int y2)` pour tracer un triangle à l'aide de 3 appels à la fonction `line(...)`. Les sommets sont indiqués par les coordonnées $(x0, y0)$, $(x1, y1)$, $(x2, y2)$. Remplir le code ci-dessous pour le faire fonctionner.

```
voi_ triangleBleu (int x0,int y0,int  x1, int y1, int x2, int y2){
    fill(...)
    etc, etc, à remplir
    ....
}
void setup(){
    size(500,500);
    triangleBleu(10,10,100,50,50,100);
    triangleBleu(10,10,400,100,100,400);
}
```

Exercice 5 (2 points) Écrire une méthode `triangleBleuTab(int[] x, int[]y)` avec le même objectif que la méthode précédente, mais avec des arguments de type tableau. Remplir le code à droite pour le faire fonctionner. Sauvegardez le programme dans un fichier `Exo5.pde`

```
... triangleBleuTab(int[] x,int[] y){
    ....
    //ajouter trois appels line(...)
    ...
}
void setup(){
    size(500,500);
    int[] ax = {10, 400, 50};
    int[] ay = {10, 200, 80};
    int[] bx = {40, 300, 80};
    int[] by = {90, 400, 50};
    int[] cx = {80, 440, 80};
    int[] cy = {10, 490, 50};
    triangleBleuTab(ax,ay);
    triangleBleuTab(bx,by);
    triangleBleuTab(cx,cy);
}
```

Exercice 6 (1 pt) Écrire une fonction `perimetreTriangle(int[] x,int[] y)` qui renvoie le périmètre d'un triangle. Il faut déterminer les longueurs des trois côtés.
Pour cela, vous pouvez utiliser le théorème de Pythagore, par exemple :
float `len01`$= sqrt((x[1]-x[0])^2+(y[1]-y[0])^2)$.

Exercice 7 (2 points) Modifier le code ci-après pour augmenter la vitesse de rotation/animation. Sauvegardez le programme dans un fichier `Exo7.pde`

```
float angle = 0;      //variable globale
void setup(){
    size(600,600);
}
void draw(){
    background(200,200,200);
    angle = angle + 2*PI/100;

    translate(300,300);
    rotate(angle);
    ellipse(0,0,400,40);
}
```

Exercice 8 (1 point) Écrire une méthode `pentaTab(int[] x, int[] y)` qui reçoit les tableaux `x` et `y` comme arguments et qui permet de *tracer un pentagone*. Il faut généraliser `triangleBleuTab` pour relier le point $(x[0], y[0])$ au point $(x[1], y[1])$, ensuite relier $(x[1], y[1])$ à $(x[2], y[2])$, $(x[2], y[2])$ à $(x[3], y[3])$, etc.

Exercice 9 (1 point) Écrire une méthode `polygone(int[] x, int[]y)` qui généralise la méthode `pentaTab(...)` de l'exercice précédent. Elle devrait tracer un polygone de taille arbitraire. Initialiser dans la méthode `setup()` des tableaux avec des valeurs aléatoires (via, ex., `(int)random(...)`) et appeler `polygone(...)`.
Indication : vous allez avoir besoin de déterminer la longueur des tableaux x et y et vous pouvez utiliser `x.length`.

Déposer vos fichiers dans une archive ZIP en utilisant l'URL :
cedric.cnam.fr/~porumbed/vari1/tp/