TP 10 VARI 1

Exercice 1 Taper la commande ci-après pour télécharger le fichier source `Toto.java`.

```
wget  cedric.cnam.fr/~porumbed/vari1/Toto.java
```

Taper `ls` pour vérifier que vous avez un nouveau fichier `Toto.java` dans le dossier courant. Ensuite, vous pouvez afficher le contenu du fichier `Toto.java` à l'aide de la commande `cat Toto.java`. Pour **compiler** le programme Java, taper :

```
javac   Toto.java
```

Et pour l'**exécuter :**

```
java   Toto
```

Exercice 2 Ouvrir ce fichier avec la commande ci-après (le & sert à lancer la commande en « background », c.à.d., en arrière-fond pour ne pas bloquer le terminal).

```
kate Toto.java &
```

Modifier le programme pour le faire afficher « Salut tout le monde ». Compiler et exécuter ce nouveau programme java.

Exercice 3 Modifier le programme précédent et introduire une boucle `for` qui permets d'afficher 9 fois le texte `Salut tout le monde!!!`.

Exercice 4 Modifier le programme précédent. Ajouter une boucle `for` pour déterminer la valeur minimale d'un tableau de 9 entiers. Modifier le programme ci-après pour le faire afficher la valeur minimale du tableau, c.-à.-d. 1.

```
class Toto{
    public static void main(String[] args){
        int [] a={9,5,1,4,3,12,9, 10, 1000};
        int min = a[0];
        //insérer ici le code pour calculer min
        System.out.println(min);
    }
}
```

Fonctions sous Processing

Exercice 5 Démarrer `processing`. Écrire une méthode (fonction qui renvoie `void`) `triangleRouge(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-après pour le faire fonctionner.

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

Exercice 6 Écrire une méthode `triangleRougeTab(int[] x, int[]y)` avec le même objectif que la méthode précédente, mais avec des arguments de type tableau. Remplir le code ci-après pour le faire fonctionner.

```
... triangleRougeTab(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};
    triangleRougeTab(ax,ay);
    triangleRougeTab(bx,by);
    triangleRougeTab(cx,cy);
}
```

Exercice 7 Modifier le code ci-après pour augmenter la vitesse de rotation/animation.

```
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 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 `triangleRougeTab` 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 Écrire une boucle `for` pour tracer 35 rectangles de taille 200×100 placés à des positions aléatoires.
Indication : `rect(random(100),random(100),50,20)` permet de tracer un rectangle de taille 50×20 à une position aléatoire avec les deux coordonnées entre 0 et 100, voir le programme ci-après.

```
void setup(){
    rect(random(100),random(100),50,20)
}
```

Exercice 10 Modifier la méthode `pentaTab(int[] x, int[]y)` pour afficher le pentagone à l'aide d'une boucle `for`. On suppose que les tableau `x` et `y` ont une longueur de 5.

Exercice 11 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`.