LABORATOIRE 4 | *Informatique 1*
FONCTIONS

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Objectifs du laboratoire (4 périodes)

1. Le but de ce laboratoire est de vous habituer à utiliser des fonctions existantes ainsi qu'à écrire vos propres fonctions.
2. La durée estimée pour réaliser ce laboratoire est de quatre périodes.
3. Vous pouvez trouver cette donnée sous forme électronique se trouve sur le site web du cours (http://inf1.begincoding.net/). Vous y trouverez également le corrigé de ce labo à la fin de celui-ci.

Partie 1 - Fonctions

Dans cette première partie, on vous demande d'implémenter et tester avec différentes fonctions.

Tâche 1

1. Créez un nouveau projet *Eclipse* pour l'intégralité du labo nommé *Lab4*.
2. Ajoutez-y une fonction nommée `test` et qui retourne le carré d'un nombre passé en argument.
3. Ajoutez-y une fonction nommée `smallest` qui retourne le plus petit de trois nombres entiers passés en argument.
4. Ajoutez-y une fonction affichant le nombre d'occurrences d'une lettre dans une chaîne de caractères. Pour cela, téléchargez la classe `StringFunctions` qui contient deux fonctions, l'une permettant d'extraire une lettre d'une chaîne de caractères et l'autre permettant de retourner le nombre de lettres dans une chaîne.
5. Ajoutez-y une fonction prenant un `String` en argument et retournant le `String` correspondant inversé. Par exemple : « `hello` » → « `olleh` ». Utilisez les fonctions données dans `StringFunctions`.
6. Utilisez les fonctions de `StringFunctions` dans une boucle de manière à afficher combien de fois apparaît chaque lettre de l'alphabet dans la phrase «*This is a nice world* ».
7. *[Un peu plus difficile]* Ajoutez-y une fonction de mise à la puissance (qui ne doit pas utiliser `Math.pow`). Testez votre fonction. Attention aux puissances négatives et à puissance 0. Pour rappel, il n'est pas permis d'avoir des puissances non entières.

Partie 2 - Dotted Flag

Le but de cet exercice est de dessiner à l'aide de *FunGraphics* un drapeau composé de ronds rouges sur fond blanc, tel que visible dans l'image suivante :

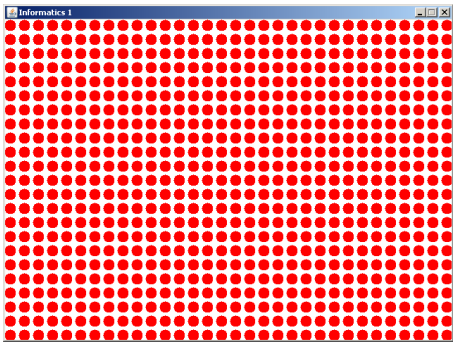


Figure 1 - Dotted Flag

Pour ce faire, vous pouvez partir d'un code existant qui dessine un disque rouge (rappel : un disque est un cercle *plein*). Le but sera de créer une méthode à partir de ce code et d'ensuite utiliser cette fonction pour dessiner le drapeau.

Tâche 2

1. Dans le projet, à l'intérieur du répertoire *src,* insérez les fichiers qui se trouvent dans l'archive *task.zip* que vous devez télécharger sur le site.
2. Le fichier *DottedFlag.java* contient le code de base qui permet de dessiner un disque rouge sur fond blanc (comme dans la dernière question optionnelle du laboratoire précédent).

3. Etudiez la méthode nommée *drawDisc* ayant comme paramètres les coordonnées x et y du centre du cercle ainsi que le rayon du cercle pour dessiner les cercles.
4. Créez maintenant deux boucles imbriquées qui vont appeler la méthode *drawDisc* pour dessiner les différents disques dans la fenêtre. Pour l'instant, dessinez-les en rouge uniquement. Le centre du premier disque est en position [10,30], le rayon des disques est de 8 pixels et la distance entre les disques est de 20 pixels selon les deux axes.
5. Dans la version actuelle du code utilisé pour dessiner un disque (méthode *drawDisc*), les deux boucles imbriquées balaient tous les pixels de l'image (*Windows area*). Or, pour un disque, on pourrait ne balayer que les pixels qui sont dans un carré déterminé par +/- le rayon du cercle depuis le centre du cercle (*scanning area*). De ce fait, on réduirait drastiquement le nombre d'opérations nécessaires quand on dessine un petit cercle. La Figure 2 représente la zone à scanner.

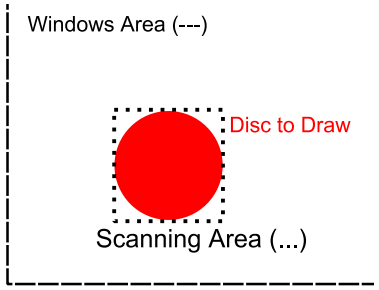


Figure 2 - Scanning Area

Ajoutez une méthode *fastDrawDisc* pour que seuls les pixels faisant partie de l'espace restreint (*scanning area*) dessiné soient balayés. Il faut aussi faire attention au fait que le *scanning area* doit rester à l'intérieur de la fenêtre (*windows area*), car il n'est pas possible d'utiliser la méthode *setPixel* avec des coordonnées qui sont à l'extérieur de la fenêtre graphique !

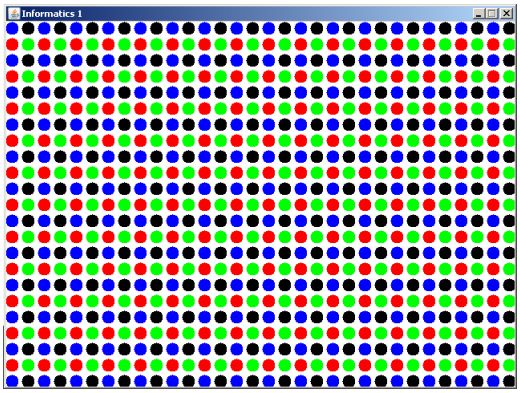


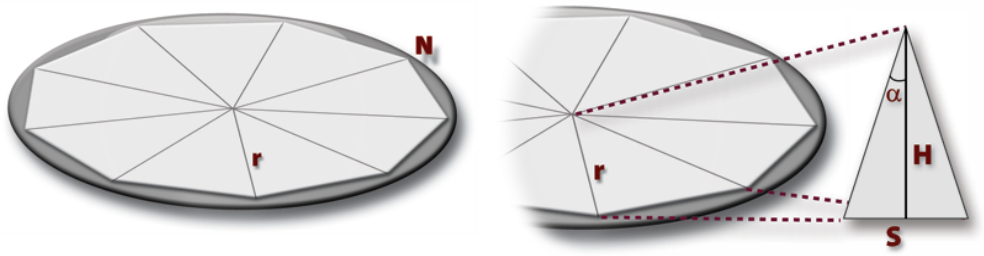
Figure 3 - Dotted Flag, multicolor

[Optionnel] A l'aide de la méthode `System.currentTimeMillis()`, mesurez le temps nécessaire pour dessiner tous les cercles à l'aide des deux méthodes (la rapide et la lente).

6. Rajoutez maintenant les conditions pour que les couleurs des disques forment un motif qui se répète comme dans la Figure 3 (une ligne de cercles noir-bleu, une ligne de cercles rouge-vert). Pour ce faire, ajoutez une fonction utilisant la fonction *drawDisc* définie au point 4 à laquelle vous ajouterez un paramètre *color*.

Partie 3 - N-Polygone

Nous souhaitons construire un programme qui calcule la surface d'un polygone régulier connaissant le nombre d'angles (N) et le rayon (r) du cercle circonscrit



Dans un premier temps, un organigramme est dessiné décrivant les tâches à effectuer (à gauche sur la figure 4). La difficulté de cet algorithme est le calcul de la surface d'un triangle. C'est pourquoi, ce problème est déplacé dans un sous-programme.

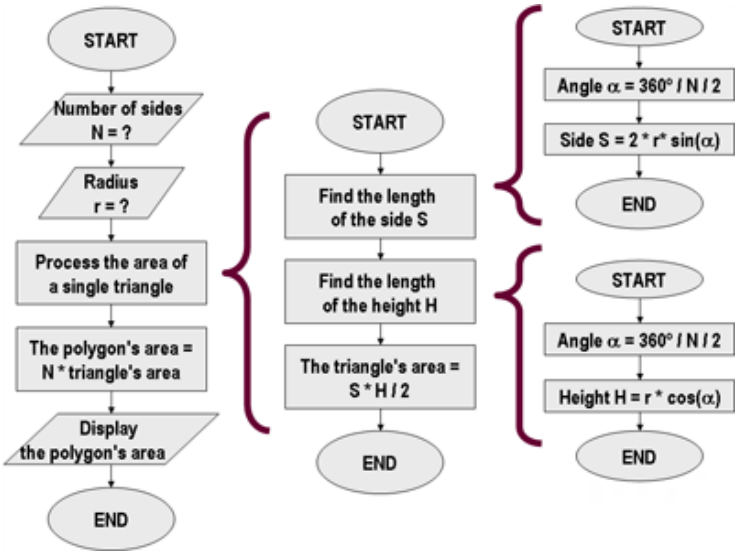


Figure 4. Algorithme du calcul de la surface d'un polygone

Pour ce sous-programme, un nouvel organigramme est dessiné (au milieu). Ici également, l'algorithme dessiné n'est pas entièrement détaillé. A nouveau, une difficulté apparaît, celle de déterminer la hauteur d'un triangle, c'est un problème en lui-même pour lequel un autre sous-programme sera créé. Finalement pour déterminer la hauteur **H** et la base **S** d'un triangle, on utilise les organigrammes de droite. Le calcul du polygone régulier a donc été subdivisé en trois sous-programmes qui peuvent être facilement écrits en Java.

Tâche 3

Dans le projet *Lab4*

1. Créez une classe *Polygon*
2. Ecrivez le programme qui implémente l'organigramme ci-dessus.

Remarque :

Les deux fonctions trigonométriques `cos()` et `sin()` sont disponibles dans la classe `Math` de l'API Java . Considérez également que ces fonctions travaillent avec des radians, c.-à-d. que l'angle en degrés (α) doit d'abord être converti en radians (r) avec la formule :

$$r = \frac{\alpha * \pi}{180}$$

Vous pouvez utiliser pour π la constante `Math.PI`. *Note* : afin de tester le bon fonctionnement de votre programme, vous pouvez vérifier que pour `n = 4` et `r = sqrt(2)*5`, le résultat doit être 100. Pour `n = 12` et `r = sqrt(2)*5`, le résultat doit être 150.