LABORATOIRE 6 | *Informatique 1*
HANGMAN

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

But du laboratoire

1. Le but de ce laboratoire est de mettre en pratique les classes par la création et l'utilisation de différents objets. Pour ce faire, nous allons dans un premier temps développer quelques classes simples puis développer une application qui simule le jeu du pendu.
2. Dans ce jeu, qui se joue à deux, l'un des joueurs entre un mot d'un certain nombre de lettres. L'autre joueur propose des lettres, l'une après l'autre. Pour chaque lettre, le programme détermine si la lettre fait partie du mot ou non. Si elle n'est pas contenue dans le mot, un élément du pendu est dessiné dans la fenêtre graphique. Dans le cas contraire, le programme montre à quelle(s) place(s) elle se trouve dans le mot. Le jeu continue jusqu'à ce que soit le mot est trouvé soit le pendu est complètement dessiné, et le joueur a perdu.
3. La durée estimée pour réaliser ce laboratoire est de quatre périodes.
4. Toutes les données de ce laboratoire se trouvent sur le site web du cours http://inf1.begincoding.net dans le thème *Laboratoires*.

Partie 1 - Classe `Person`

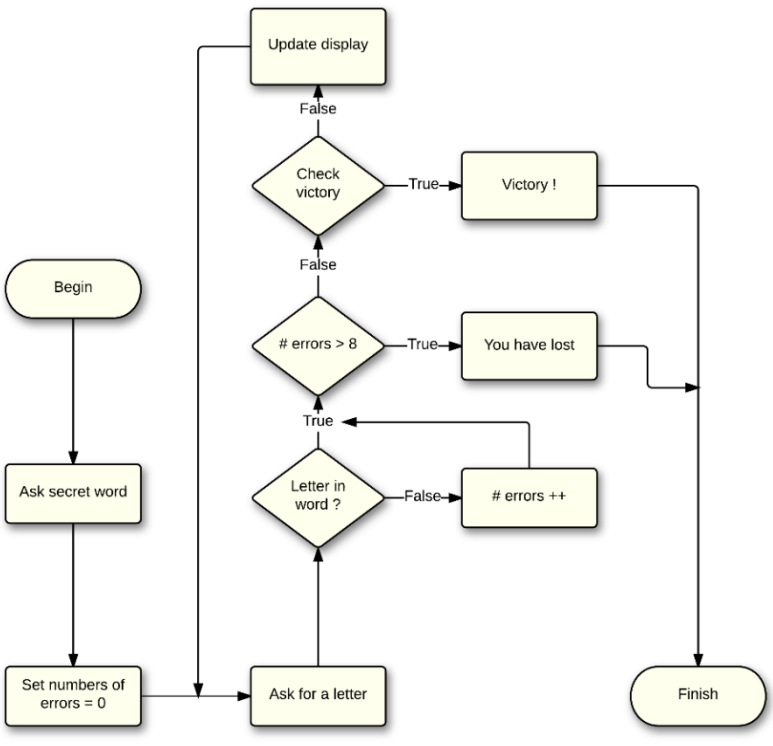
Tâche 1

Dans cette première tâche vous devez simplement développer une classe et tester son fonctionnement. Pour ce faire :

1. Créez une classe `Person` pouvant stocker ces informations sur une personne : nom, prénom, âge, taille.
2. Créez un constructeur pour cette classe afin de pouvoir initialiser tous ces attributs en une fois
3. Ajoutez une méthode `birthday()` qui sera appelée quand une personne a son anniversaire. L'effet de cette méthode est d'incrémenter l'âge de la personne. C'est tout ce qu'elle fait. Le moment où elle est appelée, dans le `main` par exemple, n'est pas important.
4. Ajoutez une méthode `display()` afin de pouvoir afficher une personne sur la console.
5. Testez votre classe avec un exemple.

Partie 2 - Le jeu du pendu en texte

Le jeu de pendu a pour but de faire deviner un mot caché à son adversaire. Ce dernier doit alors, en précisant une lettre après l'autre, trouver le mot caché. Cependant, le nombre de coups maximum est limité à une valeur fixe que l'on nommera `MAX_STEPS` et qui prendra la valeur de 8 dans ce laboratoire. À chaque fois que l'adversaire donnera une lettre qui ne fait pas partie du mot, une variable s'incrémentera et cela aura pour effet de dessiner la prochaine partie du pendu. Si le nombre d'essais ratés est égal à `MAX_STEPS`, le pendu sera totalement dessiné et la partie sera terminée. L'organigramme suivant montre le déroulement du jeu :



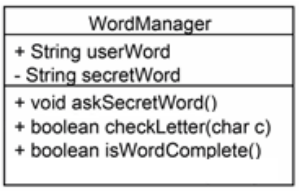
Tâche 2

Pour commencer, vous allez développer le jeu du pendu uniquement en mode console.

Pour y arriver, effectuez les opérations suivantes :

1. Créez un nouveau projet *Eclipse* et ajoutez une nouvelle classe principale nommée `HangMan`.
2. Déclarez la constante entière globale `MAX_STEPS` de valeur 8 qui contient le nombre maximum d'essais. Déclarez aussi une variable d'instance, nommée `current_step`, pour stocker le nombre d'essais infructueux dans la partie.
3. Ajoutez une nouvelle classe à votre programme que vous nommerez `WordManager`. Cette classe va contenir toutes les méthodes et variables nécessaires à réaliser la logique du pendu. Ajoutez ainsi les attributs `secretWord` et `userWord` (les deux des `String`) dans la classe `WordManager`. La variable `secretWord` ne doit pas être accessible depuis l'extérieur de cette classe.
4. Étudiez maintenant la documentation fournie en classe sur `String` afin de prendre connaissance des méthodes fournies par cette classe pour travailler sur des chaînes de caractères. Vous pouvez également regarder la documentation sur l'API en ligne du langage (https://docs.oracle.com/javase/8/docs/api/java/lang/String.html).
5. Ajoutez dans cette classe la méthode `askSecretWord()` ne prenant pas de paramètre et ne fournissant aucune valeur retour. Cette méthode demande à l'utilisateur d'entrer le mot secret à la console. Ce mot secret sera stocké dans la variable `secretWord`. Pour lire un `String` depuis la console, vous pouvez utiliser la classe *Input* qui a déjà été utilisée dans plusieurs labos. *Un conseil* : stockez les mots en minuscule après les avoirs lus afin de faciliter les comparaisons. Pour cela, vous pouvez utiliser la méthode `toLowerCase()` de la classe *String*.
6. Complétez la méthode `askSecretWord()` pour qu'une fois le mot secret enregistré, `userWord` comporte le même nombre de lettres que le mot secret, mais que toutes les lettres soient le caractère `'*'`.
7. Testez le comportement de la méthode `askSecretWord()` en instanciant cette classe depuis le `main` de la classe principale.

Tâche 3

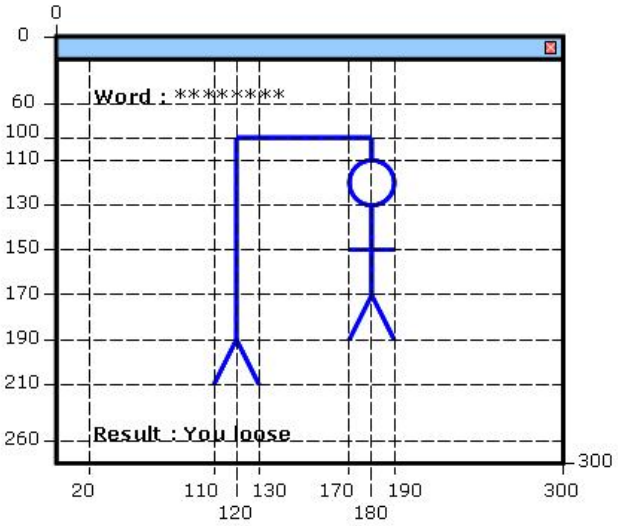


1. Créez dans la classe principale la méthode `play()` où est implémenté le cœur du jeu (poser les questions). Cette méthode est sans paramètre et retourne une valeur booléenne qui sera égale à `false` tant que la partie n'est pas terminée et à `true` lorsqu'elle sera terminée. Une partie est terminée lorsque le nombre d'essais ratés est égal à la valeur `MAX_STEPS` ou que le joueur a trouvé le mot secret.
2. Faites en sorte que la méthode `play()` demande au joueur d'entrer une lettre.
3. Ajoutez dans la classe `WordManager` une nouvelle méthode `boolean checkLetter(char c)`. Cette méthode doit faire en sorte que si la lettre passée en argument se trouve dans le mot secret, il faut modifier le contenu de la variable `userWord` afin de placer la lettre entrée par l'utilisateur à chaque endroit où elle apparaît dans le mot secret. Par exemple, si `secretWord = "hello"` et `userWord = "h****"`, appeler `checkLetter('o')` modifie `userWord` en `"h***o"`.
4. Appelez la méthode `checkLetter` dans `play()` et faites en sorte que si la lettre entrée n'est pas présente dans le mot secret, il faut incrémenter le nombre d'essais ratés. À chaque essai, affichez également le contenu de la variable `userWord` afin de vérifier le comportement correct de votre méthode.
5. Ajoutez une méthode `isWordComplete` dans la classe `WordManager` indiquant par un `boolean` si `userWord` est complet, c'est-à-dire qu'il ne contient plus d'étoiles.
6. Faites en sorte que la méthode `play()` soit appelée dans votre programme tant que la partie n'est pas terminée.
7. Lorsque la partie est terminée, affichez un texte correspondant au résultat du jeu dans la console.

Tâche 4

Modifiez votre programme pour qu'il demande au joueur s'il veut rejouer ou non. Le joueur pourra rejouer autant de fois qu'il répondra `y` ou `Y` à cette question. Pour toute autre réponse, le programme se terminera.

Partie 3 – Pendu, la partie graphique



Maintenant que votre jeu marche en mode console, il vous suffit de créer une interface graphique vous permettant d'afficher votre pendu. Vous pouvez vous inspirer de la façon de faire de ci-dessus mais sentez-vous libre de faire votre propre interface. Pour ce faire, vous aller utiliser la classe `FunGraphics` que nous avons déjà utilisé auparavant lors du Labo 3. Les méthodes suivantes sont à disposition : `drawLine()` pour dessiner des lignes, `drawCircle()` pour dessiner des cercles, `drawString()` pour écrire du texte.

Tâche 5

1. Créez une instance de `FunGraphics` pour faire apparaître une fenêtre vide de 300 x 300 pixels.
2. Créez la méthode `updateGraphicsView()` sans paramètre et sans retour. Cette méthode sera utilisée pour dessiner petit à petit les parties du pendu. Faites en sorte que les différentes parties du pendu se dessinent en fonction des coups ratés de la façon suivante :
3.

Coup raté	Partie à dessiner
1	Pieds du gibet
2	Planche verticale du gibet
3	Planche horizontale du gibet
4	Corde du gibet
5	Tête du pendu
6	Corps du pendu
7	Pieds du pendu
8(MAX_STEPS)	Bras du pendu

   Faites attention d'appeler au(x) bon(s) endroit(s) la méthode `updateGraphicsView()` pour faire fonctionner correctement votre jeu. Gérez aussi le rafraîchissement de votre fenêtre graphique.
4. Testez votre application.

Tâche 6 (pour celles et ceux qui ont déjà programmé ou qui ont envie)

1. La classe `Dialogs` disponible sur la page web de ce labo permet de faire toutes les interactions dans une fenêtre graphique. En conséquence, le mot de passe est alors caché et tout se fait comme dans une application « normale ».

2. Modifiez votre code pour faire les appels adéquats et faire votre jeu graphiquement comme montré en classe.

3. Quelques options supplémentaires possibles : donner un tip sur la fréquence des lettres les plus fréquentes (utiliser le dictionnaire complet), choisir automatiquement un mot à faire deviner dans la liste des 600 avec niveau difficulté (dépendant de la longueur du mot), afficher les lettres déjà jouées…