Séance 1b: L'APPRENTI-PROGRAMMEUR

Université Paris Cité

Objectifs:

- Exécuter un programme.
- Apprendre à lire des programmes.
- Identifier les mécanismes de la programmation impérative.
- Comprendre ce que fait un programme.
- Modifier un programme existant.

Rappel : Afin de bien organiser vos fichiers, chaque TP sera stocké dans son propre sous-répertoire, lui-même situé dans le répertoire `IP1-Java` que vous avez créé au TP précédent. On rappelle ci-dessous les étapes à suivre ; à partir du TP2, il faudra vous débrouiller !

Exercice 1 (Mise en place du TP, ⋆)
En une phrase : récupérez les fichiers du TP sur Moodle, et mettez-les dans un répertoire `TP1b`.

1. *Dans le répertoire* `IP1-Java`*, créez un sous-répertoire* `TP1b`. *Vous pouvez utiliser, au choix, une ligne de commande, ou bien passer par l'interface graphique.*
2. *Rendez vous sur la page Moodle du cours IP1, dans la section "Supports TP", et cliquez sur TP1b.*
3. *Cette fois, le TP contient plusieurs fichiers* `.java` *à télécharger. Au lieu de les télécharger un par un, utilisez sur le bouton "Télécharger le dossier" pour télécharger une archive* `TP1b-20240916.zip`. *Placez-la dans le répertoire* `TP1b`.
4. *Une archive (reconnaissable à l'extension* `.zip`*, mais il en existe d'autres comme* `.tar` *ou* `.tar.gz`*) est un moyen de stocker plusieurs fichiers en un seul. Pour pouvoir les utiliser, il faut d'abord* extraire *les fichiers. Pour cela, tapez dans un terminal :*

```
1  unzip TP1b-20240916.zip
```

   Rappel : *la touche* TAB *permet d'auto-compléter le nom du fichier, pour ne pas avoir à le recopier entièrement à la main.*
5. *Tapez la commande* `ls` *pour vérifier que les fichiers* `.java` *ont bien été extraits de l'archive. Vous pouvez maintenant supprimer le* `.zip`.

□

Avant d'apprendre à écrire des programmes, nous allons commencer par en étudier quelques-uns déjà écrits. Pour chacun des programmes, vous allez successivement : les lire, imaginer ce qu'ils font, les exécuter et enfin, les modifier légèrement. En étudiant ces programmes, vous allez découvrir la plupart des mécanismes de programmation impérative étudiés ce semestre : les affectations, les instructions conditionnelles, les boucles, les fonctions et les procédures. Ces travaux pratiques ne sont qu'une première expérimentation : nous reviendrons en détails sur chacun des mécanismes au fur et à mesure du semestre. Il est donc tout à fait normal que vous n'en compreniez pas encore tous les détails.

Exercice 2 (Hello World !, ⋆)
Le fichier `HelloWorld.java` *contient le code source d'un programme qui est traditionnellement le premier programme que l'on écrit lorsque l'on apprend un nouveau langage de programmation*[1].

1. *Lisez le code source du programme. Que fait-il selon vous ? Vérifiez votre hypothèse en exécutant les commandes suivantes dans un terminal :*

```
javac HelloWorld.java
java HelloWorld
```

2. *Quelles sont les parties du code source qui correspondent à des commentaires ? Quelles sont les parties du code source qui correspondent à du code* JAVA *? De combien d'instructions est composé ce programme ? Quel est le rôle de la procédure* `System.out.println` *?*
3. *Remplacez l'expression « `"Hello World!"` » par « `"Hello" + " " + "World!"` » puis exécutez le programme, est-ce que le comportement du programme a changé ? Que fait l'opérateur + selon vous ?*
4. *Rajoutez les deux lignes suivantes tout au début de la procédure* `main` *:*

```
System.out.println ("What is your name?");
String name = System.console ().readLine ();
```

et modifiez l'expression « `"Hello" + " " + "World!"` » en « `"Hello" + " " + name + "!"` ». Que fait cette nouvelle version du programme selon vous ? Que fait la fonction `readLine ()` *? À quoi sert la variable* `name` *? L'instruction « `String name = System.console ().readLine ()` » affiche-t-elle quelque chose sur le terminal ? Si non, quel est l'effet de cette instruction ? Vérifiez vos hypothèses en exécutant de nouveau le programme. Écrivez des commentaires avant chacune des lignes que nous venons de rajouter pour les paraphraser avec vos mots.*
5. *Modifiez le programme pour qu'il demande le nom et le prénom de l'utilisateur puis affiche :*

```
Hello prenom nom!
```

□

Exercice 3 (Pattern dans le terminal, ⋆)

1. *Ouvrez le fichier* `Ligne.java`*. Exécutez le programme. Que fait-il ? Pouvez-vous le modifier pour qu'il affiche 20 dièses sur une ligne ?*
2. *Dans le fichier* `Ligne.java`*, que se passe-t-il si à la ligne 10, vous remplacez* `System.out.print("#");` *par* `System.out.println("#");` *?*
3. *Ouvrez le fichier* `Affiche.java`*. Exécutez ce programme. Pouvez-vous utiliser une partie de ce programme (en la copiant-collant), pour que dans le fichier* `Ligne.java`*, le programme demande combien de dièses afficher sur une ligne et les affiche.*
4. *Ouvrez le fichier* `Carre.java`*. Exécutez ce programme. Que fait-il ? Que se passe-t-il si vous remplacez la ligne 14 par* `System.out.print("A");` *?*
5. *Annulez la modification précédente dans* `Carre.java`*. Que se passe-t-il si maintenant on change la ligne 18 par* `System.out.print("B");` *? En déduire une façon pour faire afficher au programme un carré plein de dièses plutôt qu'un carré vide.*
6. *Modifiez votre programme pour lui faire afficher un carré plein de dièses de taille 20 par 20 au lieu de 10 par 10. Pouvez vous le modifier pour lui faire faire un rectangle de largeur 30 et de hauteur 15 ?*
7. *Modifiez votre code pour qu'il demande à l'utilisateur de rentrer une largeur et une hauteur, récupère ces deux nombres tapés au clavier et affiche le rectangle correspondant.*

□

1. D'où vient cette tradition ? La réponse ici : http://blog.hackerrank.com/the-history-of-hello-world/

Exercice 4 (L'artiste, ⋆)
Le fichier `Dessine.java` *contient un programme qui dessine le début d'un carré. Pour cela, il utilise une bibliothèque de fonctions de dessin inspirées du langage LOGO*[2]*, un des premiers langages conçus pour apprendre la programmation.*

Important : *Pour cet exercice, notre programme utilise deux autres fichiers* `StdDraw.java` *et* `Turtle.java` *qu'il vous faut aussi télécharger et enregistrez dans votre répertoire de travail.*

1. *Lisez attentivement le code source du programme. Une fois que vous l'avez lu, exécutez-le grâce à la commande :*

```
javac Dessine.java
java Dessine
```

2. *Faites des tests afin de comprendre à quoi correspondent les procédures* `turtle.setheading` *et* `turtle.forward`*.*
3. *Modifiez le code pour finir de dessiner un carré.*
4. *Modifiez le code pour ajouter un triangle au-dessus du carré afin d'obtenir le schéma d'une 'maison'.*
5. *Ajoutez les lignes suivantes à votre programme (avant les instructions gérant le code de fermeture de la fenêtre) :*

```
turtle.up ();
turtle.moveto (150,150);
turtle.down ();
turtle.forward (60);
```

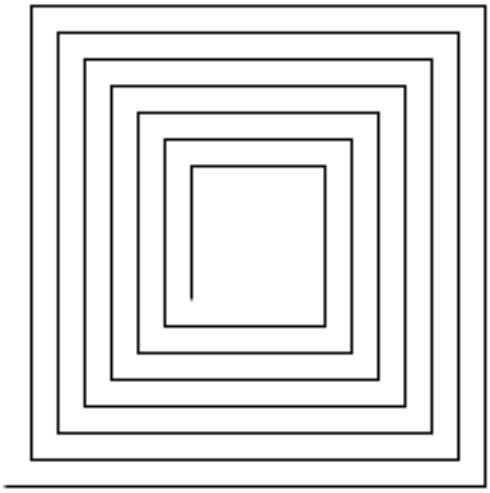
   Qu'observez-vous ?
6. *Utilisez ce que vous avez pu déduire des questions précédentes pour dessiner deux maisons l'une à côté de l'autre.*

□

Exercice 5 (Des dessins à répétition, ⋆⋆)
On s'intéresse dans cet exercice au programme se trouvant dans le fichier `Creneaux.java`*.*

1. *Le programme actuel dessine deux créneaux. Pouvez-vous le modifier pour lui en faire dessiner 4 ? Et si l'on veut lui en faire dessiner 6 ?*
2. *Comment faire, si l'on veut que les créneaux soient plus hauts (avec la même largeur) et pour les faire moins larges ?*
3. *(Bonus) En utilisant une variable* `largeur` *modifiée à chaque tour de la boucle* `for`*, modifiez votre code pour que les créneaux soient de moins en moins larges au fur et à mesure qu'ils sont dessinés.*
4. *À partir de ce que vous avez vu, créez un programme pour faire une spirale carré. Dans un premier temps, on pourra créer cette spirale sans boucle* `for` *mais ensuite il serait judicieux d'avoir un programme utilisant une boucle* `for`*. Voilà un exemple de ce que vous pouvez obtenir.*

2. http://el.media.mit.edu/logo-foundation/what_is_logo/history.html



□