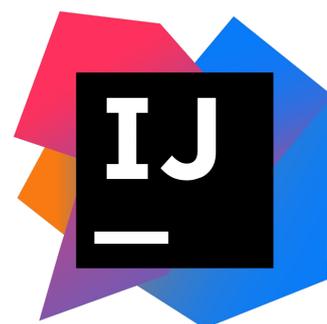
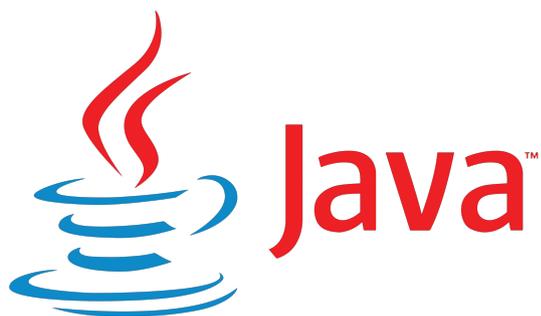


Découverte

Découverte de Java et IntelliJ



Rédigé par

David ROUMANET
Professeur BTS SIO
david.roumanet@ac-grenoble.fr



En cas d'erreur constaté dans ce cours, merci de m'en faire part, afin que je corrige cette ressource libre.

Changement

Date	Révision
2025-01-27	Corrections multiples (install Java, sout, scanner.close(), textes.indexOf(), etc.)
2025-01-31	Ajout du code complet

Sommaire

A Introduction.....	1
A.1 Présentation.....	1
A.2 Prérequis.....	1
A.3 Compétences.....	1
B Application de base.....	2
B.1 Gestion de la console.....	2
B.2 Installation de Java JDK.....	3
B.3 Création du projet avec l'IDE IntelliJ.....	4
B.3.1 Installation.....	4
B.3.2 Création du projet.....	5
B.3.3 Configurer le projet.....	5
B.4 Présentation de l'IDE.....	6
B.4.1 Partie édition.....	6
B.4.2 Partie exécution.....	7
B.4.3 Compilation via l'IDE ou sans IDE.....	8
B.4.3.a Compilation manuelle.....	8
B.4.3.b Compilation par l'IDE.....	8
B.5 Description du code.....	9
B.5.1 La classe Main.....	9
B.5.2 La méthode Main.....	9
B.5.2.a Portée publique.....	9
B.5.2.b Méthode statique.....	9
B.5.2.c Arguments de la méthode main().....	10
B.6 Saisir des caractères dans la console.....	11
B.7 Gérer une liste.....	12
B.7.1 Différence tableau statique et ArrayList.....	12
B.7.2 Utilisation de ArrayList.....	12
B.8 Gérer une liste en pratique.....	13
B.8.1 Création du projet.....	13
B.8.2 Amélioration du projet.....	14
C Annexes.....	15
C.1 Code complet.....	15

Nomenclature :

- **Assimiler** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découvrir** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Voyager** : Projet en autonomie totale. Environnement ouvert : Vous êtes le capitaine !

Les zones en bleu signalent un travail que vous devez réaliser pour vous entraîner.

Vous devez lire les indications qui vous sont données pour réaliser les tâches demandées et progresser dans vos compétences.

A Introduction

Java est le langage ayant popularisé la programmation objet, à grande échelle.

Ce langage est considéré comme exigeant, précis et nécessitant une bonne pratique avant de pouvoir être utilisé avec aisance... ce n'est pas faux !

A.1 Présentation

Cette activité de découverte doit permettre la compréhension des mécanismes de Java en mode console (compilation, exécution) et l'usage d'un IDE (débogage).

Parmi les différents langages orientés objets, Java présente l'avantage d'être disponible sur toutes les plateformes (Microsoft, Linux et Apple), de proposer plusieurs modes d'usage (console et GUI, séquentiel et événementiel) et d'avoir une grande communauté.

Le choix de l'éditeur IntelliJ de JetBrains est le résultat de plusieurs tests entre Eclipse, Netbeans et VSCode : IntelliJ s'avère un peu plus ergonomique, rapide et facile d'utilisation, tout en apportant de nombreuses fonctionnalités.

A.2 Prérequis

Il est préférable d'avoir déjà des notions de programmation classique :

- Variables et portées des variables
- Itérations
- Conditions
- Fonctions

A.3 Compétences

Les compétences du référentiel du BTS SIO abordées ici sont :

B Application de base

Nous allons commencer par une application très simple : nous saisissons une température et l'application affichera celle-ci. Ainsi, nous validerons que le programme est capable d'utiliser notre saisie.

B.1 Gestion de la console

Notre application sera en mode **Console**, car c'est le programme le plus simple à réaliser.



La console, c'est une application qui permet d'afficher et de recevoir des caractères, ce n'est pas graphique, mais textuel. Par abus de langage, les informaticiens parlent de console, dès qu'il y a une fenêtre dans laquelle on peut afficher des informations et saisir des données.

Voici une console de commandes sous Linux Manjaro :

```

david : zsh — Konsole
Nouvel onglet Scinder la vue Copier Coller Chercher
david : zsh x david : zsh x
drwxr-xr-x 2 david david 4096 13 févr. 12:02 Starwars
drwxr-xr-x 3 david david 4096 17 mai 2022 Sync
-rw-r--r-- 1 david david 109199 20 mars 09:12 'Synthèse SIO 2022-2024 Clé
USB.xmind'
drwxr-xr-x 17 david david 36864 14 juil. 17:32 Téléchargements
drwxr-xr-x 2 david david 4096 20 mai 2022 Templates
-rw-r--r-- 1 david david 6850 2 juil. 2022 testdisk.log
-rw-r--r-- 1 david david 532 16 janv. 14:19 textpourri.txt
-rw-r--r-- 1 david david 35 12 mai 2022 todo.txt
-rwxr-xr-x 1 david david 181 24 nov. 2022 unzipAll.sh
drwxr-xr-x 2 david david 4096 17 juin 2022 Videos
drwxr-xr-x 25 david david 12288 8 juin 10:43 Vidéos
drwxr-xr-x 3 david david 4096 27 juin 2022 'VirtualBox VMs'
-rw----- 1 david david 9420794368 27 juin 2022 W10.ova
-rw-r--r-- 1 david david 128 4 oct. 2022 winscp.rnd
drwxr-xr-x 33 david daemon 4096 9 juil. 22:09 WorkSpaces
drwxr-xr-x 8 david david 4096 5 avril 09:58 www
  
```

Notre programme s'exécutera dans ce genre d'environnement. Les opérations se déroulent de manière linéaire, les unes après les autres, contrairement aux applications événementielles, qui réagissent aux événements que l'utilisateur déclenche. Nous créerons une application graphique lorsque nous aurons acquis les bases pour Java.

B.2 Installation de Java JDK

L'IDE IntelliJ peut installer une version de Java, mais ce dernier ne fonctionnera qu'avec IntelliJ.

L'installation globale de Java JDK existe en un paquetage MSI (MicroSoft Installer) sur le site de Microsoft :

Actuellement la page est : <https://learn.microsoft.com/en-us/java/openjdk/download>

Platform	Architecture	Type	Download link	Other files
<i>x64</i>				
Linux	x64	tar.gz	microsoft-jdk-21.0.6-linux-x64.tar.gz	sha256 / sig
macOS	x64	pkg	microsoft-jdk-21.0.6-macos-x64.pkg	sha256
macOS	x64	tar.gz	microsoft-jdk-21.0.6-macos-x64.tar.gz	sha256 / sig
Windows	x64	msi	microsoft-jdk-21.0.6-windows-x64.msi	sha256
Windows	x64	zip	microsoft-jdk-21.0.6-windows-x64.zip	sha256 / sig
<i>AArch64</i>				
Linux	AArch64 / ARM64	tar.gz	microsoft-jdk-21.0.6-linux-aarch64.tar.gz	sha256 / sig
macOS	Apple Silicon	pkg	microsoft-jdk-21.0.6-macos-aarch64.pkg	sha256
macOS	Apple Silicon	tar.gz	microsoft-jdk-21.0.6-macos-aarch64.tar.gz	sha256 / sig
Windows	AArch64 / ARM64	msi	microsoft-jdk-21.0.6-windows-aarch64.msi	sha256
Windows	AArch64 / ARM64	zip	microsoft-jdk-21.0.6-windows-aarch64.zip	sha256 / sig

Choisissez la version x64, msi la plus récente.

Installez la version pour tous les utilisateurs. À la fin de l'installation, déconnectez votre session et reconnectez-vous.

Testez dans une invite de commandes, `java -version`.

B.3 Création du projet avec l'IDE IntelliJ

Vous pouvez utiliser n'importe quel éditeur de texte pour créer un programme Java. Cependant, les éditeurs comme Apache Netbeans, Microsoft Visual Studio Code, Eclipse (IBM) ou JetBrains IntelliJ apportent des améliorations permettant d'avoir tous les outils dans une même interface.



Un IDE (Integrated Development Environment) est donc une application centralisant les outils pour développer dans un langage. Ainsi, cela simplifie le travail du développeur, qui peut se concentrer uniquement sur la création de code.

B.3.1 Installation

Pour ce cours, nous utilisons l'IDE IntelliJ version community de JetBrain, car il est accessible pour tous les OS (Microsoft, Linux ou Mac), il est gratuit et intègre les fonctionnalités graphiques de JavaFX (des bibliothèques graphiques très pratiques).

<https://www.jetbrains.com/products/compare/?product=idea&product=idea-ce>

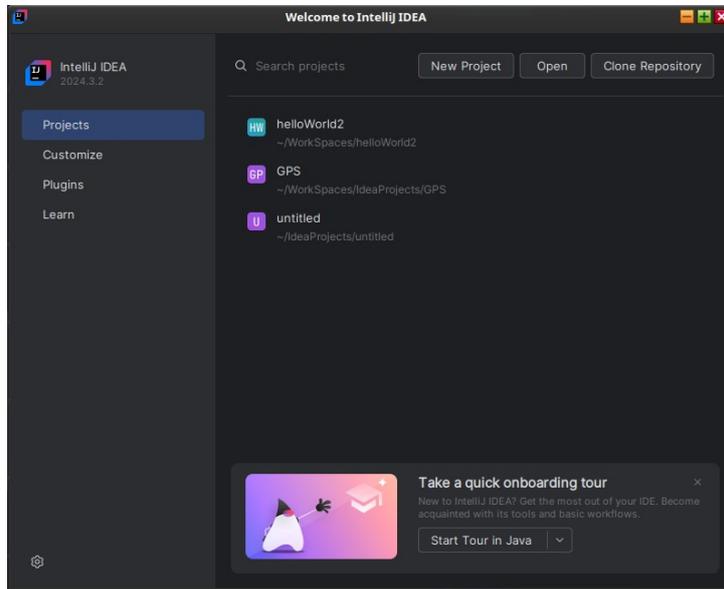
Vous choisissez donc la version Community Edition : <https://www.jetbrains.com/idea/download/>

The screenshot shows the JetBrains website's download page for IntelliJ IDEA. At the top, there are navigation links for 'Developer Tools', 'Team Tools', 'Education', 'Solutions', 'Support', and 'Store'. Below this, the 'IntelliJ IDEA' section is visible, with a 'Download' button. The page lists three operating systems: 'Windows', 'macOS', and 'Linux'. The 'IntelliJ IDEA Ultimate' version is prominently displayed, but a blue arrow points to the 'IntelliJ IDEA Community Edition' section. This section includes the text: 'We're committed to giving back to our wonderful community, which is why IntelliJ IDEA Community Edition is completely free to use'. Below this, the 'IntelliJ IDEA Community Edition' is described as 'The IDE for Java and Kotlin enthusiasts'. A 'Download' button is present, with a dropdown menu showing '.tar.gz (Linux)'. At the bottom of this section, it says 'Free, built on open source'. To the right of the main content, there is a small preview of the IntelliJ IDEA interface and some technical details like 'Version: 2024.2.2', 'Build: 243.23654.117', and '16 January 2025'.

Installer le programme par défaut.

B.3.2 Création du projet

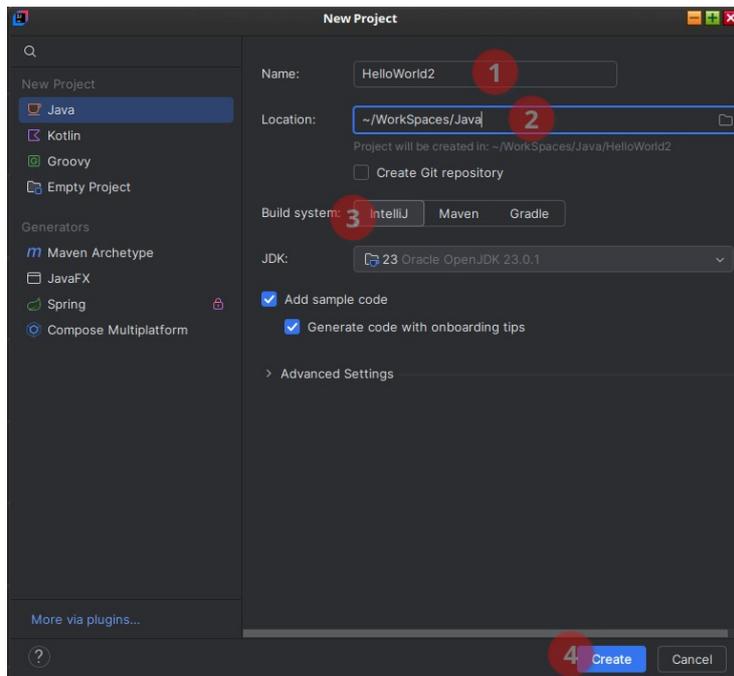
À la première utilisation, l'IDE propose une fenêtre de dialogue simplifiée :



Il suffit alors de cliquer sur le bouton [New Project]. Dans les autres cas, il suffit d'ouvrir le menu de IntelliJ et choisir **File** → **New** → **Project...**

B.3.3 Configurer le projet

Remplissez les champs indiqués ci-dessous, comme décrit dans la page suivante :



Donner un nom à votre projet, choisissez le répertoire (Workspace), et choisissez le type de projet (IntelliJ)

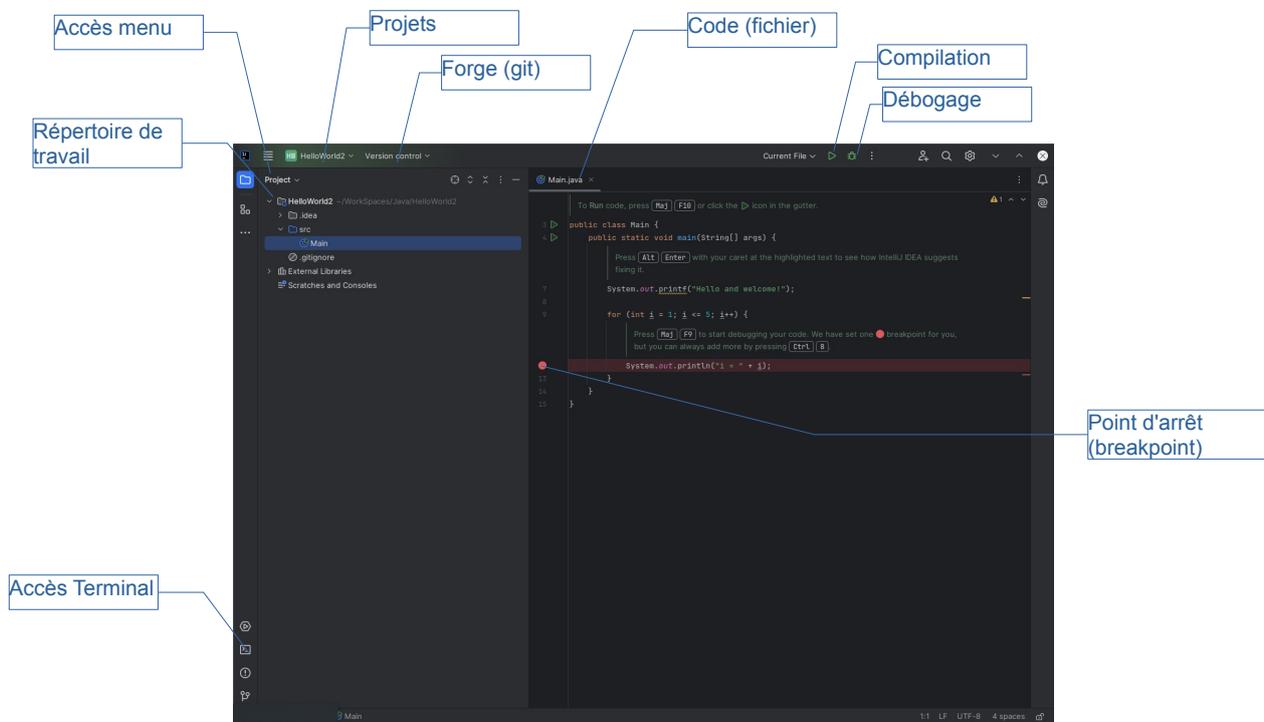
Nom	helloWorld2
Dossier	Workspace/Java
Type	IntelliJ

Cliquez sur le bouton [Create] pour lancer la création de projet.

B.4 Présentation de l'IDE

B.4.1 Partie édition

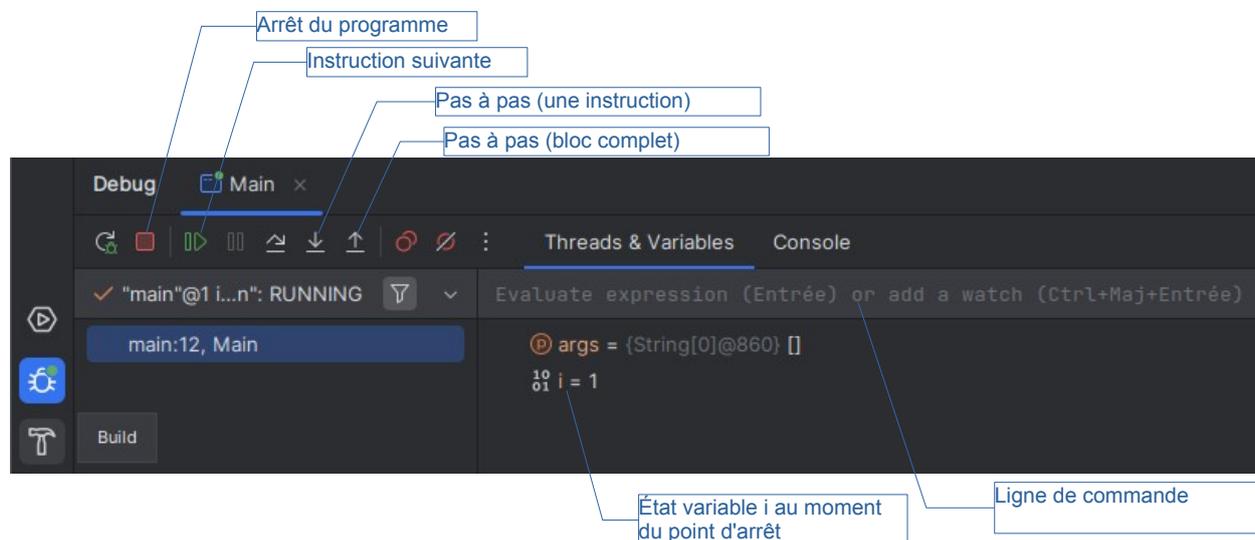
Désormais, le projet s'affiche dans l'IDE, qui comporte plusieurs zones de travail :



L'IDE permet généralement de faciliter la recherche d'erreur, la complétion d'instructions et noms de variables ou fonctions, la coloration syntaxique et ici, l'ajout de points d'arrêts dans le programme.

B.4.2 Partie exécution

La compilation du programme en mode débogage, permet l'affichage d'une console avec des options pour continuer la lecture du programme (après le point d'arrêt), sauter le bloc, etc.



Dans ce mode, l'IDE affiche l'état des variables directement concernées (ici, la variable `i`), mais il est possible d'ajouter d'autres variables avec un clic droit dans le programme.

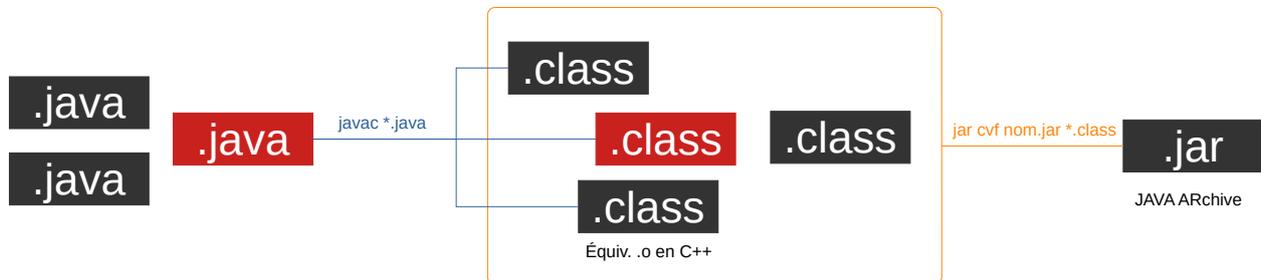
Dans la partie "ligne de commande", essayez d'ajouter la variable `args` (existe dans l'appel de la fonction) en tapant son nom puis en cliquant sur le bouton `[+]` qui apparaît alors au bout de la ligne.

Essayez d'avancer le programme en pas à pas.

L'IDE permet d'espionner votre programme et d'analyser minutieusement votre travail.

B.4.3 Compilation via l'IDE ou sans IDE

Contrairement à JavaScript/HTML, le programme doit d'abord être compilé pour être exécuté par la machine virtuelle Java (ci-dessous, le fichier Main.java est symbolisé en rouge).



B.4.3.a Compilation manuelle

La commande de compilation est `javac nomFichier.java`.

Dans votre terminal, allez dans le répertoire `src` de votre programme et lancez-y une invite de commandes. Il devrait y avoir un seul fichier `Main.java`. Compilez ce code avec la commande `javac Main.java`, puis exécutez le programme avec la commande `java Main` (sans l'extension, car Java cherche automatiquement les fichiers `.class`).

La commande **javac** appelle le compilateur (d'où le c à la fin de la commande) qui va transformer le code en byte code (ou J-Code), une sorte de code Java plus compact et plus rapide à exécuter.



Attention : cette compilation semble fonctionner comme un compilateur pour des langages comme C, C++ ou Delphi/Pascal... mais le résultat de la compilation est un fichier qui est ensuite interprété par la machine virtuelle Java. Ce mode d'exécution "just in time" fait de Java – et C#, son alternative Microsoft – un langage hybride ou semi-compilé. Un code compilé par C/C++ serait, lui, directement interprété par le microprocesseur.

Si vous observez le répertoire contenant votre premier programme en Java, vous devriez avoir le fichier source (`.java`) et le fichier compilé (`.class`) au même endroit dans le répertoire `src` ;



À chaque changement dans le code source, il faut impérativement relancer la compilation avec `javac`, sinon vous exécuterez le programme compilé précédent. Ce n'est pas comme avec JavaScript.

B.4.3.b Compilation par l'IDE

Pour conserver une logique claire, un clic sur l'icône de compilation de l'IDE va créer d'autres répertoires, pour ne pas mélanger vos codes sources avec les fichiers compilés. Dans tous les cas, l'utilisation de la commande **Build Project** (CTRL+9) va créer un répertoire **out** à la racine de votre projet.

Vérifiez ce qu'il contient.

B.5 Description du code

Bien que vous ne sachiez pas encore coder en Java, vous avez un premier code prêt à être utilisé, mais certainement obscur aux yeux de ceux qui ne connaissent pas Java.

Nous allons expliquer ce que le programme fait et décrire le formalisme de Java.

```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or  
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.  
public class Main {  
    public static void main(String[] args) {  
  
        System.out.printf("Hello and welcome!");  
  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("i = " + i);  
        }  
    }  
}
```

B.5.1 La classe Main

Elle s'appelle ainsi, mais il est possible de la renommer autrement. Dans un véritable programme, on trouvera souvent plusieurs classes (une par fichier) et une classe Main pour le programme d'appel.

B.5.2 La méthode Main

Cette méthode est le point d'entrée du programme. Dans un logiciel Java, il doit y avoir au minimum, une méthode `main()` pour que le programme puisse s'exécuter.

B.5.2.a Portée publique

Une portée publique signifie que la méthode est visible. Cela semble logique, mais il faut considérer une classe comme un œuf : tout ce qui n'est pas public, n'est pas visible en dehors de la coquille. Le mot-clé est `public`, par opposition à `private` qui rend la méthode inaccessible depuis une autre classe.

B.5.2.b Méthode statique

Une méthode statique a comme particularité, d'être créée en même temps que le programme : elle a donc une existence **dès le début du programme**, ce qui est nécessaire pour pouvoir exécuter le code. Elle n'a pas besoin d'être "instanciée" (créée, construite, initiée...) pour être accessible. Le mot-clé est `static` : les autres méthodes n'ont tout simplement pas de mot-clé pour décrire qu'il faut les instancier.

B.5.2.c Arguments de la méthode main()

L'écriture `String[] args` signifie que l'attribut **args** sera de type tableau (utilisation des `[` et `]`) de chaînes de caractères (String). L'intérêt réside dans l'appel du programme avec des arguments, en mode console.

Remplacez la ligne `System.out.printf` du programme comme suit, puis relancez le programme.

```
if (args.length > 0) {  
    System.out.printf("Hello and welcome " + args[0] + " !");  
}
```

Utilisez ensuite une invite de commande et lancez la commande `java Main "Chuck Norris"`.

Rédigez un code (utilisant une itération et la commande `println()` à la place de `printf()`) pour afficher tous les arguments transmis au programme. La réponse est visible ci-dessous (sélectionnez le texte dans votre éditeur PDF).

Recompilez votre code (javac) puis testez la commande `java Main "-heros" "Chuck Norris" "Me"`.

Ainsi, l'utilisation d'un tableau en entrée d'une méthode, permet de récupérer des arguments, séparés par des espaces, comme les commandes Linux (exemple : pour la commande `ls` on peut ajouter `-l` pour afficher la liste différemment, `ls -als`, ou la commande `cp` pour copier des fichiers `cp ./* ~/`)

Modifiez l'itération qui affiche les valeurs de `i` pour afficher la table de multiplication par 5, dans le format `i x 5 = x`.

Cette modification est relativement proche de JavaScript. Vous pouvez écrire `sout` puis Appuyer sur [Tab] comme raccourci de `System.out.println()`.

B.6 Saisir des caractères dans la console

Nous allons maintenant ajouter l'interactivité à ce code, en permettant la saisie de 3 valeurs de température pour une ville : minimum, maximum et moyenne.

Il faut d'abord importer la bibliothèque Java qui gère le clavier. Elle se trouve dans un espace de nom **java.util** et s'appelle **Scanner** : il faut le faire en début de programme (au-dessus des classes).

```
import java.util.Scanner;
```

Ensuite, il suffit d'ajouter une variable de type **String**, qui recevra les caractères lus sur le clavier :

```
// à ne déclarer qu'une fois :  
Scanner clavier = new Scanner(System.in);  
  
// Pour lire une chaîne complète  
System.out.print("Entrez une chaîne : ");  
String str = clavier.next();
```

Enfin, à la fin du programme, il faut rendre la mémoire et libérer les ressources réservées, avec l'instruction :

```
clavier.close();
```

Dans notre programme, avant la boucle d'affichage des arguments, ajoutez la lecture d'une chaîne, que vous afficherez avant la liste des arguments.

Réponse (sélectionner tout le champ pour voir) :

Noter que dans ce mode de fonctionnement, la saisie est bloquante, le programme fonctionne de manière linéaire (procédurale).

Fermez le projet helloWorld.

B.7 Gérer une liste

L'idée est ici de créer une liste dynamique : tant que l'utilisateur saisit un mot, on ajoute ce mot dans une liste et on affiche toute la liste.

B.7.1 Différence tableau statique et ArrayList

Contrairement à JavaScript, Java considère un tableau comme ayant une taille finie. Ainsi, une fois déclaré, Java refuse d'étendre un tableau.

- Un tableau est un ensemble de valeurs de même taille, stockées dans une zone contiguë de mémoire. L'accès est donc extrêmement rapide.
- Il est possible de créer un tableau d'une taille fixe supérieure à nos besoins, mais cela prend de la place en mémoire pour rien.
- Le parcours du tableau serait constant et lirait les cellules sans valeurs (`tableau.length` serait constant).

Inversement, les **ArrayList** sont des tableaux dynamiques : ils permettent de palier aux défauts des tableaux classiques. C'est la structure utilisée en Javascript par défaut.

B.7.2 Utilisation de ArrayList

La syntaxe des instructions pour utiliser les **ArrayList** est un peu plus complexe que celle des tableaux, mais permet également plus de simplicité pour ajouter ou supprimer des éléments.

Il faut d'abord importer la librairie **ArrayList** pour pouvoir utiliser son type, puis l'initialiser et enfin gérer son contenu. Voici un ensemble des fonctions utiles pour leur utilisation :

```
// à placer en début de programme
import java.util.ArrayList;

// Initialisation
ArrayList<String> liste = new ArrayList<>();

// Manipulation
liste.add("valeur"); // Équivalent de push
liste.add("Autre"); // Ajoute une deuxième cellule
liste.add("Dernière");

// Trouver l'index de la cellule contenant "Autre"
int index = liste.indexOf("Autre");
// Vérifier l'existence d'une cellule contenant "Autre"
boolean = liste.contains("Autre");
// Lire une cellule
String resultat = liste.get(1) // retourne "Autre"

// Connaître la taille du tableau dynamique
int taille = liste.size();

// Effacer les cellules
liste.remove("valeur"); // supprime la cellule contenant "valeur"
liste.remove(0); // supprime la première cellule
```

B.8 Gérer une liste en pratique

B.8.1 Création du projet

Créez un nouveau projet que vous nommerez GestionListe, dans votre environnement Workspace et toujours en mode IntelliJ.

Copiez et corrigez le programme suivant pour qu'il fonctionne :

```
import java.util.ArrayList;
import java.util.Scanner;

public class SaisieTexte {
    public static void main(String[] args) {
        // Création de l'ArrayList pour stocker les chaînes
        ArrayList<String> textes = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        String saisie;

        System.out.println("Entrez vos textes (ligne vide pour terminer) :");

        // Boucle de saisie
        do {
            System.out.print("> ");
            saisie = scanner.nextLine();

            // Ajout du texte si la ligne n'est pas vide
            if (!saisie.trim().isEmpty()) {
                textes.ajouter(saisie);
            }
        } while (!saisie.trim().isEmpty());

        // Affichage des textes saisis
        System.out.println("\nVous avez saisi " + textes.size() + " textes :");
        for (int i = 0; i < textes.size(); i++) {
            System.out.println((i+1) + ". " + textes.get(j));
        }

        scanner.close();
    }
}
```

Testez votre code.

Lorsqu'il fonctionne, remplacez la boucle d'affichage des textes saisis, par la boucle suivante :

```
// Découverte de la boucle foreach
System.out.println("\nAffichage avec foreach :");
for (String texte : textes) {
    System.out.println("- " + texte);
}
```

Vérifiez qu'il fonctionne également. La boucle **foreach** simplifie la lecture d'une liste et fera partie de vos meilleures méthodes de programmation.

Pour le moment, nous avons programmé en Java de manière linéaire, nous allons ajouter une fonction pour supprimer des éléments dans notre tableau.

B.8.2 Amélioration du projet

Vous allez modifier votre code pour utiliser des fonctions simples (sans paramètre).

Rédaction d'une fonction :

```
private static void saisirTextes() {  
    // Instructions  
}
```

Une fonction est statique et si elle n'a pas d'instruction **return**, comporte le mot-clé **void**.

Au-dessus de la fonction `main()` de votre programme (mais à l'intérieur de la classe `Main`), créez la fonction `saisirTextes()` en copiant les instructions nécessaires dans votre partie `main()` de votre programme. Supprimez les instructions devenues inutiles et pensez à appeler cette fonction à la place des instructions qui étaient dans `main()`.

Faites de même pour l'affichage des valeurs (boucle)

Essayez enfin de créer un menu permettant de faire la saisie ou l'affichage, basé sur ce bout de code :

```
private static int afficherMenu() {  
    int choix;  
    System.out.println("\n=== MENU ===");  
    System.out.println("1. Saisir des textes");  
    System.out.println("2. Afficher les textes");  
    System.out.println("3. Quitter");  
    System.out.print("Votre choix : ");  
    choix = Integer.parseInt(scanner.nextLine());  
    return choix;  
}
```

Notez que la récupération du nombre à partir de la saisie au clavier nécessite une conversion en entier avec l'instruction `Integer.parseInt()`.

C Annexes

C.1 Code complet

```
import java.util.Scanner;
import java.util.ArrayList;

public class GestionListe {
    // Création de la liste et du scanner (variables globales)
    public static ArrayList<String> mots = new ArrayList<>();
    public static Scanner clavier = new Scanner(System.in);

    // Fonction de saisie des mots/textes
    public static void SaisirTexte() {
        String mot = "";
        // Message d'accueil
        System.out.println("=== Saisissez des mots (pour quitter ne rien saisir) ===");

        do {
            // Affichage d'un prompt intelligent
            System.out.print("Mot " + (mots.size() + 1) + " : ");
            mot = clavier.nextLine();
            mots.add(mot); // Ajout à la liste
        } while (!mot.isEmpty());
    }

    // Fonction d'affichage du menu et récupération du choix
    private static int AfficherMenu() {
        int choix;
        System.out.println("\n=== MENU ===");
        System.out.println("1. Saisir des textes");
        System.out.println("2. Afficher les textes");
        System.out.println("3. Quitter");
        System.out.print("Votre choix : ");
        choix = Integer.parseInt(clavier.nextLine());
        return choix;
    }

    // Fonction d'affichage avec découverte de la boucle foreach
    public static void AfficherTexte() {
        System.out.println("Affichage avec foreach :");
        for (String unMot : mots) {
            System.out.println("- " + unMot);
        }
    }

    // ===== Fonction d'appel main() obligatoire =====
    public static void main(String[] args) {
        int monChoix;
        System.out.println("Gestion Liste");
        do {
            monChoix = AfficherMenu();
            if (monChoix == 1) SaisirTexte(); // format simplifié car une seule commande
            if (monChoix == 2) AfficherTexte();

        } while (monChoix < 3);
    }
}
```