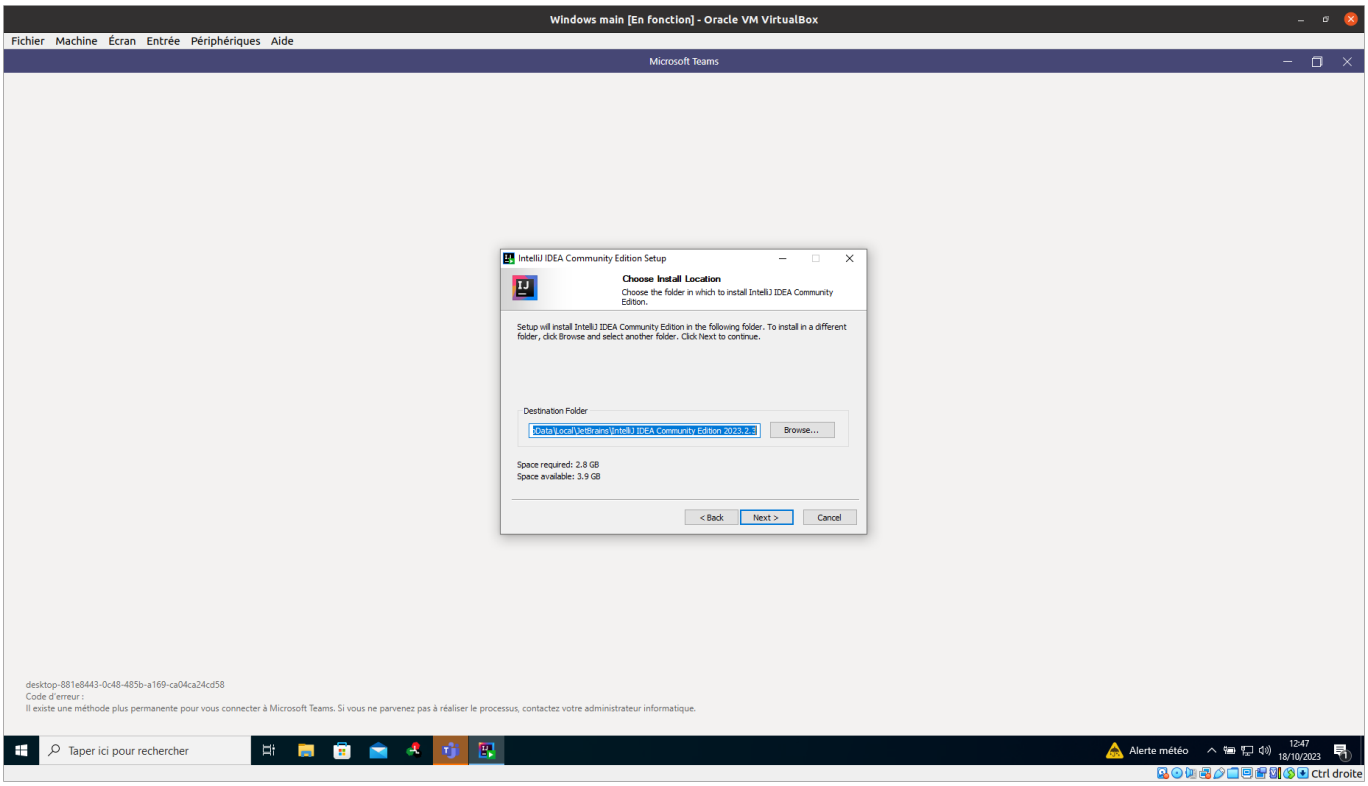
Présentation de IntelliJ

Installation :

Rendez-vous ici pour télécharger l'IDE. Choisissez la version apropriée pour votre OS.

Windows vous avertira peut-être que le fichier risque d'endommager l' ordinateur. Ignorez-le.

Choisissez le dossier dans lequel vous souhaitez installer l'application, comme montré ci-dessous :

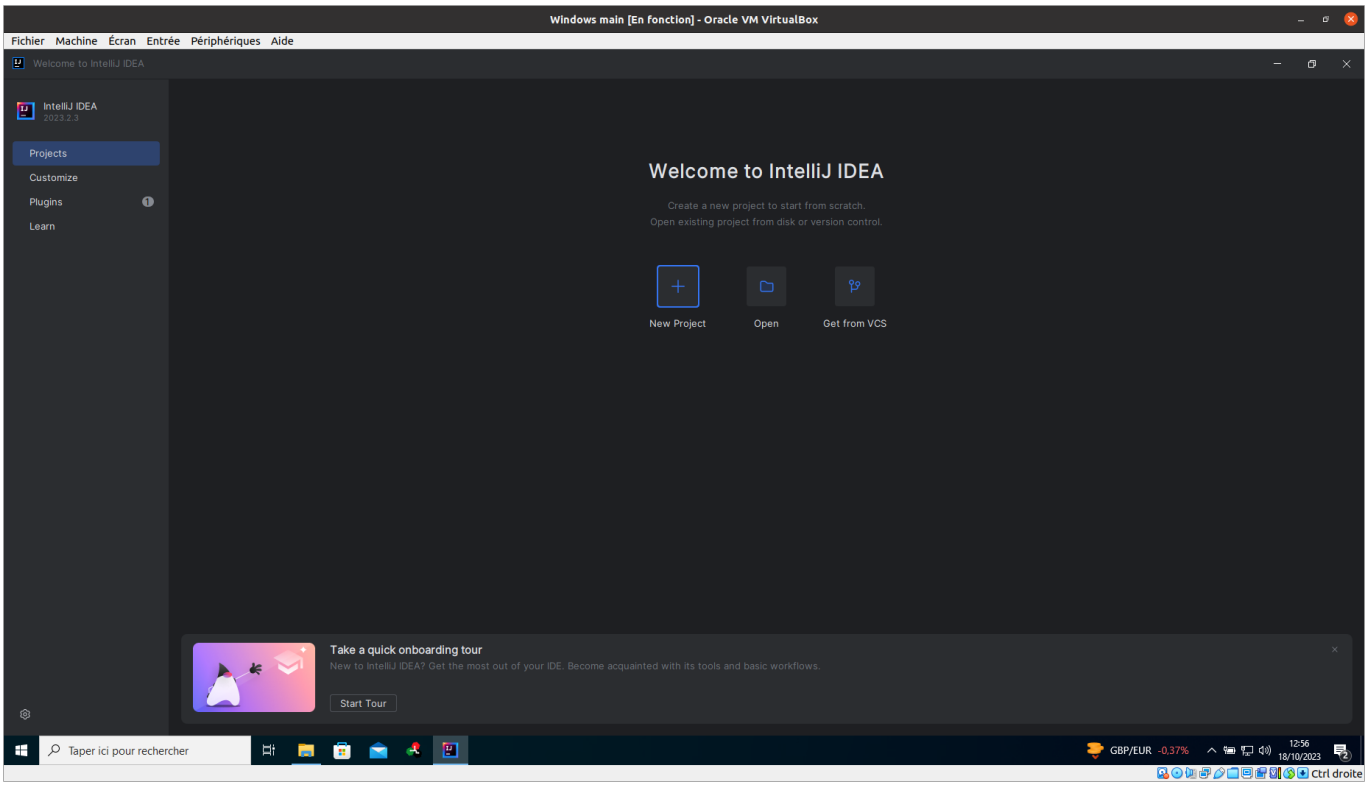


Ensuite, choisissez juste d'ajouter un raccourci sur votre bureau, et continuez avec l'installation.

Enfin, cochez la case "run intelliJ" pour lancer l'IDE.

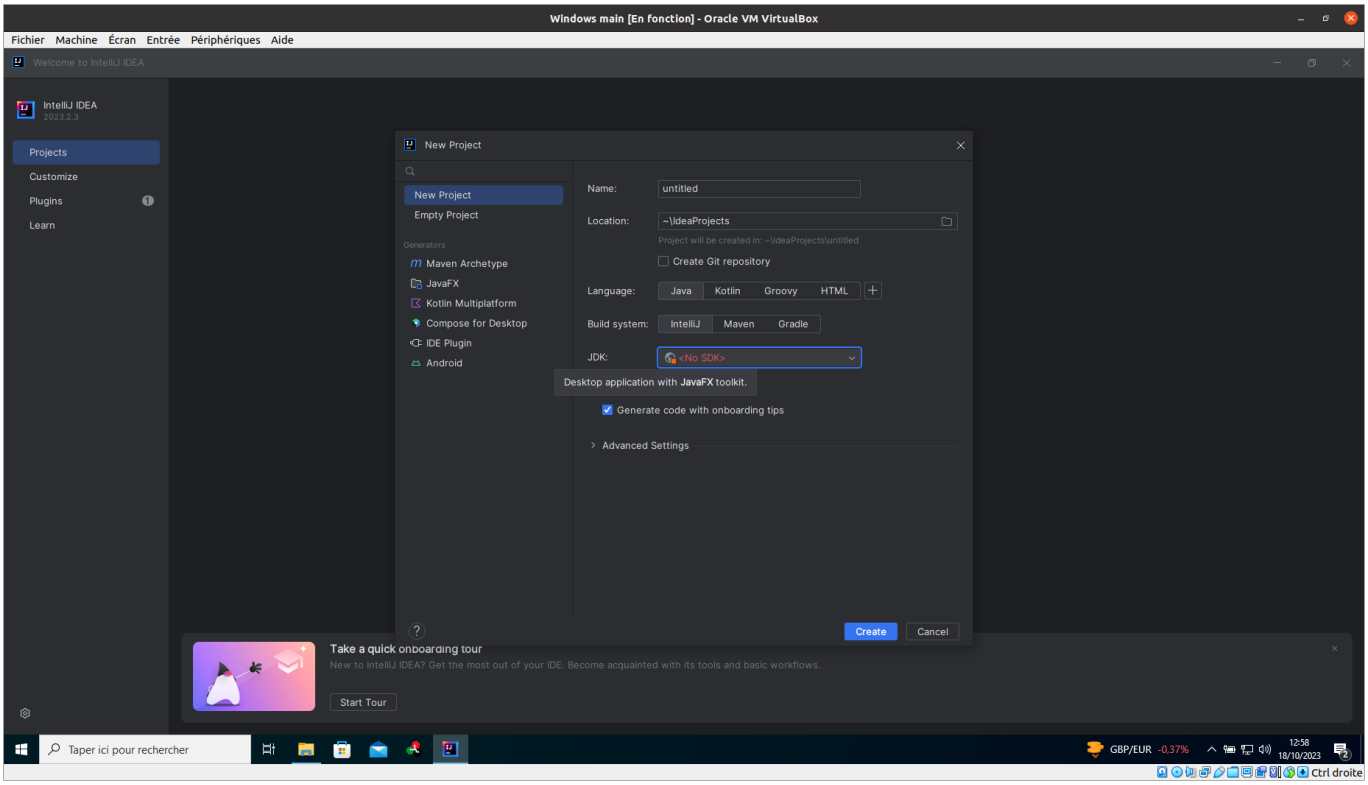
Créer un projet :

Vous devriez arriver face à cette interface :



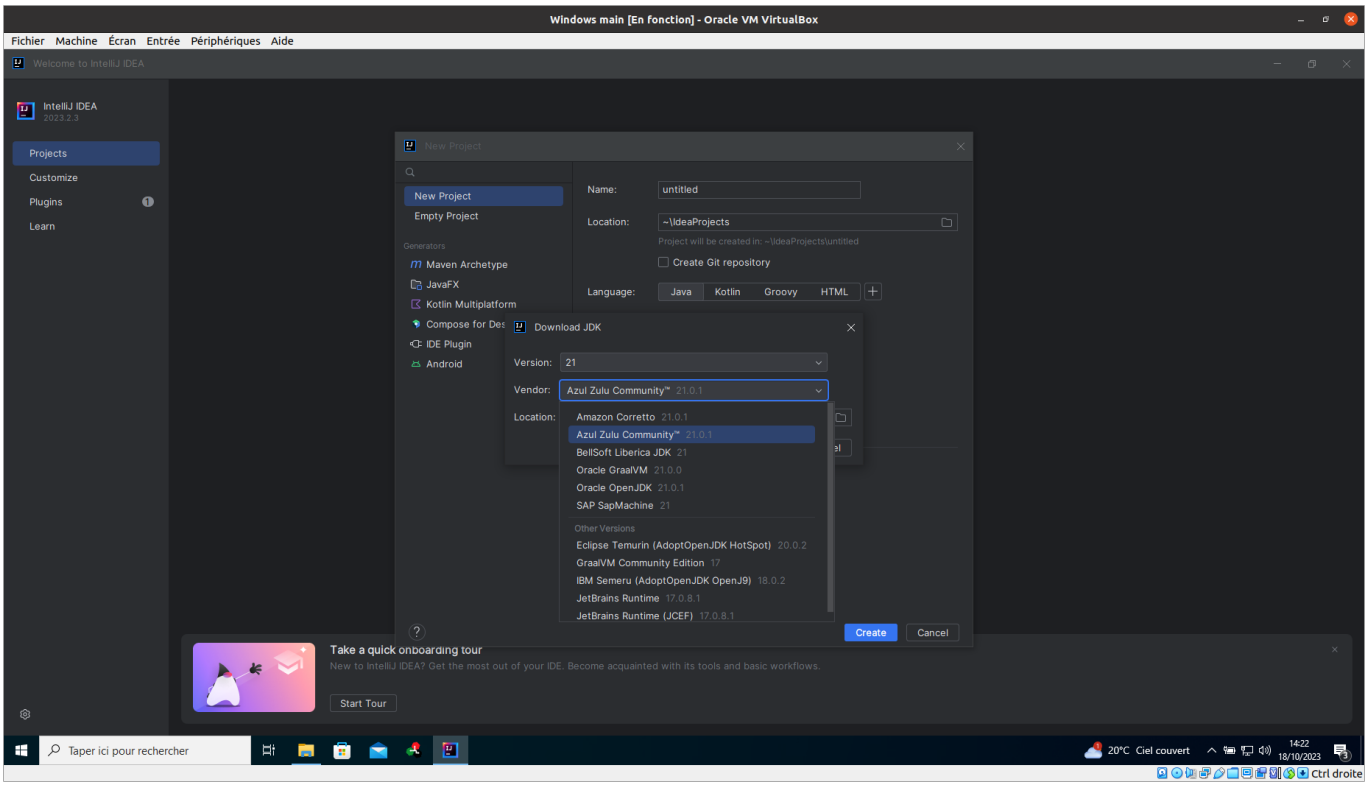
Sélectionnez "new Project".

Vous devriez vous retrouver face à cet écran



Si vous n'avez pas déjà installé Java sur votre ordinateur, vous remarquerez que la partie "JDK" est valorisée à "NO SDK". Cela veut dire que vous allez devoir en télécharger un pour faire tourner un programme Java sur votre machine.

Vous pouvez choisir la version que vous voulez dans cette interface :

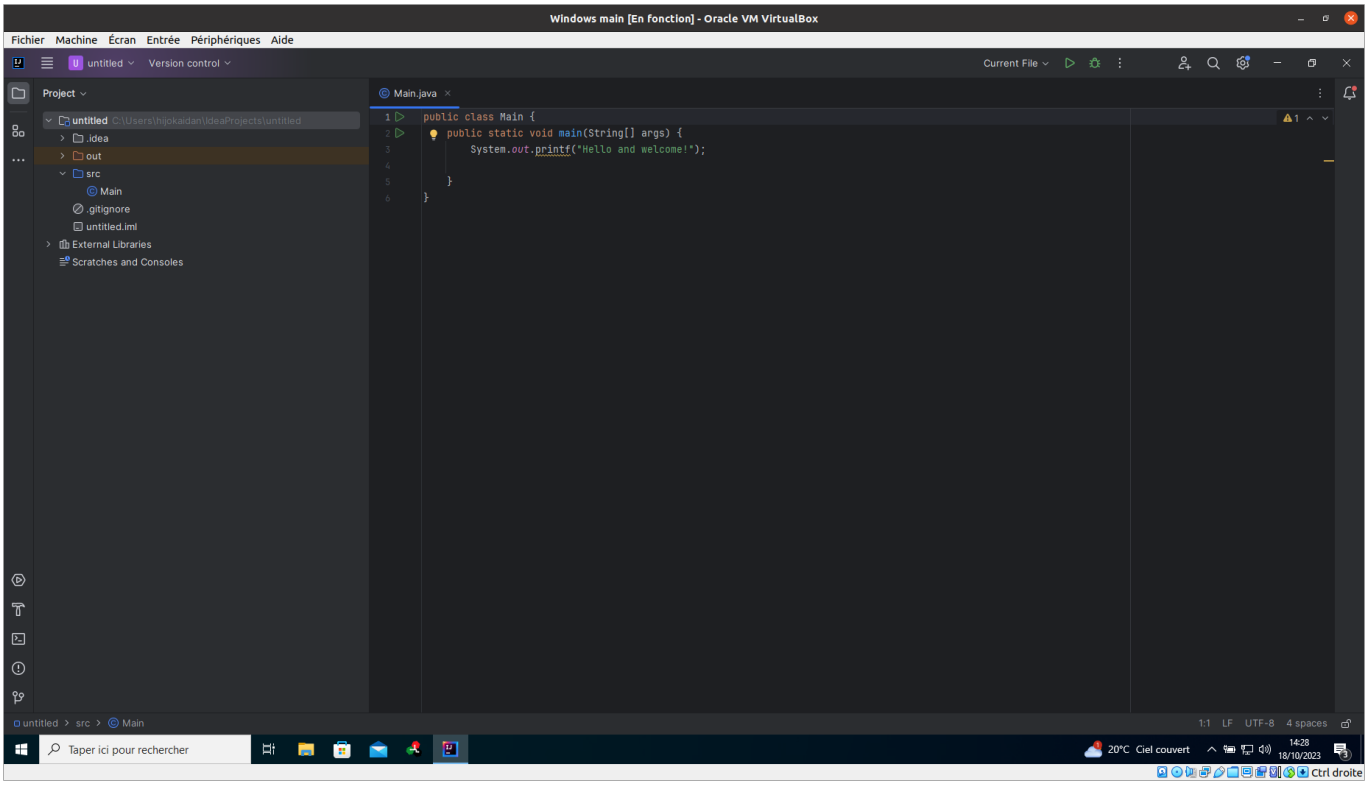


Ce que vous choisissez importe peu pour l'instant. Il existe des différences de performance sur certains points en fonction de l'éditeur que vous choisissez, mais elles ne sont pas très importantes. Cependant, si vous avez des avis tranchées sur l'Open Source ou certaines entreprises, la variété des JDK proposée est là pour s'accorder avec vos principes.

Pour le reste, nous nous contenterons pour l'instant des choix par défaut proposés par IntelliJ

Vous allez ensuite vous retrouvez sur l'interface principale de IntelliJ, avec déjà un fichier de code java présent dans le projet.

Vous pouvez lire les instructions, puis supprimer le code pour n'avoir que la partie qui nous intéresse, comme montré ci-dessous:



Faites ensuite un clique droit sur le dossier racine, et choisissez l'option "Build module". IntelliJ va analyser le JDK pour pouvoir ensuite permettre l'utilisation d'un bouton créant automatiquement un processus de compilation et d'exécution de votre code.

Une fois l'indexation terminée, vous pouvez cliquer sur ce bouton pour exécuter votre programme.

Et voilà, vous venez d'exécuter votre premier programme Java!

IntelliJ : fonctionalités de base

Créer un module

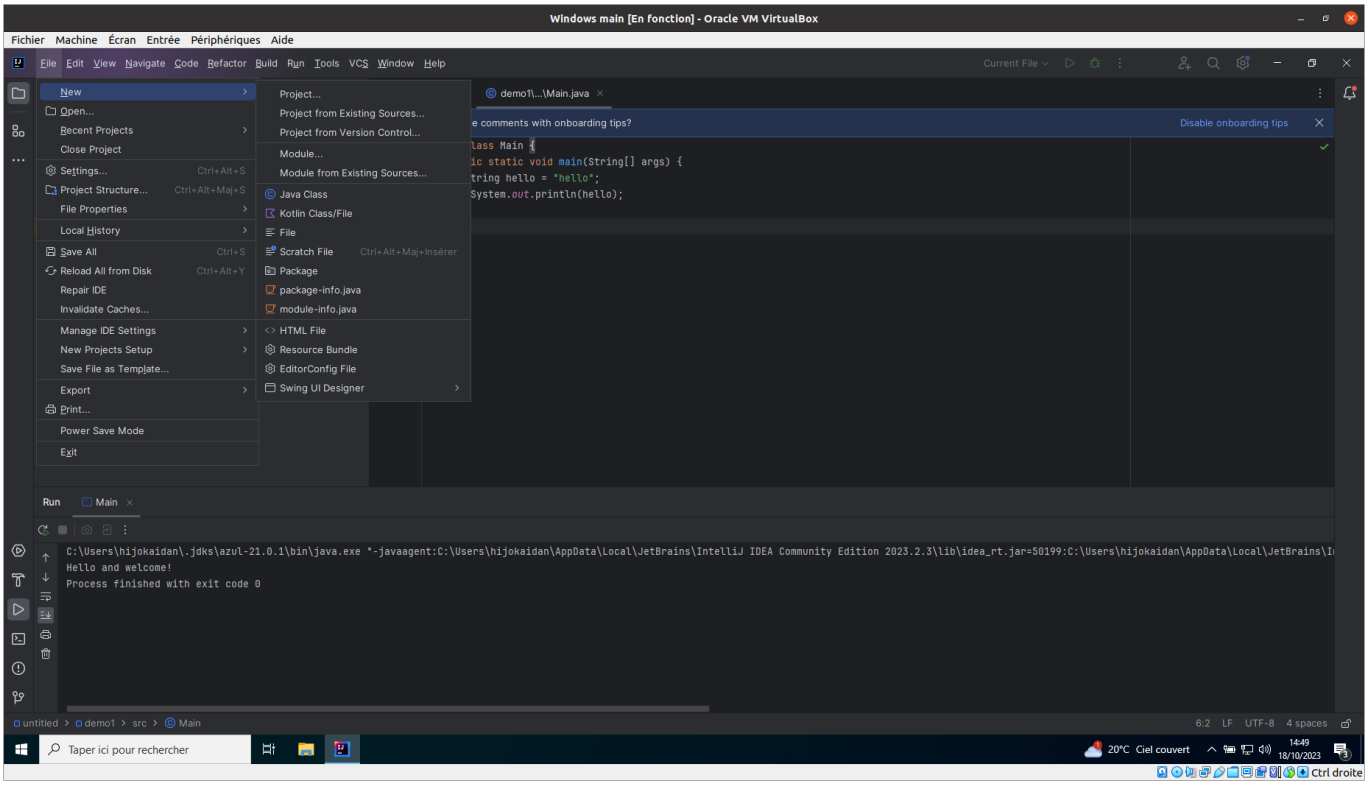
Pour exécuter du code Java avec la configuration que nous avons définies, il faut que ce code soit dans un module. Vous pouvez créer un module en faisant un clique droit sur l'endroit où vous souhaitez le créer, et en choisissant `new -> module`.

Dans l'interface suivante, choisissez le nom que vous voulez donner à ce module, et vous pourrez commencer à exécuter du code à l'intérieur.

Ainsi, vous pourrez avoir plusieurs modules différents dans un même dossier.

Créer un projet

Bon, il est bien gentil notre projet "untitled", mais il va falloir apprendre à en créer avec des noms plus explicites. En haut de l'interface, cliquez sur le petit menu en hamburger, ensuite faites `file -> new -> project`.



Ensuite, gardez les options par défaut proposées par IntelliJ, mais changez le nom du projet par "demoJava". Dans la partie "Location", choisissez un dossier dans lequel vous rangerez tous vos projets pour cette formation.

Vous pouvez également décocher la case qui vous crée le fichier principal.

Pour créer une classe java, faites un clique droit sur le dossier source, et sélectionner `marque directory as -> source root`

Vous pouvez également créer un nouveau module et supprimer le dossier src/ présent hors du module (gardez celui du module).

Ensuite, cliquez-droit sur le dossier src/ que vous venez de marqué comme source root ou sur celui du nouveau module, et choisissez `new -> Java Class`.

Donnez lui le nom que vous voulez, mais commencez par une majuscule.

Copiez-collé le code suivant à l'intérieur de la classe :

```
    public static void main(String[] args) {
        System.out.println("hello mom");
    }
```

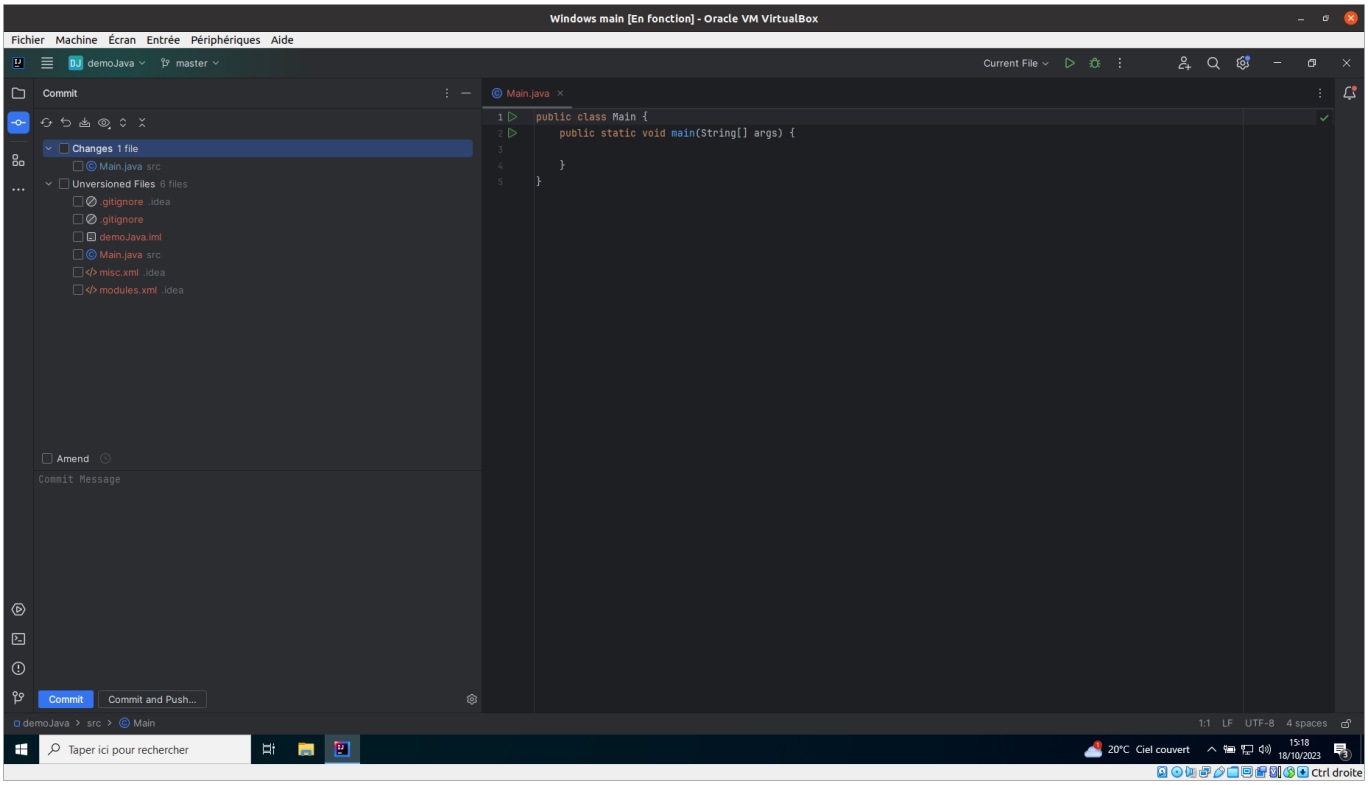
Et voilà, vous êtes maintenant capable de créer un nouveau projet.

Utiliser Git

En haut de votre interface se trouve un bouton sur lequel est marqué "Version control"

Appuyez dessus, et sélectionnez l'option "create git repository"

Vous devriez voir sur la barre de gauche s'ajouter une icone ressemblant à cela : "-o-". Il s'agit d'un onglet permettant de gérer git via une interface graphique.

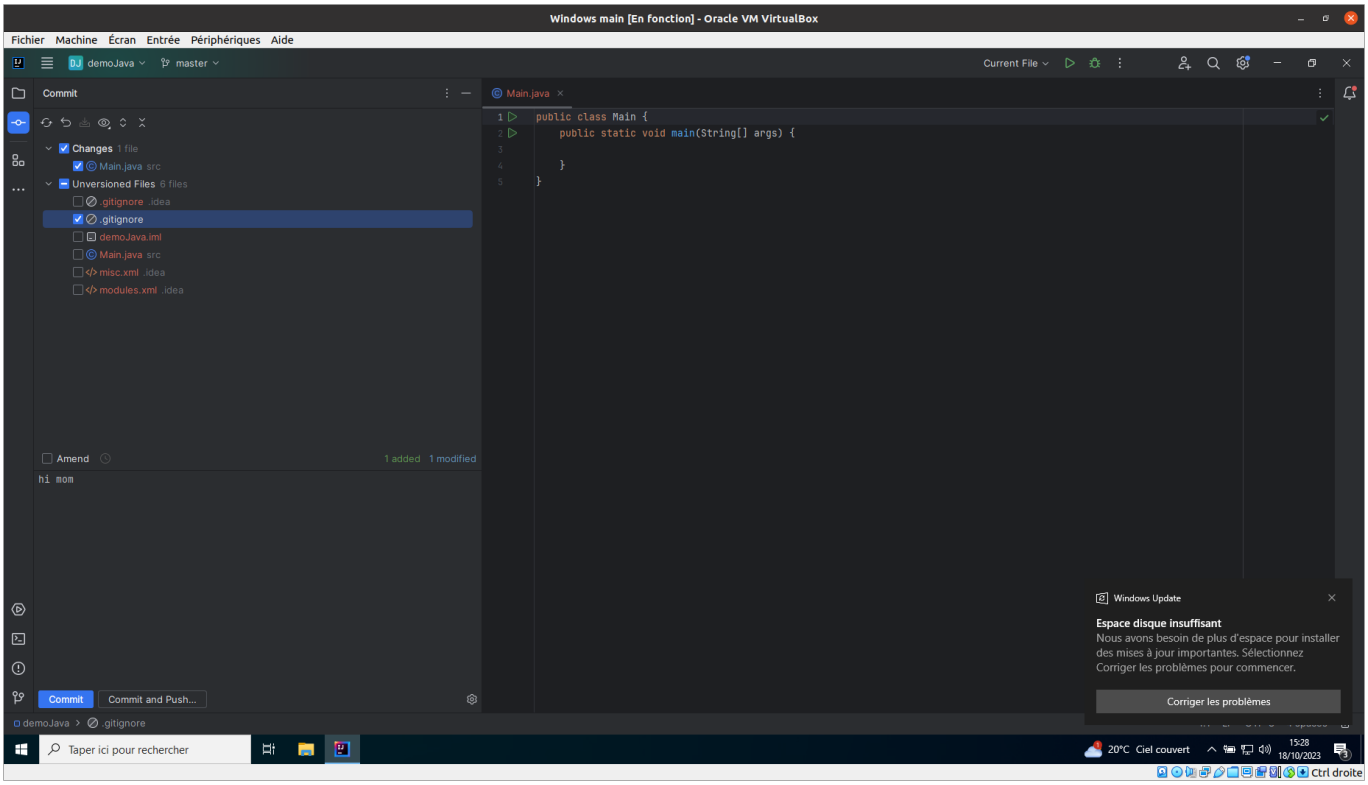


Sur cette interface, vous pouvez choisir les fichiers à commit. Les fichiers déjà suivis par git sont indiqués dans `Changes`, et les fichiers non suivis sont indiqués dans `Unversionned Files`.

Sélectionnez parmis ceux-ci les fichiers qui vous intéressent, donc évitez d'ajouter les fichiers dans le dossier .idea.

```
Astuce : si vous ne voulez plus voir ces fichiers dans vos commits,
ajoutez-les au .gitignore
```

Écrivez ensuite votre message de commit dans la fenêtre où se trouve `Commit Message`, et appuyez sur le bouton commit.

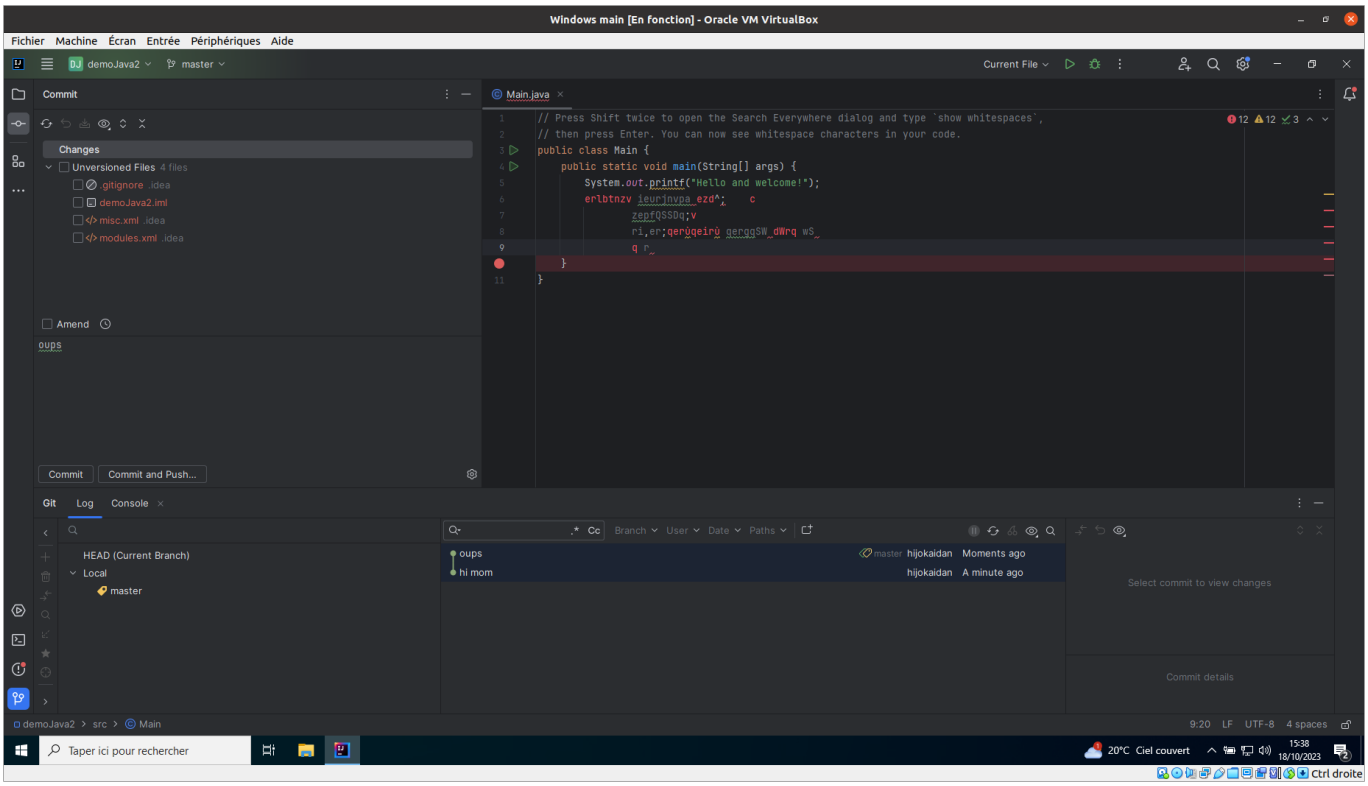


Et voilà, vous avez fait votre premier commit!

Maintenant, allez sur votre fichier java, et roulez votre tête sur le clavier. Faites un deuxième commit.

A moins d'un miracle, IntelliJ va vous afficher un très grand nombre d'erreurs et vous ne devriez plus pouvoir lancer l'application. Mais pas de panique, nous allons revenir au commit précédent

Pour cela, cliquez sur l'icône en forme de branche tout en bas à gauche de l'interface. Cela ouvrira une fenêtre avec l'historique des commits. Faites un clique droit sur votre premier commit, et choisissez l'option `reset current branch to here`, et choisissez l'option `hard` dans la fenêtre suivante.



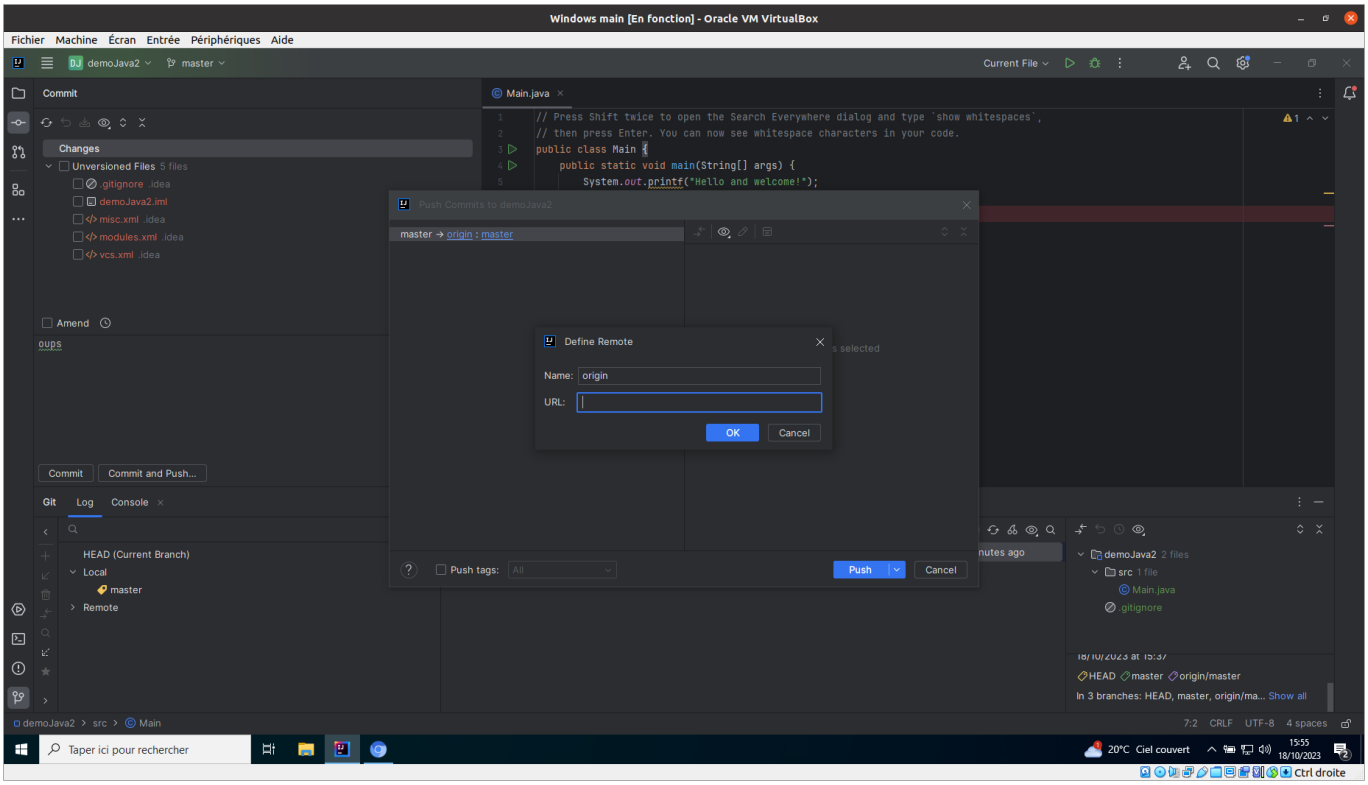
Vous devriez donc être revenu au commit précédent, et pouvez recommencer à travailler sur un document propre.

Vous pouvez donc maintenant sans crainte rouler votre tête sur le clavier!

Maintenant, il peut être intéressant de rajouter un dépôt distant, par exemple l'un de vos dépôt github.

Pour cela, créez un dépôt sur github avec juste un README.

Ensuite, sur votre IDE, cliquez sur le bouton du haut avec une icone de branche. Choisissez l'option `push`. Sur la fenêtre qui va s'afficher, cliquez sur `Define remote`. Copiez-coller l'URL de votre dépôt GitHub.



Maintenant, chaque fois que vous effectuerez un push, votre code sera envoyé sur le serveur de GitHub.

Et voilà, vous devriez maintenant être capable d'utiliser IntelliJ!

Bien évidemment, il est possible de faire beaucoup d'autres choses avec cette IDE. Il est possible de rajouter des plugins, on peut utiliser les fonctionalités de git pour créer de nouvelles branches, en changer, faire des merges...

A vous de voir comment vous souhaitez l'utiliser, et à quel point vous souhaitez personnaliser votre expérience.

Annexe 1 : le JDK

Le **JDK** (Java developpement kit) est l'ensemble des outils qui permettent d'écrire et exécuter du code Java sur une machine donnée.

Java se veut un langage capable de s'exécuter de la même façon sur toutes les machines quelque soient leur OS (Windows, Debian, MacOS, etc).

Pour cela, Java n'est pas directement compilé en langage machine, mais compilé en **bytecode**. C'est ce bytecode qui sera transformé en langage machine par un outil du JDK que l'on appel la **JVM** (pour Java Virtual Machine). Ainsi, le langage Java est compilé en bytecode pour la JVM, et la JVM compile ensuite ce bytecode en langage machine.

Le JDK est donc composé des éléments suivants:

- Un compilateur
- Un debugger
- La JVM

Il existe plusieurs versions du JDK, en fonction de son éditeur et de la version de Java que vous souhaitez utiliser.

Annexe 2 : les IDE

IntelliJ est ce que l'on appel communément un **IDE** (Integrated Developpement Environnement). Concrètement, pour écrire du code java, vous n'avez besoin que d'un éditeur de texte et du JDK.

Cependant, entre devoir installer le JDK sur votre PC, écrire du code Java dans notepad, et utiliser les lignes de commande du terminal pour lancer votre application, et utiliser les outils d'un IDE, il y a **une différence de complexité très importante**.

Un IDE est là pour **cacher cette complexité**, et vous permettre de coder sans avoir à vous soucier de ces détails techniques. Elle vous offre également des **outils de gestion de projet** comme une interface pour git, possède son propre **éditeur de texte**, et possède soit un **analyseur de code** (comme c'est le cas pour IntelliJ) où une connexion à un **serveur de langage** utilisant le protocole **LSP** pour vous fournir de **l'autocomplétion et des actions sur le code** comme de la refacto ou de la création de méthodes, de classes, etc.

Il existe d'autres solutions que IntelliJ pour cela, comme NetBeans, Eclipse, ou même NeoVim, et si vous souhaitez utiliser votre propre IDE ou éditeur de texte, faites-vous plaisir. Cette présentation n'est ici que pour vous proposer une solution simple et rapide pour commencer, mais si vous avez vos préférences, ce n'est pas un problème.

Tant que vous pouvez lancer un projet Java, la solution que vous utilisez importe peu.