Labo 1 – First steps

101.1 Programmation impérative

Objectifs et donnée du laboratoire

Les objectifs de ce laboratoire sont :

1. se familiariser avec les outils utilisés dans le cours *101.1 Programmation impérative*;
2. faire vos premiers pas en *Scala* (si ce n'est pas déjà fait);

La durée de laboratoire est de **2 périodes**. Vous aurez l'occasion durant ce premier laboratoire de créer votre premier projet *Scala* et de faire connaissance avec l'environnement de développement *IntelliJ*.

Partie 1 – Les outils et la documentation

Plateforme ISC Learn

Tous les cours en ISC utilisent la même plateforme pour assurer l'échange d'informations entre étudiant·e·s et professeur·e·s. Celle-ci est disponible sur le lien https://isc.hevs.ch/learn et vous devriez avoir reçu vos informations de login par email normalement.

Sur le cours *101.1 Programmation impérative*, vous trouverez des informations sur le cours, des anciens examens, les dernières nouvelles… Cette plate-forme est accessible tant à l'intérieur qu'à l'extérieur de la HES-SO Valais mais avec votre login.

Tâche 1 : Exploration du site web

1. Trouvez les slides du cours du jours ainsi que la donnée de ce laboratoire au format informatique.
2. Téléchargez les fichiers `RoomCalc.scala` et `Input.scala` et retrouvez ces fichiers sur votre disque dur.

Tâche 2 : Installation des outils

Pour ce cours, nous allons utiliser le langage Scala avec comme éditeur de code *IntelliJ*. Ces deux outils sont disponibles gratuitement en téléchargement pour les principaux systèmes d'exploitation Windows, MacOS ou Linux.

Voici comment procéder à l'installation :

1. Téléchargez et installez IntelliJ Community Edition en suivant le lien https://www.jetbrains.com/idea/download. Durant l'installation, vous pouvez choisir quels plugins installer. Choisissez le **plugin Scala**.
2. Si vous avez loupé l'installation du plugin ou vous avez déjà une version d'IntelliJ fonctionnelle, vous pouvez télécharger et installer le plugin Scala en suivant les instructions suivantes (cherchez *"Scala"* dans le menu des plugins)

Partie 2 – Premiers pas

Maintenant que les outils sont installés, vous allez les utiliser pour écrire votre premier code.

Tâche 3 : Créer votre premier projet

1. Lancez *IntelliJ* et cliquez sur `File -> New -> Project`
2. Donnez lui un nom, comme par exemple `First project`

3. Choisissez où vous souhaitez mettre ce projet.
4. Dans le langage, choisissez **Scala** et comme `Build system` mettez `IntelliJ`.

Vous devez ensuite choisir le JDK, c'est-à-dire la machine virtuelle Java. Cliquez sur la flèche, puis `Download`. Nous allons toujours utiliser une version `17` du JDK et comme machine la version `GraalVM`. Pour le Scala SDK (c'est-à-dire la version de Scala) choisissez **impérativement** une version 2.13 (e.g. 2.13.XX).

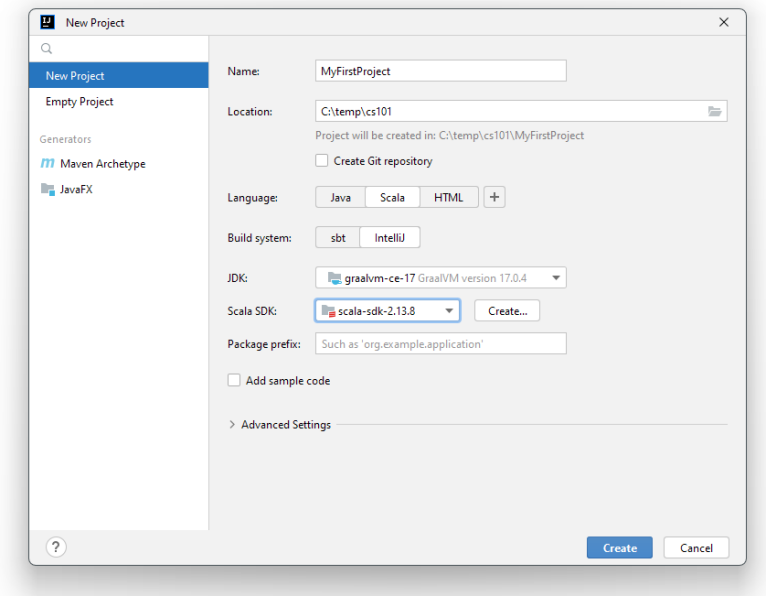

Figure 1: Menu de création du projet

Merci de ne PAS choisir une version 3.0 de Scala car cette version est encore trop instable pour être utilisée dans notre cours. Avant de cliquer, vérifiez que cela ressemble à ce que vous voyez sur la figure 1

Lorsque vous avez terminé, pressez sur le bouton `Create`. Votre premier projet devrait maintenant être crée.

Tâche 3 : Créer votre premier programme

1. Dans *IntelliJ*, faites un clic droit sur le dossier `src`, choisissez `new Scala class`
2. Donnez-lui comme nom `MyFirstCode`, choisissez le type `Object`
3. Complétez le code de manière à avoir *exactement* le même contenu que ci-dessous :

```
object MyFirstClass extends App{
  println("Hello World")
}
```

4. Lancez le programme en cliquant sur la flèche verte.
5. Bravo ! vous avez exécuté votre premier programme `Scala`.

Tâche 4 : Un petit peu plus compliqué

1. Créez un nouveau programme en faisant à nouveau clic droit sur le dossier `src`. Nommez-le `MyProgram`
2. Complétez votre programme pour avoir **exactement** le code suivant.

```
object MyProgram extends App {

  // Declares two values
  val toto: Int = 3
  val titi: Int = 4
```

```
6
7     // Compute the sum of two values
8     val theSum: Int = toto + titi
9
10    // Display the result
11    print("The sum is equal to : ")
12    print(theSum)
13  }
```

3. Peut-être que pendant l'écriture du code vous verrez messages d'erreur ou d'attention. Ne vous en inquiétez surtout pas **tant que vous n'avez pas terminé**. En effet, l'environnement de développement[1] appelle en permanence le compilateur sur votre code pour vous aider à trouver d'éventuelles erreurs ou problèmes mais cela risque surtout de vous stresser au début.

4. Exécutez maintenant votre programme

    1. Cliquez sur la flèche verte à gauche de la ligne contenant `objet MyProgram`.
    2. Vous verrez alors le résultat de votre programme affiché sur la console (la fenêtre au fond de l'écran), à savoir `The sum is equal to : 7`.
    3. Expérimentez en changeant le programme pour avoir une multiplication plutôt qu'une somme. Essayez également de modifier les chiffres pour voir ce qui se passe.

Tâche 5 – Ajouter des fichiers

Comme vous l'avez constaté, il est possible d'avoir plusieurs fichiers *Scala* dans le dossier `src` et donc plusieurs programmes différents que l'on peut lancer.

> Le dossier `src` dans un vrai projet comporte en général beaucoup de fichiers différents dans lesquels on range les différentes choses que l'on veut faire. Certains de ces fichiers peuvent être exécutés grâce à la petite flèche verte ou au menu `Run`. Nous verrons comment spécifier quels fichiers sont exécutables et ce qui les rend spécial dans un prochain labo.

1. Copiez les fichiers `RoomCalc.scala` et `Input.scala` dans le même répertoire que votre fichier`MyProgram.scala`
2. Regardez maintenant dans l'explorateur de projet pour vérifier que `RoomCalc.scala` et `Input.scala` sont bien ajoutés dans les fichiers source (`src`) de votre projet.
3. Vous pouvez exécuter le programme `RoomCalc.scala` en cliquant avec le bouton droit sur le fichier `RoomCalc.scala` dans l'explorateur à gauche et en sélectionnant 'Run "RoomCalc"'.
4. Vous pouvez ensuite cliquer dans la console et entrer les valeurs demandées.
5. Essayez de comprendre ce qui se passe dans le fichier. Bien que nous n'ayons pas vu ces éléments ensemble, vous ne pouvez pas encore comprendre toutes les finesses mais essayez d'imaginer à quoi peuvent server les différents éléments du programme.

1) Modifiez le fichier afin :

    1. D'afficher le message « Volume calculator, by John Doe» (en mettant votre nom à la place de John Doe) au début du programme.
    2. D'afficher le message « Goodbye and thank you » à la fin de l'exécution du programme.
    3. Essayez de calculer et afficher le volume en gallons impériaux (une mesure de volume anglaise). Utilisez pour cela l'information qu'un gallon vaut 0.00454609 m3.

Tâche 4 – Votre consommation d'essence

1. Ajoutez une nouvelle classe à votre projet (`File -> New Class`) que vous nommerez `Fuel`.

2. En vous inspirant de l'exemple ci-dessus, faites un programme permettant de calculer votre consommation d'essence sur une certaine distance. Un exemple possible de ce qui est attendu pourrait-être :

[1] *IntelliJ* est une **IDE**, un acronyme signifiant **I**ntegrated **D**evelopment **E**nvironment, ou environnement de développement intégré.

```
Liters / 100 km : 12
Distance driven : 10
You used : 1.2 liters of fuel.
```

Tâche 5 – Flottaison d'une sphère

Nous voudrions écrire un programme qui permet de déterminer si une sphère en métal creuse flotte si on la plonge dans l'eau. Il doit être possible de choisir la matière qui compose la sphère, l'utilisateur devra donc saisir la masse volumique de cette matière (à titre d'exemple, la masse volumique de l'aluminium est de 2.6 g/cm3).

L'utilisateur doit donc saisir le **rayon** de la sphère (ici le rayon extérieur), **l'épaisseur de la surface** et la **masse volumique** du matériau utilisé.

À partir de ces données, **calculez le volume intérieur de la sphère** ainsi que la **masse de la sphère creuse**. Si le rapport entre la masse et le volume est inférieur à 1, on peut considérer que l'objet flotte. Un exemple d'exécution nous donnerait :

```
Please enter outer sphere radius (in cm): 15
Please enter surface thickness (in cm): 1
Total outer sphere volume: 14137.1669411540 cm3
Enter material density (in g/cm3): 2
Total object density: 0.37392592592592594 => The object is floating
```