Programmation
Licence 1 UPEC 2022/2023
Travaux Machine 1 : Prise en main de Robusta

Le langage utilisé pour cette première année s'appelle **Robusta** une variété de café Java.

Installation

Pour permettre l'utilisation de Robusta quelques petites démarches sont nécessaires :

1. Il faut avoir installé *Java* (JDK 8 ou plus) sur sa machine. Quand vous installez java sur votre machine, vérifiez que vous avez bien installé la JDK (Java Development Kit). La JDK est nécessaire pour *compiler et exécuter* les programmes Robusta.
2. Il faut installer le logiciel *Visual Studio Code* (qu'on appelera par la suite VSCode). Les règles d'installation changent selon le système d'exploitation (Linux, Windows ou Mac). Ce logiciel est nécessaire pour *écrire* les programmes.
3. Il faut télécharger le fichier "robusta.jar" qui contient le code nécessaire pour compiler et éxecuter les programmes. Le fichier se trouve à `https://robusta.oss-us-east-1.aliyuncs.com/vs-tasks-swing-terminal/robusta.jar`. Il faut sauvgarder ce fichier dans un endroit accessible par l'utilisateur. Ce fichier permet de *faire la liaison* entre Robusta et Java.
4. Il faut installer l'extension "Robusta" pour *Visual Studio Code*. Sur EPREL-V2 vous trouvez un vidéo qui vous montre comment installer ce plugin.
5. Il faut dire à ce Plugin où se trouve la JDK, et où se trouve le fichier "robusta.jar"

Un document PDF se trouve également sur EPREL-V2 pour vous aider dans ces étapes. Les étapes 1, 2 et 3 ont déjà été réalisées sur les machines de l'UPEC, mais vous devez les réaliser sur votre propre machine vous serez aidés par vos enseignants.

D'autres instructions se trouvent aussi à la page : `https://github.com/Meshredded/robusta`.

Ecrire, compiler, et éxecuter ses programmes

Dans le logiciel VSCode, il faut ouvrir un dossier avant de commencer à travailler (opération à exécuter en cliquant sur l'onglet File $\mapsto$ Open Folder). Choisissez le dossier sur le quel vous travaillerez et souvenez-vous de ce choix, il sera utile pour les séances suivantes. (Attention, sur les Sessions Windows des machines UPEC vous devez utiliser un repertoire du disque Z : - on vous montrera ça.) Pensez aussi à créer un dossier par séance.

Vous rédigez les programmes dans la fênetre d'écriture de VS Code, et vous les sauvegarderez avec l'extension `.jvs`. (Pourquoi cette extension ? Car Robusta est une évolution de JaVaScool).

Pour compiler un programme nommé `toto.jvs`, vous ferez click droit sur le fichier et l'option de compilation apparaîtra. Une fois compilé correctement, un fichier `toto.jar` apparaît dans votre répertoire. En faisant click droit dessus, vous aurez l'option de l'éxecuter. Un video vous montrant cela se trouve à `https://github.com/Meshredded/robusta`.

Exercices

Exercice 1:

Sur EPREL, téléchargez le fichier `balle.jvs`. Ouvrez ce fichier avec VSCode. Compilez le programme et éxecutez-le pour vous habituer aux deux opérations (compilation puis lancement en exécution).

La vitesse d'affichage dans le programme est régie par un sleep dans le code. Pouvez-vous le trouver ? Modifier la valeur, recompiler et réexécuter le programme avec la nouvelle valeur pour ralentir ou accélérer la vitesse d'affichage.

Exercice 2: **Faites votre première animation**

Sur EPREL, régardez la vidéo *Animation Robusta*. Écrivez un programme Robusta qui réalise la même animation. Les deux nombres à la fin doivent être tirés au hasard entre 1 et 6.

Exercice 3: **Conditionnel**

1. Copier le programme suivant qui affiche une émote :

```
void main() {
  print("(");
  print("o");
  print("_");
  print("o");
  println(")");
}
```

2. Sauvegarder, compiler puis exécuter ce programme.
3. Modifier le programme précédent pour qu'aléatoirement, les yeux soient `o` ou `*`. Cela donne les possibilités `o_*`, `*_*`, `o_o`, et `*_o`.
4. Faire de même pour la bouche : `_` ou `.` et les oreilles : `)` ou `)@` et `(` ou `@(`.
5. Modifier le programme pour que toujours aléatoirement, les yeux soient `o`, `*` ou `-` . Il faudra mettre un `if` dans un `else`.
6. Assurez-vous que toutes les émotes apparaissent avec la même probabilité.

Exercice 4: **Interaction**

Copier le programme suivant qui demande puis affiche l'âge :

```
void main() {
  int age;
  age = readInt("Quel est votre age?");
  if (age == 1) {
    println("Vous avez 1 an");
  } else {
    print("Vous avez ");
    print(age);
    println(" ans");
  }
}
```

Sauvegarder, compiler puis exécuter ce programme.

Exercice 5:

Modifier le programme précédent pour demander en plus le nom et le prénom pour afficher ces informations sous la forme : Laurent Financier, 25 ans. (Bien sûr, le nom et le prénom que votre programme devrait afficher doit dépendre de ce que l'utilisateur saisit !) Respecter le format demandé au caractère près !

Exercice 6:

Copier le programme suivant qui permet de jouer à "pierre, papier, ciseaux", le compiler et l'exécuter.

```
void main() {
  int a;
  int b;
  a = readInt("1:Pierre 2:Papier 3:Ciseaux");
  b = readInt("1:Pierre 2:Papier 3:Ciseaux");
  if (a < 1 || a > 3 || b < 1 || b > 3) {
    println("Coup invalide");
  } else if (a == b) {
    println("Égalité");
  } else if (a == 1 && b == 2) {
    println("Joueur 2 gagne");
  } else if (a == 2 && b == 3) {
    println("Joueur 2 gagne");
  } else if (a == 3 && b == 1) {
    println("Joueur 2 gagne");
  } else {
    println("Joueur 1 gagne");
  }
}
```

Ensuite, modifier le programme pour y ajouter la possibilité de jouer "puits", qui gagne face à "pierre" et "ciseaux" mais perd face à "papier".

Démarche pour validation de votre programme : Lancez votre programme plusieurs fois avec plusieurs variantes de saisies et vérifiez que votre programme fait bien ce qu'on attend de lui dans chaque variante, sinon corrigez-le ! (Dans l'idéal, vous devriez vérifier tous les 4×4 choix de valeurs possibles pour `a` et `b` !)