Section 1

Structures de contrôle

La structure conditionnelle : `if`

- La **structure conditionnelle** permet d'exécuter un bloc d'instructions que si une condition est vérifiée.

```
if (people.isGoodGuy()) {
  Sout.println("Greetings, " + people.getName() + "!");
}
```

```
if (people.isGoodGuy()) {
  Sout.println("Greetings, " + people.getName() + "!");
} else {
  Sout.println("You are not welcome here, "
               + people.getName() + ".");
}
```

La répétition “tant que” : `while(condition)`

- Le `while` permet de répéter un bloc d'instructions tant que sa condition est vraie :

```
int count = 10;
while (count > 0) {
  System.out.println(count + "...");
  count = count - 1;
}
System.out.println("BOUM!!!");
```

Attention

Le test est effectué juste avant d'effectuer la première instruction du bloc à chaque itération !

La répétition bornée : `for`

- Dans certains cas, il est plus direct d'utiliser une boucle `for`, qui isole :
  - l'initialisation de la boucle,
  - la condition d'arrêt,
  - l'instruction de progression.

```
for (int count = 10; count > 0; count = count - 1) {
    System.out.println(count + "...");
}
System.out.println("BOUM!!!");
```

- On peut aussi utiliser `break` et `return` dans une boucle `for`.

La répétition pour toujours (*forever*) : `for(;;)`

- La boucle `for(;;)` permet de répéter un bloc d'instructions tant qu'on ne quitte pas la boucle avec les instructions `break` ou `return`.

```
int count = 10;
for(;;) {
  System.out.println(count + "...");
  if (count <= 0) break;
  count = count - 1;
}
System.out.println("BOUM!!!");
```

Note

on peut avoir plusieurs instructions `break` dans la même boucle.

L'itération pour chaque : `for`

- Pour les objets qui sont des **collections**, implémentant l'interface `Iterable<Elt>`, on peut utiliser la boucle `for` ainsi:

```
for (Elt item : myCollection) {
  item.doSomething();
  // ...
}
```

Exemple de collections

- `List<Elt>`
- les tableaux

Imbrication des structures de contrôle

Les structures de contrôles définissent des blocs d'instructions, entre { et } :

- possible d'utiliser d'autres structures de contrôle dans ces blocs, par **imbrication**,
- les blocs définissent les **portées** des variables, une variable ne vit que dans le bloc où elle est définie et les blocs imbriqués dedans.
- les structures de contrôle imbriquées compliquent la lecture du code par un humain : il ne faut pas en abuser !