Syntaxe générale du langage Javascool / Java

1. Structure générale d'un programme

```
void main() {
	// Déclaration des variables
	// Programme principal
}
```

2. Déclaration des variables et des constantes

```
	type_1 nom_var1; (par exemple: int x;)
	type_2 nom_var2, nom_var3; (par exemple: double y, z;)
	…
	final type_1 nom_const_1 = valeur_constante;
		(par exemple: final double pi = 3.14159;)
```

3. Types

`int, double, boolean, char, String`

Rq : *Les caractères sont encadrés par des simples quotes :* `'o'`
Les chaînes sont délimitées par des doubles quotes : `"chaîne"`

4. Commentaires

```
// Exemple de commentaire
```

5. Séquence

```
{
   Action_1
   Action_2
   …
}
```

6. Action de lecture/écriture

```
variable = readInt(); (ou readDouble(), readString(), etc.)
println(liste des éléments à afficher);
```

7. Affectation

```
variable = expression;
```

8. Action conditionnelle

```
if (condition) {
	actions_si
} else {
	actions_sinon
}
```

9. Choix multiple

```
switch (variable) {
	case valeur_1: actions_1
			break;
	case valeur_2: actions_2
			break;
	…
	default: actions_sinon
}
```

10. Actions itératives

```
for (variable = indice_début; variable <= indice_fin; variable++) {
	actions
}

while (condition) {
	actions
}

do {
	actions
} while (condition)
```

11. Sous-programmes

Syntaxe d'une procédure :

```
void nom_procédure()                    void nom_procédure(paramètres)
{                                       {
  // Déclaration des variables  ou      // Déclaration des variables
  // Actions                            // Actions
}                                       }
```

Appel d'une procédure :

`nom_procédure()` *ou* `nom_procédure(variables ou constantes)`

Les paramètres sont séparés par des virgules et spécifiés sous la forme : `type nom_paramètre`

Les paramètres appartenant à un type de base (`int, double, boolean, char`)sont forcément des paramètres d'entrée, les autres sont des paramètres d'Entrée/Sortie

Syntaxe d'une fonction (les paramètres fonctionnent de manière similaire) :

```
type_retour nom_fonction()                  type_retour nom_fonction(paramètres)
{                                           {
  // Déclaration des variables      ou       // Déclaration des variables
  // Actions                                 // Actions
}                                           }
```

Pour « retourner » une valeur, on place dans le corps de la fonction une ou plusieurs actions de type :

```
return(valeur)
```

Appel d'une fonction :

`Variable = nom_fonction()` *ou* `variable = nom_fonction(variables ou constantes)`

12. Fichiers

Manipulation par des variables de ou type `PrintWriter` (écriture) ou `BufferedReader` (lecture)

Création d'un nouveau fichier (pour écriture) :

```
PrintWriter fichier = new PrintWriter(new FileWriter("monfichier.txt"), false);
```

(si le fichier existait déjà les données sont perdues)

Ajout de données dans un fichier existant (pour écriture) :

```
PrintWriter fichier = new PrintWriter(new FileWriter("monfichier.txt"), true);
```

Désignation d'un fichier existant (pour lecture) :

```
BufferedReader fichier2  = new BufferedReader(new FileReader("monfichier.txt"));
```

Fermeture : `fichier.close();`

Lecture : `String ligne = fichier2.readLine();`

Cette fonction renvoie **null** s'il n'y a plus de lignes à lire dans le fichier.

Ecriture : `fichier.println(données);`

13. Structures

```
class nom_structure
{
        …
}
```

Déclaration de variable : `nom_structure variable = new nom_structure();`

Accès à un élément d'une variable de type structure : `variable.élément`

14. Vecteurs et tableaux

Déclaration de variables :

```
type_éléments[] nom_vecteur = new type_éléments[nombre_éléments]; ou
type_éléments[][]...[] nom_tableau = new type_éléments[dim_1][dim_2]...;
```

Accès à une case à une position donnée :

`nom_vecteur[position]` *ou* *`nom_tableau[pos_1][pos_2][…]`*

15. Opérations diverses

Modulo : `entier_1 % entier_2`

Division entière : `entier_1 / entier_2` (division entière si les deux variables sont des entiers)

Nombre aléatoire entre deux entiers `a` et `b` inclus : `random(a,b)`

16. Pointeurs

En Javascool toute variable n'appartenant pas à un type de base (`int, double, boolean, char`)est forcément un pointeur.

Attention en modifiant les champs d'une variable de type structure...