Série 1

Type int :

- Nombre compris entre -2^{31} et $2^{31}-1$;
- **Priorités des opérations** : $\{*,\ /, \%\} \supseteq \{+, -\}$; % : opérateur modulo
- Division entière dans ce type ;

Boucles : $for\ (int\ i = 0\ ;i <= 9\ ;i++)\{\dots\}$

- Permet de répéter plusieurs fois les mêmes instructions ;
- Numéro d'itération disponible dans une variable appelé compteur de boucle ;
- En-tête formé du mot clé suivi d'une initialisation, d'une condition et d'une incrémentation séparés par des « ; » entre parenthèses :
  - « $int\ i = 0$ » : Déclare le compteur de boucle, son type et sa valeur initiale ;
  - « $i <= 9$ » : Définit sous quelle condition l'exécution de la boucle continue ;
  - « $i++$ » = « $i = i + 1$ » : Incrémentation de la boucle.
- Corps de la boucle (entre accolades) est exécuté à chaque itération de la boucle.

Série 2

Type $String$:

- **Chaîne de caractère** = Texte ;
- Ecrit entre guillemets double ;
- **Chaine vide** : "" ;
- **Opérateur de concaténation** : + (hors des guillemets) ;
- **Séquence $\backslash n$** : Passage à une nouvelle ligne.
- Chaines de caractères composée de caractère représenté par le **type $char$** ; $(char \subseteq String)$
- **Commandes sur les $String$** :
  - **Longueur de chaines de caractère s** : $s.length()$;
    Un caractère s de longueur de chaines n numérote ses caractères de 0 à $n-1$;
  - **Retrouver le i^e caractère d'un caractère s** : $s.charAt(i)$.

Type $char$:

- Compris entre guillemets simples ;
- Peut comprendre une majuscule, un minuscule, un chiffre, un signe de caractères ou encore des caractères spéciaux. Attention : il ne s'agit de qu'un seul caractères et impossible de concaténer.

Conversion de chaines de caractère en entier et vice-versa :

- **Conversion de int en $String$** : « $String\ String.valueOf(int\ x)$ » ;
- **Conversion de $String$ en int** : « $int\ Integer.valueof(String\ ch)$ ».
- Si on met une chaînes de caractère lors de la conversion de $String$ en int, il y aura une erreur à l'exécution du programme.

Expression Booléennes dans les conditionnelles :

$if\ (condition)\{\dots\}\ else\ if\ (condition2)\{\dots\}\ else\ \{\dots\}$

- **Type** : Boolean ;
- **Valeurs** : true ou false ;
- **Expression avec un et (&& en Java)** : Vraie ssi les expressions à gauche et à droite sont vraies ;
- **Expression avec un ou (|| en Java)** : Vraie ssi au moins une des deux expressions est vraies ;
- **Expression avec une négation (! en Java)** : Vraie si l'expression après la négation est fausse ;
- **Priorité des opérateur** : ! prioritaire à && prioritaire à || ;
- On peut combiner ces opérateurs.

Tests sur les chaînes de caractères :

- On ne peut pas tester des chaînes de caractères avec ==. Pour 2 variables $st1$ et $st2$, on test leur égalité avec la commandes : $st1.equals(st2)$;
- On peut aussi directement introduire des chaîne de caractères à la place de variables
- En revanche on peut tester l'égalité de deux caractères avec ==.

Expression conditionnelle :

- **Expression dont la valeur dépend d'une condition** : opérateur ternaire ? ;
- **Syntaxe** : $condition\ ?\ expression1\ :\ expression2$ est la valeur de l'expression 1 si la condition évalué est vraie et expression 2 si la condition évalué est fausse.

Série 3

Accumulateur dans les boucles : Si une **variable est déclarée avant une boucle**, sa valeur persiste d'une itération de la boucle à la suivante.

Boucle imbriquée : Boucle dans une boucle.

Série 4

Tableaux d'entiers :

- Suite de case contenant une valeur ;
- **Type** : « $int[\ ]$ » ;
- **Nombre de case** : « $t.length$ » et allant de 0 à n appelé indices ; $n = (t.length - 1)$
- **Lire la valeur du tableau à la position** i : « $t[i]$ » ;
- **Modifier la valeur** de la position i en e : « $t[i] = e$ » ;
- Attention, ne pas écrire hors des limites du tableau : « Out of Bound ».
- **Le déclarer** : « $int[\ ]t = new\ int[n]$ » = création d'un tableau de taille n.

Série 6

Tableau de chaine de caractère : Type « $String[\,]$ ».

Tableau de tableau :

- **Type** : « $int[\,][\,]$ » ;
- **Créer une taille** : « $new\ int[taille\ du\ tableau][taille\ du\ tableau\ de\ tableau]$ » ;
- Création des tableaux à l'intérieur possible plus tard.

Série 7

Instruction « $switch$ » :

- Prend une instruction int ou $String$ et réalise l'instruction selon la valeur exacte de cette expression ;
- **Utilisation** : On liste avec les « $case$ » les différentes valeurs pour lesquelles on va tester l'égalité et le mot « $default$ » pour traiter les cas non-présents ;
- **Instruction « $break$ »** : Arrête l'exécution du « $switch$ »
- $Switch\ (condition)\{$
  $case\ \ldots : commande\ \ldots;$
  $break;$
  $\ldots$
  $default : commande\ \ldots;$
  $break;\}$

Boucle $while$/boucle non bornée : $while\ (condition)\{\ commande;\}$

- Permet de répéter des instruction tant qu'une expression booléenne est vraie. Utile lorsque l'on ne connait pas le nombre d'itération ;
- Une boucle non bornée ne peut jamais se terminer lorsque sa condition est toujours vraie.

Boucle $do\ while$: $do\{\ commande\ \}\ while\ (condition)\ ;$

- Boucle $while$ mais exécute une fois la commande avant de faire la vérification.

Variable booléenne :

- Utilisé pour vérifier une condition ;
- Appelé « drapeaux » (flag en anglais).

Parcourir les tableaux : à la place de faire des boucle pour un tableau tab

- **Tableaux** : $for\ ([variable]\ x :\ tab)\{\ldots\}$;
- **Tableau de tableaux** : $for\ (int[\,]t : tab)\{for\ (int\ x : t)\{\ldots\}\}$;
- **Chaîne de caractère** : $for\ (char\ c : s.charArray(\,))\{\ldots\}$.

Série 8

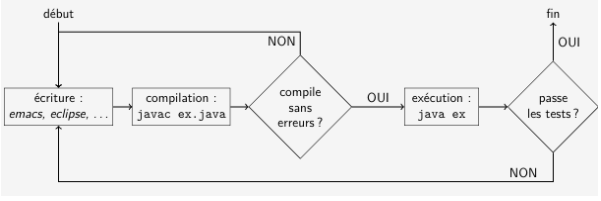
Structure d'un programme Java :

- **Nom de programme** suit le mot clé « $class$ » ;
- **Nom du fichier** doit être le nom du programme suivi du suffixe « $.java$ » ;
- Programme doit **contenir** la fonction « $main$ » ;
- Possibilité de créer des fonctions auxiliaires pour regrouper des suites d'instructions en des « briques » réutilisable en mettant leurs noms.

Structures des fonctions :

- **Fonction** : Série d'instruction qui renvoie une valeur ;
- **Utilité** :
  - Rendre le programme plus compréhensible = On donne un nom clair et logique à la fonction ;
  - Factoriser du code = éviter de répéter les mêmes séries d'instructions.
- **Caractérisation** :
  - Corps : Le bloc d'instruction qui décrit ce que la fonction calcule ;
  - Nom : Par lequel on désignera la fonction ;
  - Paramètres (entrées) : L'ensemble des Variables extérieures à la fonction dont le corps dépend pour fonctionner et changeant à chaque appel de fonction ;
  - Valeur de retour (sortie) : Ce que la fonction renvoie à son appelant.
- **1e ligne de la fonction** : En-tête/prototype (=déclare la fonction) et précise le type de sa valeur de retour, son nom et les types et les noms de chacun de ses paramètres ;
- **Choix des noms de la fonction et de ses paramètres** : Pour faire plus de sens ;
- **Dans le corps de la fonction** : Instructions montrant comment on a la valeur de sortie ;
- **La valeur retournée par la fonction** est indiquée par l'instruction où elle a le même type que celui de la valeur de sortie déclaré en en-tête de la fonction ;
- **Fonction de l'instruction return** : 2 possibles :
  1. Elle **précise la valeur** qui sera fournie par la fonction en résultat ;
  2. Elle met **fin à l'exécution des instructions** de la fonction.
- Possible d'avoir **plusieurs occurrences** de return dans le même corps si l'on a par exemple des conditions ;
- **S'assurer** que tout chemin d'exécution possible du corps d'une fonction mène à un return sinon Java rejette le programme. Il va afficher **l'erreur de compilation** :
  « ExoInvalidMinimum.java:...: missing return statement ».
- **Appel/invocation de fonction** : Utilisation possible de cette fonction partout une fois définie. L'appel d'une fonction doit respecter le prototype de cette fonction, c'est-à-dire le nombre et les types des paramètre ainsi que le type de la valeur de sortie.

Etapes principales de la programmation : →



Passer des paramètres dans la fonction main :

En-tête : « public static void main (String[] args) ».
Prend en paramètre un tableau de chaines de

caractères, où sont stockés les valeurs qui suivent le nom du programme dans la commande qui provoque son exécution.

Série 9

Variable globales : Variable définie en dehors de toute fonction et est accessible pour toutes les fonctions.

Variable locale : Variable définie dans le corps d'une fonction et n'appartient qu'à cette fonction.

Fonctions qui modifient le tableau : On modifie l'adresse du tableau.
Pour modifier son contenu, on doit appeler la fonction dans le main.

Série 10

La récursivité : Technique de programmation où une fonction s'appelle elle-même. Elle repose sur 2 éléments essentiels :

- **Cas de base** : Une condition où la fonction arrête de s'appeler elle-même. Cela empêche une boucle infini ;
- **Appel récursive** : Une situation où la fonction s'appelle avec des donner modifiées pour progresser vers le cas de base.

Remarque :

- Elle caractérise des objets finis ;
- La récursion se termine.

Il faut penser à une représentation finie : Raisonner en anticipant la fin.

Approche itératif : Utilisation de boucle for, on construit la solution depuis le cas de base.