Aide - mémoire Java

Principaux opérateurs

Opérateur d'affectation
• = : affecte (le membre de droite au membre de gauche)
Opérateurs arithmétiques:
Les basiques :
• +
• -
• *
• /
• % (modulo)
Les 2-en-1
• ++ incrément (opérateur unaire postfixé : x++)
• - - décrément (opérateur unaire postfixé : x--)
• += : ajoute le membre de droite au membre de gauche
• - = soustrait membre de droite au membre de gauche
• *= multiplie le membre de gauche par le membre de droite
• /= divise le membre de gauche par le membre de droite
Opérateurs logiques :
• && (ET logique)
• || (OU logique) ('|' : alt gr 6)
• ! (NON logique)
• ? : commutateur (syntaxe : booléen?valeur_si_vrai:valeur_si_faux)
Opérateur de concaténation de chaînes
• +
Opérateurs de comparaison :
• == égalité
• != inégalité
• <
• >
• <=
• >=
Opérateurs sur digits :
• ! : inversion des digits (1 <-> 0)
• & : ET entre digits
• | : OU entre digits ('|' : alt gr 6)
• ^ : OU exclusif (XOR) entre digits ('^' : alt gr 9)

Comment faire pour ....

Créer une classe qui soit un programme	`class Nom_de_classe{`
	`  ...`
	`  public static void main(String[] nom)`
	`  {`
	`  ...`
	`  }`
	`...`
	`}`

Interrompre un programme en cours d'exécution depuis la console	Ctrl C
Afficher un message sur la console de sortie	`System.out.println()`

Changer le type d'une variable (ou d'une valeur) de type primitif	`(nouveau_type_primitif) nom_de_variable`
	ou
	`(nouveau_type_primitif) valeur`

Déclarer une variable de type primitif ou String	`type  nom_de_variable` (ex : `char c, int i, String chaine`…)
Déclarer un tableau	`type_du_contenu[] nom_du_tableau` exemple : `int[] monTableauDEntier;`
Créer un tableau de dimension n	`nom_du_tableau = new  type_du_contenu[n]` `ex : monTableauDEntiers=new int[12];`
Accéder à la valeur de l'élément d'indice *i* d'un tableau	`nom_du_tableau[i]`
Accéder au nombre d'éléments d'un tableau	`nom_du_tableau.length`

Effectuer une séquence d'instructions si une condition est réalisée.	`if (condition){`
	`  .... séquence d'instructions...`
	`}`

Effectuer une séquence d'instructions si une condition est réalisée, et une autre séquence sinon.	`if(condition){`
	`  ...séquence d'instructions`
	`}`
	`else{`
	`  ....autre séquence d'instructions`
	`}`

Sélectionner une séquence d'instruction en fonction de la valeur d'un sélecteur (entier, caractère, chaîne)	`switch (selecteur){`
	`  case valeur1 :`
	`  ... séquence d'instructions...`
	`  break;`
	`  case valeur2:`
	`  ..séquence d'instructions...`
	`  break;`
	`  ....`
	`  default:`
	`  ...séquence d'instructions ...`
	`  break;`
	`}`

Répéter une séquence d'instructions tant qu'une condition est vérifiée	`while (condition){`
	`  ...  séquence d'instructions à répéter...`
	`}`
	
	ou bien
	
	`do  {`
	`  ...séquence d'instructions à répéter..`
	`}`
	`while (condition);`

Répéter une séquence d'instructions *n* fois.	`for (int i=0;i<n;i=i+1){`
	`  ...séquence d'instructions à répéter n fois...`
	`}`

Obtenir un nombre aléatoire compris entre 0 et 1.	`Math.random()`

Déclarer une méthode prenant des paramètres et n'ayant pas de valeur de retour	`void  nom_de_methode(type_du_parametre1 nom_du_parametre1, type_du_parametre2 nom_du_parametre2, ...) {`
	`  ...instructions`
	`}`

Déclarer une méthode prenant des paramètres et ayant une valeur de retour.	`type_de_retour  nom_de_methode(type_du_parametre1 nom_du_parametre1, type_du_parametre2 nom_du_parametre2,...) {`
	`  ...instructions...`
	`return valeur_de_retour;`
	`}`

Utiliser des méthodes et des constantes de classes contenues dans des bibliothèques	`nom_de_bibliotheque.nom_de_classe.nom_de_methode()`
	
	ou
	
	`nom_de_bibliotheque.nom_de_classe.nom_de_variable`

Importer une bibliothèque	`import nom_de_bibliotheque.*;`
Importer une seule classe d'une bibliothèque	`import nom_de_bibliotheque.Nom_de_classe;`

Portée des variables :

Une variable est accessible depuis le bloc de code où elle a été déclarée (sous-blocs inclus). Un bloc de code est un ensemble de lignes de code délimité par des accolades { }
Exception : une variable déclarée dans l'en-tête d'une boucle for est accessible seulement depuis le corps de cette boucle.

Types primitifs :

identifiant	char	byte	short	int	long	float	double	boolean
nature	Caractère ou entier non signé entre 0 et 2^{16}-1	Entier signé entre -128 et 127	Entier signé entre -2^{15} et 2^{15}-1	Entier signé entre -2^{31} et 2^{31}-1	Entier signé entre -2^{63} et 2^{63}-1	Réel en virgule flottante	Réel en virgule flottante double précision	Valeur logique (vrai ou faux, true ou false)