CHAPITRE RE

RECAPITULATIF JAVA

• Commentaires

// pour une ligne
/* pour plusieurs lignes */
/** pour Javadoc */

• Constantes

true, false	Booléenne
null	Non initialisée

• Déclaration de variable

boolean *nomVariable*	Booléenne
char *nomVariable*	Caractère
byte, short, int, long *nomVariable*	Entier
float, double *nomVariable*	Flottant
NomClasse *nomVariable*	Type objet
NomInterface *nomVariable*	Type interface

• Modificateurs de variables de classe et d'instance

static *declaration*	Variable de classe
final *declaration*	Constante
public, private, protected *declaration*	Contrôle d'accès
volatile *nomVariable*	Modification asynchrone
transcient *nomVariable*	Non persistant

• Affectation de variable

nomVariable = *valeur*	Affectation
nomVariable ++, --	Post incrémentation, post décrémentation
++, -- *nomVariable*	Pré incrémentation, pré décrémentation
nomVariable ++	Post incrémentation
NomVariable opérateur= *valeur*	Opération et Affectation

• Opérateurs

argument + *argument*	Addition, concaténation
argument - *argument*	Soustraction
argument * *argument*	Multiplication
argument / *argument*	Division
argument % *argument*	Modulo
argument < *argument*	Strictement plus petit
argument > *argument*	Strictement plus grand
argument <= *argument*	Plus petit ou égal
argument >= *argument*	Plus grand ou égal
argument == *argument*	Test d'identité
argument != *argument*	Test de non identité
argument && *argument*	ET logique partiel
argument || *argument*	OU logique partiel
! *argument*	NON logique
argument & *argument*	ET
argument | *argument*	OU
argument ^ *argument*	OU exclusif
argument << *argument*	Décalage gauche
argument >> *argument*	Décalage droite
argument >>> *argument*	Décalage droite avec remplissage à 0
~ *argument*	Complément binaire
(type) *argument*	Caste
Argument instanceof *NomClasse*	Test d'appartenance à une classe

• Structure de contrôle

{…}	Bloc d'instructions
;	Termine une instruction
if (*condition*) {…} else {…}	Conditionnelle

switch (*expression*) {
case *valeur*:*instructions*
break
case *valeur*:*instructions*
default :*instructions*
}
	Conditionnelle multiple

for (*init* ;*condition* ;*incrémentation*) {…}	Boucle pour
while (*condition*) {…}	Boucle tant que
do {…} while (*condition*) {…}	Boucle tant que test après
break	Sortie de switch et de boucles
continue	Continue la boucle
label :	Nom de label

• Importation

import *Nompackage. NomClasse* ;	Importation d'une classe
import *Nompackage. ** ;	Importation de toutes classes

• Définir une classe

class *NomClasse* {…}	Classe simple
final class *NomClasse* {…}	Ne peut pas être sous classée
public class *NomClasse* {…}	Accessible hors du package
abstract class *NomClasse* {…}	Ne peut pas être instanciée
class *NomClasse* extends *NomSuperClass* {…}	Héritage d'une super classe → Sous classe
class *NomClasse* implements *NomInterfa* {…}	Implémentation d'une interface

• Définir une interface

interface *NomInterface* {…}	Classe simple
interface *NomInterface* extends *Nom* {…}	Sous interface
public interface *NomInterface* {…}	Accessible hors du package

• Méthodes

typeRetour *nomMéthode*() {…}	Méthode simple (void : si rien n'est retourné)
typeRetour *nom* (type arg, type arg, …) {…}	Méthode avec paramètres
private typeRetour *nomMéthode*() {…}	Méthode visible que par la classe
public typeRetour *nomMéthode*() {…}	Méthode visible à tous les packages
protected typeRetour *nomMéthode*() {…}	Méthode visible par toute les sous classes
private protected typeRetour *nom* () {…}	visible par les sous classes d'un même package
return *valeur*	Sortie de la méthode

• Manipulation des objets

new *NomClasse()*	Création d'une instance
objet.nomVariable	Variable d'instance
objet.nomVariableClasse	Variable de classe
objet.nomMethode()	Méthode d'instance
objet.nomMethodeClasse()	Méthode de classe
this	Référence à l'objet courant
Super	Référence à la super classe

• Tableaux

type nomVariable[]	Variable de tableau
type[] *nomVariable*	Variable de tableau
type nomVariable[] []	Variable de tableau à 2 dimensions
new *type* [nombre d'élements]	Déclaration de tableau
nomTableau [index]	Accès à un élément (débute à 0)
nomTableau .length	Accès à la longueur

• Exceptions

try {*instructions*}	Instructions à tester
catch (exception) {*instructions*}	Si problème instructions à exécuter
finally {*instructions*}	Instructions à exécuter avant de quitter

STRUCTURE GENERALE D'UNE CLASSE

```
//Déclaration du package
package nomPackage;
// importation de classes
import nomPackage.NomClasseImportée;
// déclaration de la classe
public class NomClasse extends NomClasse2  implements nomInterface {

// déclaration des attributs
private String varChaine ;
private NomClassex varClassex ;

// constructeurs
        // 1- sans paramètre
        public NomClasse() {
        //Appel du constructeur de la classe mère
        super() ;
        …}
        // 2- paramètre varChaine de type String reçu pour renseigner l'attribut
        public NomClasse(String varChaine ) {
        //Initialise les attributs
        this.varChaine = varChaine ;
                        }
        }
// Méthodes
        // 1- Recevant deux paramètres et ne revoyant rien
        public void nomMethode(int i, boolean a) {
                //traitement
        }
        // 2- Ne recevant rien, retournant un booléen et générant une exception
        public boolean nomMethode2() throws NomClasseException {
                if (condition)  {
                        throw NomClasseException ;
                        return false ;
                        }
                else     return true ;
        }
// getters/setters

}
```

STRUCTURE GENERALE D'UN EXECUTABLE

```
//Déclaration du package
package nomPackage;
// importation de classes
import nomPackage. NomClasseException ;
// déclaration de la classe
public class TestNomClasse {
//Point d'entrée d'une Application
        public static void main (String args[]) {
                // création d'objets
        NomClasse monObjet = new NomDuConstructeurDeClasse();
                //déclaration et initialisation de variable de travail
        int i=10 ;
        boolean a = true;
        //Appel de la méthode de l'objet
        monObjet .nomMethode(i, a) {
        //Affichage du résultat à la console système
                System.out.println(«le résultat est » + monObjet.getVarchaine()) ;
        //Appel de la méthode de l'objet forçant la gestion d'une erreur
        try {
                // monObjet .nomMethode2();
                }
                catch (NomClasseEception e)   {
                System.out.println(«erreur » + e.getMessage()) ;
                }

        }
}
```