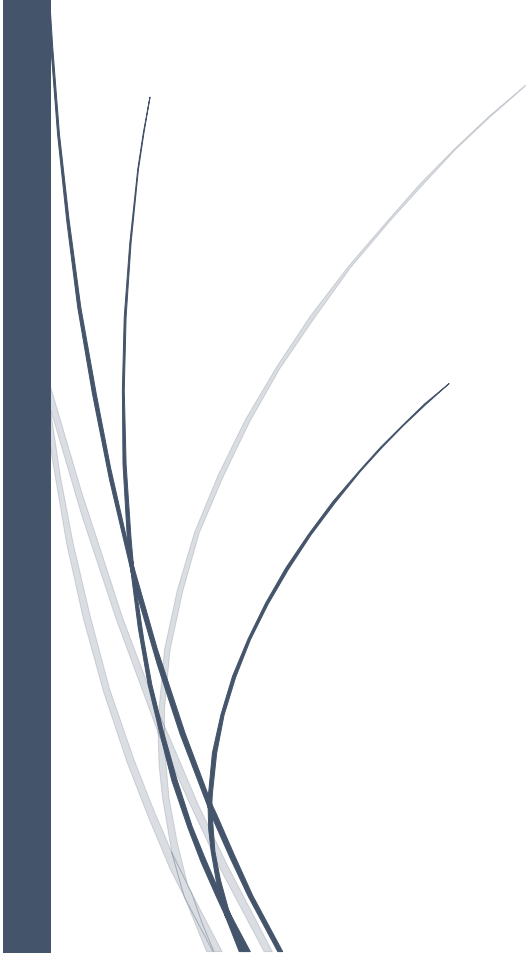
08/02/2024

420-210-MV
Rencontre 05

Programmation orientée objet



1. La conversion de type en Java

Le terme le plus utilisé pour la technique de conversion est « Casting ».

La spécificité de Java, est de convertir (to cast) n'importe quel objet vers un autre objet.

Conversion 1 : String vers les numériques

- Dans cet exemple, il nous faut convertir la String « 20 » vers les différents types numériques.
- On utilise la méthode valueOf(param) qui se trouve dans toutes les enveloppes des nombres primitifs.

De	Vers	
String	Byte	`Byte b1 = Byte.valueOf("20");`
	Short	`Short s2 = Short.valueOf("20");`
	Integer	`Integer i3 = Integer.valueOf("20");`
	Long	`Long l4 = Long.valueOf("20");`
	Float	`Float f5 = Float.valueOf("20");`
	Double	`Double d6 = Double.valueOf("20");`

Conversion 2 : Les numériques vers une String en utilisant la méthode valueOf()

- Dans cet exemple, il nous faut convertir les numériques vers une String.
- On utilise la méthode valueOf(param) de la classe String.

De	Vers	
Byte	String	`Byte by = 20;` `String sb1 = String.valueOf(by);`
Short		`Short sh = 20;` `String sb2 = String.valueOf(sh);`
Integer		`Integer in = 20` `String sb3 = String.valueOf(in);`
Long		`Long lo = 20;` `String sb4 = String.valueOf(lo);`
Float		`Float fl = 20.20f;` `String sb5 = String.valueOf(fl);`
Double		`Double do = 20.2365` `String sb6 = String.valueOf(do);`

Conversion 3 : Les numériques vers une String en utilisant la méthode toString()

- Dans cet exemple, il nous faut convertir les numériques vers une String.
- On utilise la méthode toString() qui se trouve dans tous les objets et qui provient de la super classe « Object »

De	Vers	
Byte	String	`Byte by1 = 20;` `String sbr1 = Byte.toString(by1);`
Short		`Short sh1 = 20;` `String sbr2 = Short.toString(sh1);`
Integer		`Integer in1 = 20;` `String sbr3 = Integer.toString(in1);`
Long		`Long lo1 = 20;` `String sbr4 = Long.toString(lo1);`
Float		`Float fl1 = 20.20f;` `String sbr5 = Float.toString(fl1);`
Double		`Double do1 = 2150.2364;` `String sbr6 = Double.toString(do1);`

Conversion 4 : String vers les primitifs

- Dans cet exemple, il nous faut convertir la String « 20 » vers les différents types numériques.
- On utilise la méthode parse…()

De	Vers	
String	byte	`byte b4 = Byte.parseByte("20");`
	short	`short s4 = Short.parseShort("20");`
	int	`int i4 = Integer.parseInt("20");`
	long	`long l4 = Long.parseLong("20");`
	float	`float f4 = Float.parseFloat("20");`
	double	`double d4 = Double.parseDouble("20");`

Conversion 5 : Primitifs vers String

- Dans cet exemple, il nous faut convertir les primitifs vers une String.
- On utilise la méthode toString() qui se trouve dans tous les objets et qui provient de la super classe « Object »

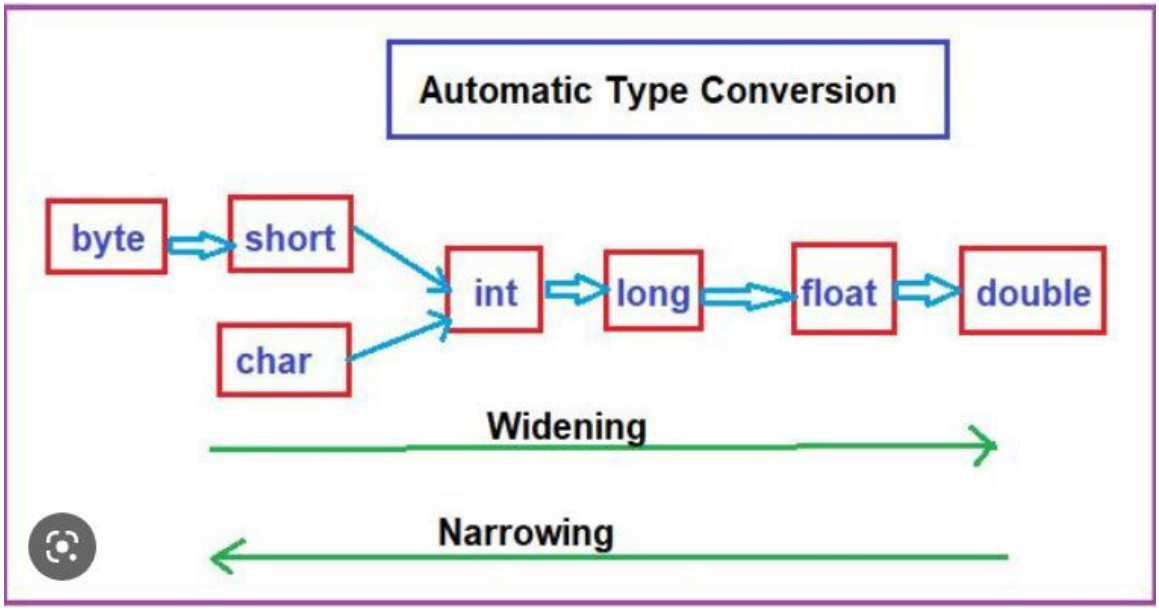
De	Vers	
byte	String	`byte b5 = 20;` `String str1 = Byte.toString(b5);`
short		`short s5 = 20;` `String str2 = Short.toString(s5);`
int		`int i5 = 20;` `String str3 = Integer.toString(i5);`
long		`long l5 = 20;` `String str4= Long.toString(l5);`
float		`float f5 = 20.20f;` `String str5 = Float.toString(f5);`
double		`double d5 = 2150.2364;` `String str6 = Double.toString(d5);`

Conversion 6 : Les primitifs vers une String en utilisant la méthode valueOf()

- Dans cet exemple, il nous faut convertir les numériques vers une String.
- On utilise la méthode valueOf(param) de la classe String.

De	Vers	
byte	String	`byte b6 = 20;` `String str61 = String.valueOf(b6);`
short		`short s6 = 20;` `String str62 = String.valueOf (s6);`
int		`int i6 = 20;` `String str63 = String.valueOf(i6);`
long		`long l6 = 20;` `String str64= String.valueOf(l6);`
float		`float f6 = 20.20f;` `String str65 = String.valueOf(f6);`
double		`double d6 = 2150.2364;` `String str66 = String.valueOf(d6);`

Conversion 7 : Conversion entre les types primitifs



Il y a 2 types de Casting :

Cas 1

Conversion automatique : il s'agit d'une conversion d'un type plus petit en un type plus grand

byte -> short -> char -> int -> long -> float -> double

Type	Conversion automatique				
byte b = 20;	short s = b;	Int I = b;	long l = b;	float f = b;	double d = b;
short s = 20;		Int i = s	long l = s;	float f = s;	double d = s;
int i = 20;			long l = i;	float f = i;	double d = i;
long l = 20;				float f = l;	double d = l;
float f = f;					double d= f;

Cas 2

Conversion manuelle : il s'agit de la conversion d'un type plus grand en un type de taille plus petite. Il suffit de rajouter le casting du type voulu à droite.

double -> float -> long -> int -> char -> short -> byte

```
double d = 2000.23;
```

```
float f = (float)d;
```

```
long l = (long) f;
```

Pour résumer :

- Dans le cas d'une conversion de petit vers grand ➔ on ne fait rien.
- Dans le cas d'une conversion d'un grand vers un petit ➔ il faut « Caster » la partie droite en type du petit.

Les énumérations

Une énumération va nous permettre de définir un ensemble de constantes qui sont fonctionnellement liées entre elles. La déclaration se fait de la manière suivante :

```
public enum Jour
{
  DIMANCHE,
  LUNDI,
  MARDI,
  MERCREDI,
  JEUDI,
  VENDREDI,
  SAMEDI
}
```

La première valeur de l'énumération est initialisée à zéro. Les constantes suivantes sont ensuite initialisées avec un incrément de un. La déclaration précédente aurait donc pu s'écrire :

```
public class Jour
{
  public static final int DIMANCHE=0;
  public static final int LUNDI=1;
  public static final int MARDI=2;
  public static final int MERCREDI=3;
  public static final int JEUDI=4;
  public static final int VENDREDI=5;
  public static final int SAMEDI=6;
}
```

C'est approximativement ce que fait le compilateur lorsqu'il analyse le code de l'énumération.

En fait, la déclaration d'une énumération est une déclaration de classe « déguisée ». Cette classe hérite implicitement de la classe java.lang.Enum. Les éléments définis dans l'énumération sont les seules instances possibles de cette classe.

La portée d'une énumération suit les mêmes règles que celle des variables.

Une variable de type énumération peut facilement être utilisée dans une structure switch ... case. Il n'est dans ce cas pas nécessaire de faire précéder les membres de l'énumération du nom de l'énumération.

```
switch (repere)
{
  case LUNDI:
  case MARDI:
  case MERCREDI:
  case JEUDI:
      System.out.println("C\'est dur de travailler");
      break;
  case VENDREDI:
      System.out.println("Bientôt le week end !");
      break;
  case SAMEDI:
      System.out.println("Enfin !");
       break;
  case DIMANCHE:
       System.out.println("Et ça recommence !");
       break;
}
```