Types

Type primitif	Taille (bits)	Valeur par défaut	Wrapper
byte	8	0	Byte
short	16	0	Short
int	32	0	Integer
long	64	0L	Long
float	32	0.0f	Float
double	64	0.0d	Double
char	16	'\u000'	Character
boolean	1 (non précisé)	false	Boolean
-	16 (SB capacity)	""	String

Interface

valeurs public static final / *méthodes* public abstract

Peut contenir des méthodes **private** ou **default** qui seront alors implémentées.

Overrider ses méthodes ne peut se faire avec un accesor plus restrictif.

Classe

class *MyClass* ***extends*** *OtherClass* ***implements*** *interface1, interface2*

Une classe **abstract** n'est pas tenue d'implémenter les méthodes abstraites héritées (classe ou interface), c'est le rôle des classes concrètes.

Contient des membres **static** et d'**instance**. Une méthode statique ne peut accéder qu'aux champs statiques de la classe.

Un **constructeur** par défaut est fourni à la compilation si aucun n'est fourni.

- **this** peut invoquer les méthodes overrided ou overloaded
- **super** invoque les membres des classes parent
- Un membre **final** doit être initialisée avant celle de la classe (assignation directe, ou par constructeur ou initializer)

Enum

valeurs public static final

Classe particulière contenant des constantes énumérées. Chaque constante est une instance unique de la valeur.

Méthodes natives: values() / valuesOf() / name() / ordinal()

Peut contenir des champs, constructeurs, méthodes. Peut implémenter des interfaces.

Enums spécialisés: **EnumMap** et **EnumSet**.

Methode

Une méthodes peut avoir des **paramètres**, mais on lui fournit des **arguments**.

Type d'argument

- **Primitive** - une copie est utilisée sans modifier la valeur originale
- **Wrapper** - le pointeur modifie l'objet original

Retour

Toujours un **Wrapper**, jamais une primitive.

Overloading

Différentes versions d'une méthode avec des **signatures différentes** (paramètres et retour).

Overriding

Surcharge d'une méthode héritée, à la **signature identique**.

L'accesor doit être aussi permissif. Impossible sur une méthode private évidemment...

Enum example

```
public enum Jour {
        LUNDI, MARDI, MERCREDI;
}
public interface Descri ptible {
        String getDes cri pti on();
}
public enum Jour implements Descri ptible {
        LUN DI( " Lun ") {
                @Ov erride
                public String getDes cri ption() {
                        return "1st day week";
                }
       }, MARDI( " Mar "), MERCRE DI( " Mer ");
        private final String abrev; // Champ
       // Constr ucteur (toujours private)
        Jou r(S tring abrev) {
                thi s.abrev = abrev;
       }
       // Méthode
        public String getAbrev() {
                return abrev;
       }
}
```