Chapitre 1

Types en Java

1.1 Types

Le langage Java est un langage statiquement typé. Les types de Java sont construits de la manière suivante :

- types de base : si τ est un type de base alors τ est un type ;
- tableaux : si τ est un type alors $\tau[]$ est un type ;
- classes : si C est une classe alors C est un type ;
- interfaces : si I est une interface alors I est un type.

1.2 Sous-types et typage dynamique

En Java la relation d'héritage entraine la relation de sous-typage ($\leq$).

- si τ est un type alors $\tau \leq \tau$;
- si SC est sous-classe de C, alors $SC \leq C$;
- si SI est sous-interface de I, alors $SI \leq I$;
- si C implante I alors $C \leq I$;
- si $\tau_2 \leq \tau_1$, alors $\tau_2[] \leq \tau_1[]$

Si $\tau_2 \leq \tau_1 T$, alors toute valeur de type τ_2 peut etre utilisée en place et lieu d'une valeur de type τ_1.

Soit le programme Java suivant :

```
class A {
    int m1 (A x) { System.out.println(1+" "); return (1);}
    boolean n1 (A x, A y) {System.out.println(4+" "); return (false);}
}

class B extends A {
    int m2 (B x) { System.out.println(5+" "); return (5);}
    int m1 (A x) { System.out.println(6+" "); return (6);}
    boolean n2 (B x, B y) {System.out.println(8+ " "); return (false);}
}
```

```
class ex0 {
    public static void main(String [] args) {
        A a1 = new A ();
        B b1 = new B();
        A a2 = b1;
        B b2 = (B)a2;;

        System.out.println("*1-------");
        a1.m1(a1);
        b1.m2(b1);
        b1.m1(b1);
        a1.n1(a1,a1);
        b1.n2(b1,b1);
        System.out.println("*2-------");
        a1.m1(b1);
        b1.m2((B)a2);
        b1.m1(b1);
        a1.n1(b1,b1);
        b1.n2((B)a2,b2);
        System.out.println("*3-------");
        a2.m1(a1);
        a2.n1(b1,b1);

    }
}
```

Son exécution donne :

```
*1-------
1
5
6
4
8
*2-------
1
5
6
4
8
*3-------
6
4
```

1.3 Surcharge

La surchage en Java est résolue statiquement. Une méthode est du point de vue des types un couple contenant le nom de la classe de définition et le produit cartésien des types des paramètres de la méthode.

Dans l'ensemble des méthodes compatibles du point de vue des types, le compilateur Java va choisir celle de plus petit type définie dans la classe de plus petit type. Néanmoins il peut exister des cas d'ambiguité.

1. premier exemple

```
class A {
    int m (A x) { System.out.println(1+" "); return (1);}
    boolean n (A x) {System.out.println(2+" "); return (true);}
    int m (A x, A y) {System.out.println(3+" "); return 3;}
    boolean n (A x, A y) {System.out.println(4+" "); return (false);}
}

class B extends A {
    int m (B x) { System.out.println(5+" "); return (5);}
    boolean n (B x) {System.out.println(6+" "); return (true);}
    int m (B x, B y) {System.out.println(7+" "); return 7;}
    boolean n (B x, B y) {System.out.println(8+ " "); return (false);}
}


class ex1 {
    public static void main(String [] args) {
        A a1 = new A ();
        B b1 = new B();
        A a2 = b1;
        System.out.println("*1-------");
        a1.m(a1);
        a1.n(a1);
        a1.m(a1,a1);
        a1.n(a1,a1);
        System.out.println("*2-------");
        b1.m(b1);
        b1.n(b1);
        b1.m(b1,b1);
        b1.n(b1,b1);
        System.out.println("*3-------");
        b1.m(a1);
        b1.n(a1);
        b1.m(a1,a1);
        b1.n(a1,a1);
        System.out.println("*4-------");
        a2.m(a1);
        a2.n(a1);
```

```
        a2.m(a1,a1);
        a2.n(a1,a1);
        System.out.println("*5-------");
        a2.m(a2);
        a2.n(a2);
        a2.m(a2,a2);
        a2.n(a2,a2);
        System.out.println("*6-------");
        b1.m(a1);
        b1.n(b1);
        b1.m(a1,b1);
        b1.n(a1,b1);

    }
}
```

2. sortie du premier exemple

```
*1-------
1
2
3
4
*2-------
5
6
7
8
*3-------
1
2
3
4
*4-------
1
2
3
4
*5-------
1
2
3
4
*6-------
1
6
3
```

```
4
```

3. deuxième exemple : dépendance croisée

```
class A {
    int m (A x) { System.out.println(1+" "); return (1);}
    boolean n (B x) {System.out.println(2+" "); return (true);}
    int m (A x, A y) {System.out.println(3+" "); return 3;}
    boolean n (B x, B y) {System.out.println(4+" "); return (false);}
}

class B extends A {
    int m (B x) { System.out.println(5+" "); return (5);}
    boolean n (A x) {System.out.println(6+" "); return (true);}
    int m (B x, B y) {System.out.println(7+" "); return 7;}
    boolean n (A x, A y) {System.out.println(8+ " "); return (false);}
}


class ex2 {
    public static void main(String [] args) {
        A a1 = new A ();
        B b1 = new B();
        A a2 = b1;
        System.out.println("*1-------");
        a1.m(a1);
        a1.n(b1);
        a1.m(a1,a1);
        a1.n(b1,b1);
        System.out.println("*2-------");
        b1.m(b1);
        b1.n(b1);
        b1.m(b1,b1);
        b1.n(b1,b1);
        System.out.println("*3-------");
        b1.m(a1);
        b1.n(a1);
        b1.m(a1,a1);
        b1.n(a1,a1);
        System.out.println("*4-------");
        a2.m(a1);
        a2.n(b1);
        a2.m(a1,a1);
        a2.n(b1,b1);
```

```
        System.out.println("*5-------");
        a2.m(a2);
        a2.n((B)a2);
        a2.m(a2,a2);
        a2.n((B)a2,(B)a2);
        System.out.println("*6-------");
        b1.m(a1);
        b1.n(b1);
        b1.m(a1,b1);
        b1.n(a1,b1);

    }
}
```

4. sortie du deuxième exemple

```
*1-------
1
2
3
4
*2-------
5
2
7
4
*3-------
1
6
3
8
*4-------
1
2
3
4
*5-------
1
2
3
4
*6-------
1
2
3
```

```
8
```

5. troisième exemple

```
class A {
    int m (A x) { System.out.println(1+" "); return (1);}
    boolean n (B x) {System.out.println(2+" "); return (true);}
    int m (A x, A y) {System.out.println(3+" "); return 3;}
    boolean n (A x, B y) {System.out.println(4+" "); return (false);}
}

class B extends A {
    int m (B x) { System.out.println(5+" "); return (5);}
    boolean n (A x) {System.out.println(6+" "); return (true);}
    int m (B x, B y) {System.out.println(7+" "); return 7;}
    boolean n (B x, A y) {System.out.println(8+ " "); return (false);}
}


class ex3 {
    public static void main(String [] args) {
        A a1 = new A ();
        B b1 = new B();
        A a2 = b1;
        System.out.println("*1-------");
        a1.m(a1);
        a1.n(b1);
        a1.m(a1,a1);
        a1.n(b1,b1);
        System.out.println("*2-------");
        b1.m(b1);
        b1.n(b1);
        b1.m(b1,b1);
        b1.n(b1,b1);
        System.out.println("*3-------");
        b1.m(a1);
        b1.n(a1);
        b1.m(a1,a1);
        b1.n(b1,b1);
        System.out.println("*4-------");
        a2.m(a1);
        a2.n(b1);
        a2.m(a1,a1);
        a2.n(b1,b1);
```

```
        System.out.println("*5-------");
        a2.m(a2);
        a2.n((B)a2);
        a2.m(a2,a2);
        a2.n((B)a2,(B)a2);
        System.out.println("*6-------");
        b1.m(a1);
        b1.n(b1);
        b1.m(b1,b1);
        b1.n(b1,b1);

    }
}
```

6. sortie du troisième exemple

```
ex3.java:31: reference to n is ambiguous, both method n(A,B) in A and methoc
        b1.n(b1,b1);
          ^
ex3.java:36: reference to n is ambiguous, both method n(A,B) in A and methoc
        b1.n(b1,b1);
          ^
ex3.java:51: reference to n is ambiguous, both method n(A,B) in A and methoc
        b1.n(b1,b1);
          ^
3 errors
```

7. cas d'ambiguité

```
class A {
    int m (A x) { System.out.println(1+" "); return (1);}
    int m (A x, A y) {System.out.println(3+" "); return 3;}
}

class B extends A {
    int m (B x) { System.out.println(5+" "); return (5);}
    int m (A x, B y) {System.out.println(7+"bis  "); return 7;}
    int m (B x, A y) {System.out.println(7+"ter  "); return 7;}
}


class ex4 {
    public static void main(String [] args) {
        A a1 = new A ();
        B b1 = new B();
```

```
        A a2 = b1;
        System.out.println("*1-------");
        a1.m(a1);
        a1.m(a1,a1);
        System.out.println("*2-------");
        b1.m(b1);
        b1.m(b1,b1);
        System.out.println("*3-------");
        b1.m(a1);
        b1.m(a1,a1);
        System.out.println("*4-------");
        a2.m(a1);
        a2.m(a1,a1);
        System.out.println("*5-------");
        a2.m(a2);
        a2.m(a2,a2);
        System.out.println("*6-------");
        b1.m(a1);
        b1.m(b1,b1);

    }
}
```

8. sortie de la compilation

```
ex4.java:24: reference to m is ambiguous, both method m(A,B) in B and method m(B,A)
        b1.m(b1,b1);
          ^
ex4.java:36: reference to m is ambiguous, both method m(A,B) in B and method m(B,A)
        b1.m(b1,b1);
          ^
2 errors
```