```
class Point {
 double latitude;
 double longitude;
 double altitude;

 Point (final double x, final double y, final double z) {
  this.latitude = x; this.longitude = y; this.altitude = z;}}
```

La définition d'un constructeur ressemble un peu à la définition d'une fonction : un constructeur a des arguments dont on indique le type, il a un corps qui est exécuté au moment de l'appel du constructeur. Mais

- un constructeur doit toujours avoir le même nom que le type auquel il appartient,
- dans la définition d'un constructeur, on n'indique jamais le type de l'objet retourné,
- on n'utilise jamais l'instruction `return` dans le corps d'un constructeur,
- dans le corps d'un constructeur, on peut affecter les champs d'un enregistrement `this` qui est l'enregistrement construit.

Il est possible de définir plusieurs constructeurs pour un même type. Comme ces constructeurs ont nécessairement le même nom, ils sont surchargés et doivent donc être distingués par le nombre ou le type de leurs arguments. Par exemple, il est possible d'ajouter un constructeur binaire au type `Point` qui construit un point d'altitude nulle.

Quand aucun constructeur n'est défini, il y a un constructeur par défaut sans arguments, que l'on a utilisé ci-dessus dans l'expression `new Point()`. Ce constructeur par défaut n'est plus utilisable dès qu'un autre constructeur a été défini.

Enfin, il est aussi possible dans la déclaration d'un type de donner des valeurs par défaut pour chaque champ

```
class Point {
 double latitude = 90.0;
 double longitude = 0.0;
 double altitude = 0.0;}
```

Dans ce cas, si un constructeur n'affecte pas l'un des champs de l'objet `this`, la valeur par défaut utilisée n'est pas `0.0`, mais celle donnée dans cette définition.

4.1.6 La sémantique des enregistrements

Pour étendre la fonction Σ aux enregistrements, il est nécessaire de lui ajouter un cinquième argument qui est la liste des types construits, chaque