La classe java.math.BigInteger

Temps de lecture : 2 minutes

Pourquoi `BigInteger` ?

En Java, on peut représenter des entiers en utilisant les types primitifs `byte`, `short`, `int` et `long`.

Cependant, ceux-ci sont limités en taille. Par exemple, la déclaration suivante provoque une erreur de compilation :

```
long tooLong = 11111111111111111111L; // Dépasse 9223372036854775807L
```

La taille de long permet de couvrir beaucoup de cas d'utilisation de la vie réelle car le nombre maximale est déjà très grand (il comporte 19 chiffres).

Cependant, si des calculs plus poussés nécessitent des entiers plus grand que long, nous avons besoin d'utiliser la classe `BigInteger`.

Utilisation de `BigInteger`

Il est possible de définir des `BigInteger` de la manière suivante :

```
// Dépasse 92233720368547758071 mais pas de problème
var notTooLongAnymore = new BigInteger("11111111111111111111");
```

Le constructeur reçoit ici une chaîne de caractères comme paramètre d'entrée. La raison est assez logique : on veut pouvoir représenter des entiers plus long que la taille maximale de long. Il n'est donc pas possible de passer un long en entrée.

Point d'attention : Si la chaîne de caractère passé à `BigInteger` n'est pas un entier, un `NumberFormatException` sera levée à l'exécution. Il ne faut donc pas oublier la gestion d'erreurs lors de la création d'un `BigInteger`.

Méthodes de calcul

La classe `BigInteger` déclare des méthodes de calculs afin de permettre leur manipulation :

```
var bigInt = new BigInteger("11111111111111111111");
var bigInt2 = new BigInteger("34");
bigInt.add(bigInt2); // +
bigInt.subtract(bigInt2); // -
bigInt.multiply(bigInt2); // *
bigInt.divide(bigInt2); // /
bigInt.pow(2);
bigInt.sqrt();
```

Copier

Contrairement à `java.lang.Math`, les méthodes ne sont pas statiques et s'appliquent directement aux objets `BigInteger`.

Note : Il n'y a pas de méthodes de calculs trigonométriques dans la classe `BigInteger`.