Java: Généricité

```java
class Pair<F, S> {
        // premier élément
        public final F fst;
        // second élément
        public final S snd;
        // constructeur
        public Pair(F fst, S snd) {
                this.fst = fst;
                this.snd = snd;
        }
        static public <F, S> Pair<S, F> swap(Pair<F, S> p) {
                return new Pair<S, F>(p.snd, p.fst);
        }
}
```

```java
Pair<String, Integer> nameAge = new Pair<String, Integer>("Jean", 17);
Pair<Integer, String> ageName = Pair.swap(nameAge);
```

- Pour des raisons historiques, la généricité en Java possède les limitations suivantes :
  1. la création de tableaux dont les éléments ont un type générique est interdite;
  2. les tests d'instance impliquant des types génériques sont interdits;
  3. les transtypages sur des types génériques ne sont pas sûrs, c-à-d qu'ils produisent un avertissement lors de la compilation et un résultat potentiellement incorrect à l'exécution;
  4. la définition d'exceptions génériques est interdite.

- La version brute d'un type interagit avec la version générique de ce même type de la manière suivante :
  - une version générique peut être utilisée partout où la version brute est attendue, sans provoquer de message à la compilation,
  - la version brute peut être utilisée partout où une version générique est attendue, mais cela provoque un avertissement du compilateur.

Java: Try-catch-finally

```
code normal 1
try {
      instructions dangereuses bloc interrompu en
      cas de levée d'exception
}
catch (TypeException1 variable) {
      code à exécuter si erreur de ce type
}
catch (TypeException2 variable) {
      code à exécuter si erreur de ce type
}
finally {
      code à toujours exécuter
}
code normal 2
```