# POGL 2020 – Spécification des opérations

Un des objectifs de la phase d'analyse est de préciser le comportement attendu de chacune des fonctionnalités du système à développer. Cela apparait dès la première phase d'analyse, dans la présentation en tableau de la description des cas d'utilisation, avec en particulier les lignes suivantes (illustrées pour un cas d'utilisation « vente d'un ticket pour une visite guidée ») :

**Précondition.** Indique les conditions devant être réunies pour que le cas d'utilisation puisse être activé. Par exemple : l'agent du musée est connecté sur sa machine de caisse.

**Postcondition.** Indique l'effet obtenu après réalisation du cas d'utilisation. Par exemple : la place est réservée, le billet est imprimé, le paiement est encaissé.

**Situations d'erreur.** Indique les cas particuliers pouvant donner lieu à des comportements alternatifs. Par exemple : la visite demandée est déjà complète, ou le paiment est impossible.

**En cas d'erreur.** Indique les effets des comportements alternatifs correspondant aux situations d'erreur. Par exemple : le billet n'est pas imprimé et aucun paiement n'est encaissé (fonctionne dans les deux situations), et éventuellement dans la première situation une autre visite est proposée.

Ce principe se transpose dans les phases de conception, par exemple lors de la définition des classes et de leurs méthodes. Supposons une méthode de la signature suivante :

```
public t m(t1 p1, ..., tn pn)
```

Nous parlons donc d'une méthode `m` prenant en entrée $n$ paramètres `p1` à `pn` de types respectifs `t1` à `tn` et ayant un type de retour `t` (éventuellement, `t` est `void`). Lors de la conception de cette méthode, il convient de préciser les éléments suivants, qui forment la **spécification** de la méthode :

— une précondition qui décrit, dans l'hypothèse d'un appel de méthode `obj.m(o1, ..., on)`, les conditions sur le paramètre implicite `obj` et sur les paramètres `o1` à `on` pour que l'appel de méthode se déroule convenablement ; et

— une postcondition qui décrit le résultat de l'appel de méthode, cette description comportant :

1. une description du résultat renvoyé par la méthode (dans le cas où le type de retour `t` n'est pas `void`), et
2. une description de l'état des objets `obj` et `o1` à `on` (dans le cas où la méthode est susceptible de modifier des attributs de l'un des objets).

**Exemple 1.** La méthode `double sqrt(double d)` de calcul de racine carrée a la spécification suivante :

**Précondition.** Le paramètre `d` doit être positif ou nul. Remarque : le fait que le paramètre `d` doivent être de type `double` ne fait **pas** partie de la précondition ; cette information est déjà présente dans la signature de la méthode.

**Postcondition.**

1. Le résultat renvoyé par la méthode est la racine carrée du paramètre `d`.
2. L'état des objets n'est pas modifié.

**Exemple 2.** La méthode `void add(T o)` d'ajout d'un élément à un `ArrayList` a la spécification suivante :

**Précondition.** Pas de précondition (la méthode peut toujours être appelée).

**Postcondition.**

1. Pas de résultat renvoyé.
2. Après l'appel `l.add(o)`, la liste `l` contient les mêmes éléments qu'avant, plus l'objet `o`.