Instanciation

Le processus de creation d'un objet a partir de sa classe (son moule) est appele **l'instanciation**. A partir d'une classe on peut instancier au tant d'objet qui sont similaire.

Un objet n'est l'instance que d'une seule classe. L'heritage n'a rien a voir la dedans. La vrai classe d'un objet est "celle qui est tout en bas".

Cycle de vie d'un objet

- Dynamique comme toute autre valeur: Creer utilise, puis detruit pendant l'execution.
- La construction et Destruction sont deux phases tres importantes (et formalisees) dans la vie d'un objet.

Construction

Elle repond a deux problematiques:

- Vision atomique de la phase de creation. Bien fabriquer l'objet en entier.
- Assurer la coherence interne (aspect agregatifs). Verifier que l'objet ai du sens.

Le constructeur est une procedure speciale. (Pas de type de retour). La construction c'est uniquement des effets de bord.

Ils sont fourni par defaut (mais modifiable). Les nom des constructeur sont predefini.

Exemple C++

Definition:

```
Human::Human (const std::string& name, gender g, unsigned birth_year)
: name_ (name), gender_ (g), birth_year_ (birth_year)
{}
```

Instanciation:

```
// Pour la pile
Human h1 = Human ("Alain Terieur", gender::male, 1970);
Human h2 ("Alain Terieur", gender::male, 1970);
Human h3 { "Alain Terieur", gender::male, 1970 };
Human h4 = Human { "Alain Terieur", gender::male, 1970 };

// Pour le tas
Human* h5 = new Human ("Corinne titgoutte", gender::female, 1990);
auto h6 = std::make_unique<Human>
  ("Justine Titgoutte", gender::female, 1995);
```

Exemple Java

Definition:

```
class Human
{
  String name;
  gender gender;
  int birthYear;
  Human (String _name, Gender _gender, int _birthYear)
  {
    name = _name;
    gender = _gender;
    birthYear = _birthYear;
  }
}
```

Instanciation:

```
Human h1 = new Human ("Alain Terieur", Gender.male, 1970);
```

Destruction

Un destructeur doit:

- Etre atomique, avoir lieu sans interuption
- Assurer le netoyage (liberation de la memoire)

C'est formaliser dans les langages de POO classique. Le concept de destructeur a de sens uniquement lorsque l'on gere la memoire a la main.

Un destructeur n'a pas de valeur de retour. Il est fourni par defaut (mais modifiable). Celui par defaut va juste nettoyer la memoire.

Exemple C++

Definition:

```
Human::~Human()
{}
```

Appel:

```
// Called automatically for stack-allocated objects
// Called by explicit pointer deletion:
delete h5;
// Called automatically for smart pointers
```

Garbage collector

C'est un composant logiciel qui fait le menage a notre place. Dans un langage a GC la gestion de la memoire est automatique. On ne soucie pas de free les objets.

Le tout premier garbage collector a ete invente au debut des annees 70 avec *Lips*. Avant les annees 70 on pensait que c'etait une bonne chose que le programeur est le controle de tout. A cette epoque la connaissance informatique tenait dans un cerveau.

En *java* / *C#* il y a un GC. On a donc pas de destructeur. Cependant il y a des fonctions comme **finalize** qui va etre appele lorsque le GC fait le menage. On ne sait donc pas quand un finaliseur s'execute.