**APOO semaine 2**

Associations :

Il est possible d’alléger une classe objet en créant une autre classe objet qui se rajoutera par la suite sur notre classe. C’est ce qu’on appelle une association. C’est lorsqu’une classe possède un attribut qui est une autre classe, cela permet d’alléger la classe.

La flèche indique une association. Elle par de personne a Adresse car la classe Personne possède un attribut de type Adresse

**EX :**

|  |  |  |
| --- | --- | --- |
| **Personne** |  | **Adresse** |
| **-prenom : String**  **-nom : String**  **-jour : int**  **-mois : int**  **-annee : int**  **-domicile : Adresse** |  | **-rue : String**  **-numero : String**  **-codePostal : String**  **-ville : String** |
| **+Personne (nPrenom : String, nNom : String, nJour : int, nMois : int, nAnnee : int, nDomicile : Adresse)**  **+calculerAge() : int**  **+calculerAgeEn(an : int) :int** |  | **+Adresse (nRue : String, nNumero : String, nCodePostal : String,**  **nVille : String )**  **+getRue : String**  **+setVille (ville :String) : String** |

Encapsulation :

L’encapsulation est le fait de limité l’accès à l’état ou au comportement.

Il existe : private et public. **On se doit de les mettre dans l’UML avec – pour private et + pour public**

**Pour récupérer ou changer les valeurs d’une variable private on utilise les méthodes getter ou setter.**

*Getter :*

*getNomAttribut(), cette méthode permets d’obtenir la valeur de la variables .*

*isNomAttribut() pour les booléen.*

*Setter :*

*setNomAttribut(nouvelleValeur : typeVariable) : void //en UML*

*public void setNomAttribut(typeVariable nouvelleValeur)*

La représentation des objets en mémoire

// comment représenter le objets en mémoire

pers adr



|  |  |  |
| --- | --- | --- |
| :Personne |  | :Adresse |
| nom = «Dupont»  prenom = « jean »  anneeNaissance = 432  moisNaissance =2  jourNaissance = 34  domicile = |  | rue = « rue Neuve »  numero = « 34 »  codePostal = « 4454 »  ville = « Ath » |