IUT2 Université Grenoble Alpes
Département STID

Programmation Objet

R3.VCOD.10

Martin LENTSCHAT

martin.lentschat@univ-grenoble-alpes.fr

Les principales notions de la POO

- Classe (avec ses Attributs et Méthodes) et Objet/Instance
- Envoi de messages
- Encapsulation
- Héritage
- Polymorphisme
- **Associations entre classes**
- Agrégation et Composition

IUT2 Université Grenoble Alpes
Département STID

Associations entre classes

- **Deux classes** peuvent être **associées** via leurs attributs.
- Cela permet en particulier la **navigation entre les classes** par le biais de l'envoi de messages

On associe une personne à une voiture

```python
import voiture

class Personne:
    def __init__(self, nom : str, prenom : str, adresse : str, v : voiture.Voiture):
        self.firstname = prenom
        self.lastname = nom
        self.address = adresse
        self.voiture = v

    def get_voiture(self):
        return self.voiture
```

personne.py

```python
class Voiture:
    def __init__(self, num : str, coul: str):
        self.immat = num
        self.couleur = coul

    def get_couleur(self):
        return self.couleur
```

voiture.py

Envoi d'un message à p1 pour connaitre sa voiture, puis envoi d'un message à la voiture de p1 pour connaitre sa couleur

```python
import personne
import voiture

v1 = voiture.Voiture("BH-346-BH", "Blanc")

p1 = personne.Personne("Durand", "Paul", "Grenoble", v1)

print(p1.get_voiture().get_couleur())
```

main.py

IUT2 Université Grenoble Alpes
Département STID

Exercice

1. Définissez une classe *Pilote* avec les attributs d'objet *name* et *car* (initialisé à *None*)
2. Définissez une classe *Voiture* avec les attributs d'objet *model* et *condition* (initialisé à *100*) et une méthode *usure()* diminuant la valeur de *condition* de 1
3. Créez une instance *p1* de *Pilote* et une instance *v1* de *Voiture*
4. Créez une méthode *buy_car()* dans *Pilote,* permettant d'associer une *Voiture* à un *Pilote*
5. Créez une méthode *rouler()* dans *Pilote*, celle-ci doit afficher un message et diminuer la *condition* de la *Voiture* du *Pilote* de 1
6. Créez une méthode *build_car()* dans *Pilote*, permettant de créer une nouvelle *Voiture* et de l'associer au *Pilote*

   a) Qu'arrive-t-il si on exécute *p1.buy_car(v1)* puis *p1.build_car(…)* ?

   b) Qu'arrive-t-il à sa *Voiture* si un *Pilote* est supprimé ?

IUT2 Université Grenoble Alpes
Département STID

Association multiples entre classes

- Cas d'un *Pilote* qui a plusieurs *Voitures* ?

  Il y a plusieurs possibilités pour gérer les associations multiples entre groupes d'objets :

  1) Ajouter un attribut à *Pilote* (une liste) contenant les instances d'objets *Voiture* associés (☹)
  2) Définir une classe gérant les associations entre instances d'objets

IUT2 Université Grenoble Alpes
Département STID

Exercice

1. En partant de l'exercice précédent, créer une classe *Possede* avec des attributs d'objet *driver* et *car* et un attribut de classe *relations* (une liste vide)
2. Créer des méthodes *buy_car()* et *loose_car*() dans *Possede* permettant de créer, ajouter et supprimer des associations entre *Pilote* et *Voiture* dans *relations*
3. Créer les méthodes *driven_cars()* et *car_drivers()* permettant de récupérer toutes les *Voitures* d'un *Pilote* et les *Pilotes* d'une *Voiture*
4. Créer la méthodes *sell_car()* permettant de transférer une *Voiture* d'un *Pilote* à un autre.
   a) Bonus : créer/modifier les méthodes des classes *Pilote* et *Voiture* pour pouvoir y appeler les méthodes de *Possede*
5. Question:

- Quelles sont les différences entre l'approche de cet exercice et lister les *Voitures* comme attribut de la classe *Pilote* ?