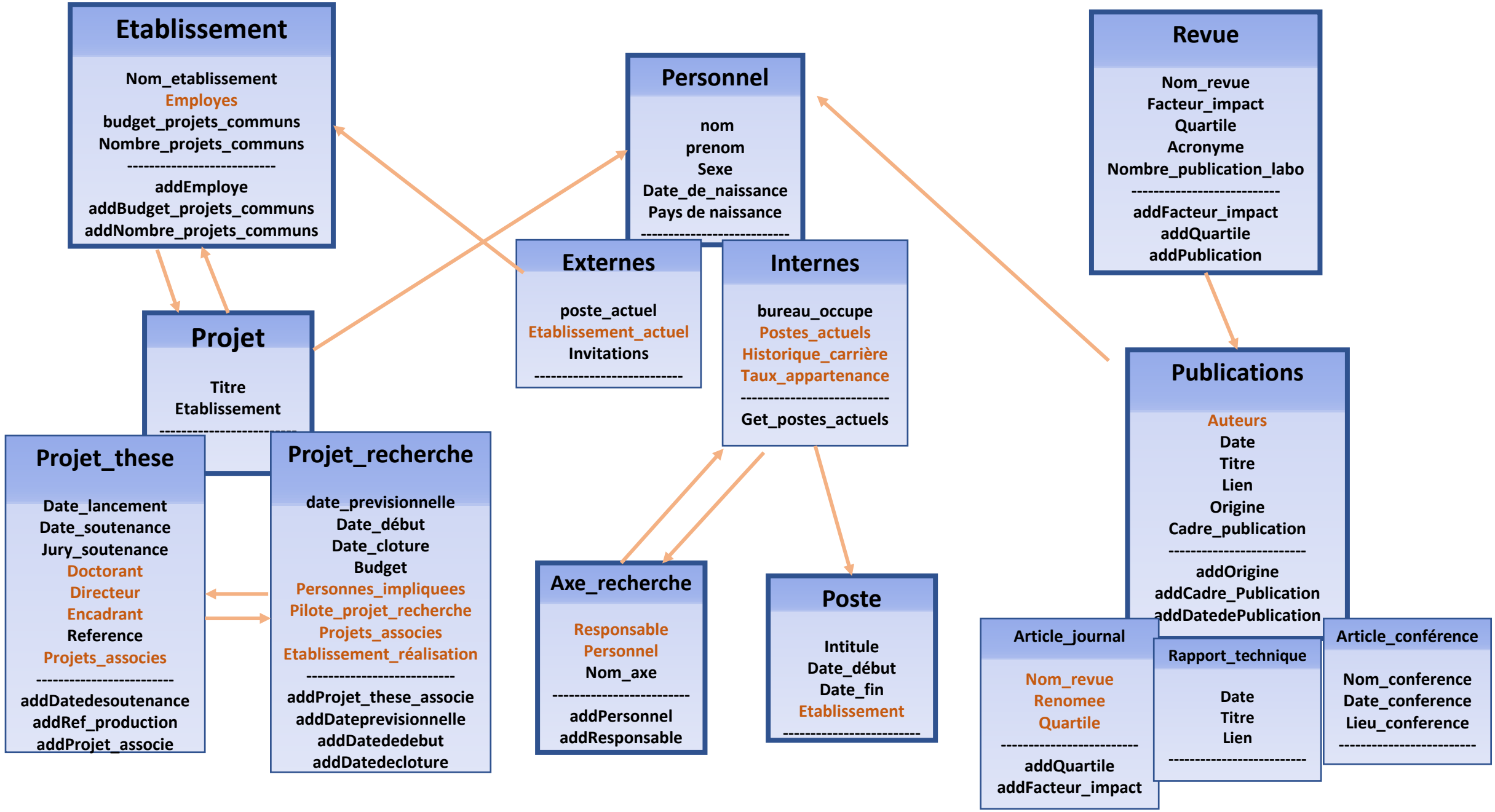
Object Oriented Programming

Project MINI_OOP – 2020_S2

Zhang Alex


Etablissement
Nom_etablissement
Employes
budget_projets_communs
Nombre_projets_communs
addEmploye
addBudget_projets_communs
addNombre_projets_communs
Personnel
nom
prenom
Sexe
Date_de_naissance
Pays de naissance
Revue
Nom_revue
Facteur_impact
Quartile
Acronyme
Nombre_publication_labo
addFacteur_impact
addQuartile
addPublication
Externes
poste_actuel
Etablissement_actuel
Invitations
Internes
bureau_occupe
Postes_actuels
Historique_carrière
Taux_appartenance
Get_postes_actuels
Projet
Titre
Etablissement
Publications
Auteurs
Date
Titre
Lien
Origine
Cadre_publication
addOrigine
addCadre_Publication
addDatedePublication
Projet_these
Date_lancement
Date_soutenance
Jury_soutenance
Doctorant
Directeur
Encadrant
Reference
Projets_associes
addDatedesoutenance
addRef_production
addProjet_associe
Projet_recherche
date_previsionnelle
Date_début
Date_cloture
Budget
Personnes_impliquees
Pilote_projet_recherche
Projets_associes
Etablissement_réalisation
addProjet_these_associe
addDateprevisionnelle
addDatededebut
addDatedecloture
Axe_recherche
Responsable
Personnel
Nom_axe
addPersonnel
addResponsable
Poste
Intitule
Date_début
Date_fin
Etablissement
Article_journal
Nom_revue
Renomee
Quartile
addQuartile
addFacteur_impact
Rapport_technique
Date
Titre
Lien
Article_conférence
Nom_conference
Date_conference
Lieu_conference

```python
class Internes(Personnel): #classe fille de Personnel
    def __init__(self, lenom, leprenom, lesexe,ledate_de_naissance, lepays_de_naissance, lebureau_occupe,leh    rique_carriere, letaux_appartenance):
        super().__init__(lenom, leprenom, lesexe,ledate_de_naissance, lepays_de_naissance) #appel des méthod    e la classe mère
        self._bureau_occupe = str(lebureau_occupe)#chaine de caractère car les bureaux peuvent avoir des nom    mple
        self._historique_carriere = list(lehistorique_carriere) #sous forme d'une liste de liste avec les in    imbriquées
        self._taux_appartenance = dict(letaux_appartenance) #liste de 3 flottants compris entre 0 et 1 dans    dre (Taux_conception, Taux_fabrication, Taux_commande)
        self._postes_actuels = []  #un interne peut occuper 2 postes au seins de l'établissement


    #Assesseurs
    def get_bureau_occupe(self):
        return self._bureau_occupe
    def get_historique_carriere(self):
        return self._historique_carriere
    def get_taux_appartenance(self):
        return self._taux_appartenance


    # méthodes dela classe
    def get_postes_actuels(self):
        self._postes_actuels=[] #réinitialisation. Sinon la méthode ajoute le même poste plusieurs fois
        for chaqueposte in self._historique_carriere:
            if chaqueposte.date_fin == None :
                self._postes_actuels.append(chaqueposte.intitule)
        return self._postes_actuels


    #passage en propriété pour simplifier l'accès ensuite
    bureau_occupe= property(get_bureau_occupe)
    postes_actuels = property(get_postes_actuels)
    historique_carriere =  property(get_historique_carriere)
    taux_appartenance = property(get_taux_appartenance)


    def __repr__(self):
        return "{} {}".format(self.nom, self.prenom)
```

Constructors

Accessors / mutators

Methods

Properties

Repr

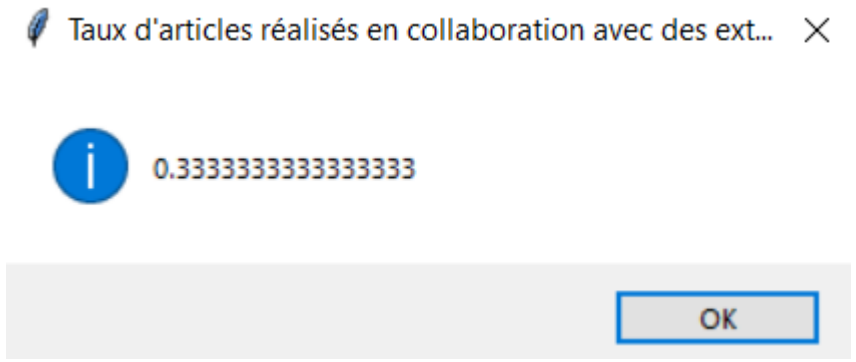
Specific features

```
589
590         self.Dict_Etablissement={}
591         self.Dict_Axes={}
592         self.Dict_Projets_These={}
593         self.Dict_Projets_Recherche={}
594         self.Dict_Projets = {}
595         self.Dict_Rate_Directeur={} #clé : Nom Objet : rate
596         self.Dict_Rate_Encadrant={} #clé : Nom Objet : rate
597         self.Dict_Role_Jury={}
598         self.Dict_Poste = {}
599         self.Dict_Budget_Etablissements_Realisation={}
600         self.Dict_Revues={}
601         self.Dict_Quartiles={}
602         self.Dict_Facteur_Impact={}
603         self.Dict_Article_Journal={}
604         self.Dict_Article_Conference={}
605         self.Dict_Article_Technique={}
606         self.Dict_Quartiles={}
607         self.Dict_Facteur_Impact={}
608         self.Dict_Organisation={}
609
```

Dictionaries based functionalities

```
for qqun in self.Dict_Internes.keys():
    sonage=(dt.date.today()-(self.Dict_Internes[qqun].date_de_naissance))
```

Datetime module

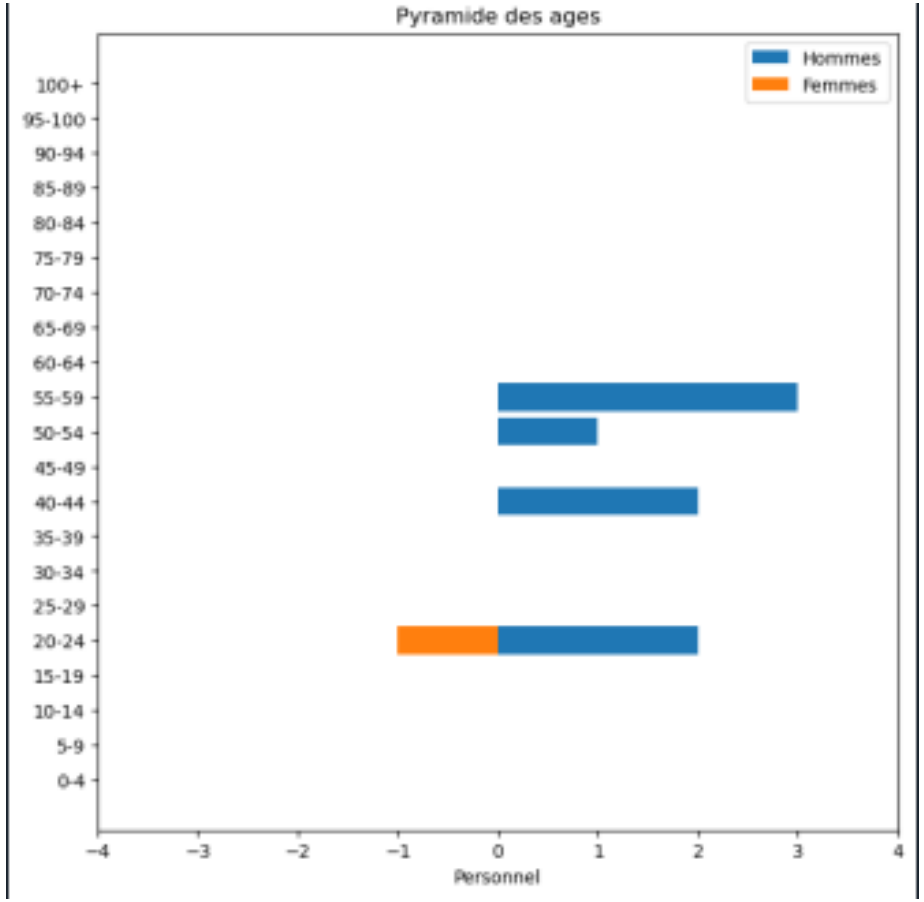


Interface with message boxes / pop-up windows

Processing

```
#Le taux d'articles rédigés en collaboration avec des personnels externes

def taux_articles_externes():
    Nombre_Articles= len(Dict_Publication.keys())
    Nombre_Publi_exterieures = 0
    for pub in Dict_Publication.keys():
        ecrit_avec_auteur_exterieur = False
        for chaque_auteur in Dict_Publication[pub].auteurs:
            if not chaque_auteur in Dict_Internes.keys():
                ecrit_avec_auteur_exterieur= True
        if ecrit_avec_auteur_exterieur == True:
            Nombre_Publi_exterieures+=1
    return Nombre_Publi_exterieures/Nombre_Articles
```



Achievements

Tasks	Functional ?
Implement classes	OK
Read xml file	OK (no matter the xml file)
Processing	OK (everything is implemented in the interface)
Visualize information	OK
Save information	Average (Interns / Externs saving OK)