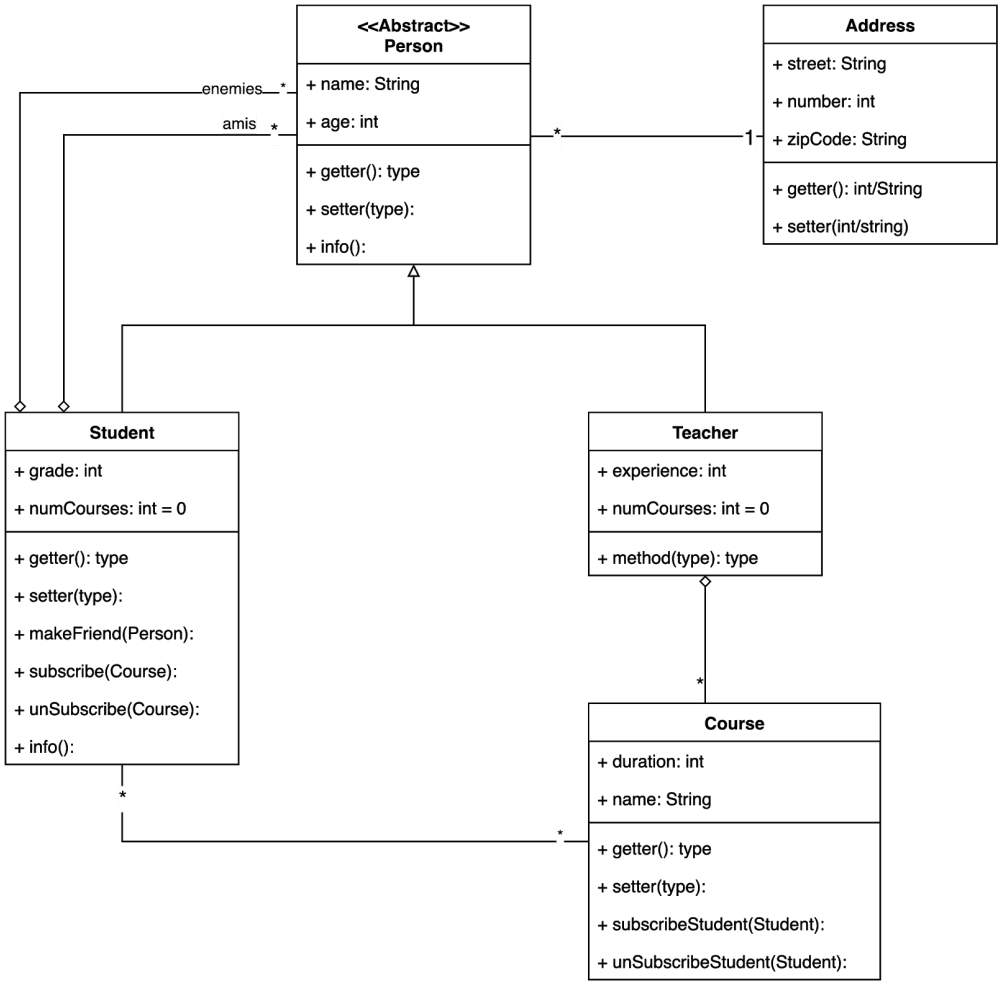
TP 4 : Refactoring



1) Refactoring réalisé

```
39      /* Extract Method
40       * Il y a les duplications du code
41       * Possibilité d'extraire dans une méthode pour faciliter la compréhension
42       * On veut afficher tous les informations de la personne
43       */
44      public void info() {
45          System.out.println("Name : " + name);
46          System.out.println("Age : " + age);
47
48          //mon adresse
49          printMyAddress();
50      }
51
52      private void printMyAddress() {
53          System.out.println("Address number : " + address.getNumber());
54          System.out.println("Street name: " + address.getStreet());
55          System.out.println("Zip code : " + address.getZipCode());
56      }
57
58  }
```

```java
public void makeFriend(Person person) {
    /* Extract Method
     * Il y a une possibilité d'extraire une/deux méthode afin de faciliter la compréhension des conditions
     */
    if(!isAFriend(person) && !isAnEnemy(person)) {
        friends.add(person);
    }else {
        System.out.println("Impossible d'ajouter cette instance dans la liste.");
    }
}

private boolean isAnEnemy(Person person) {
    return enemies.contains(person);
}

private boolean isAFriend(Person person) {
    return friends.contains(person);
}
```

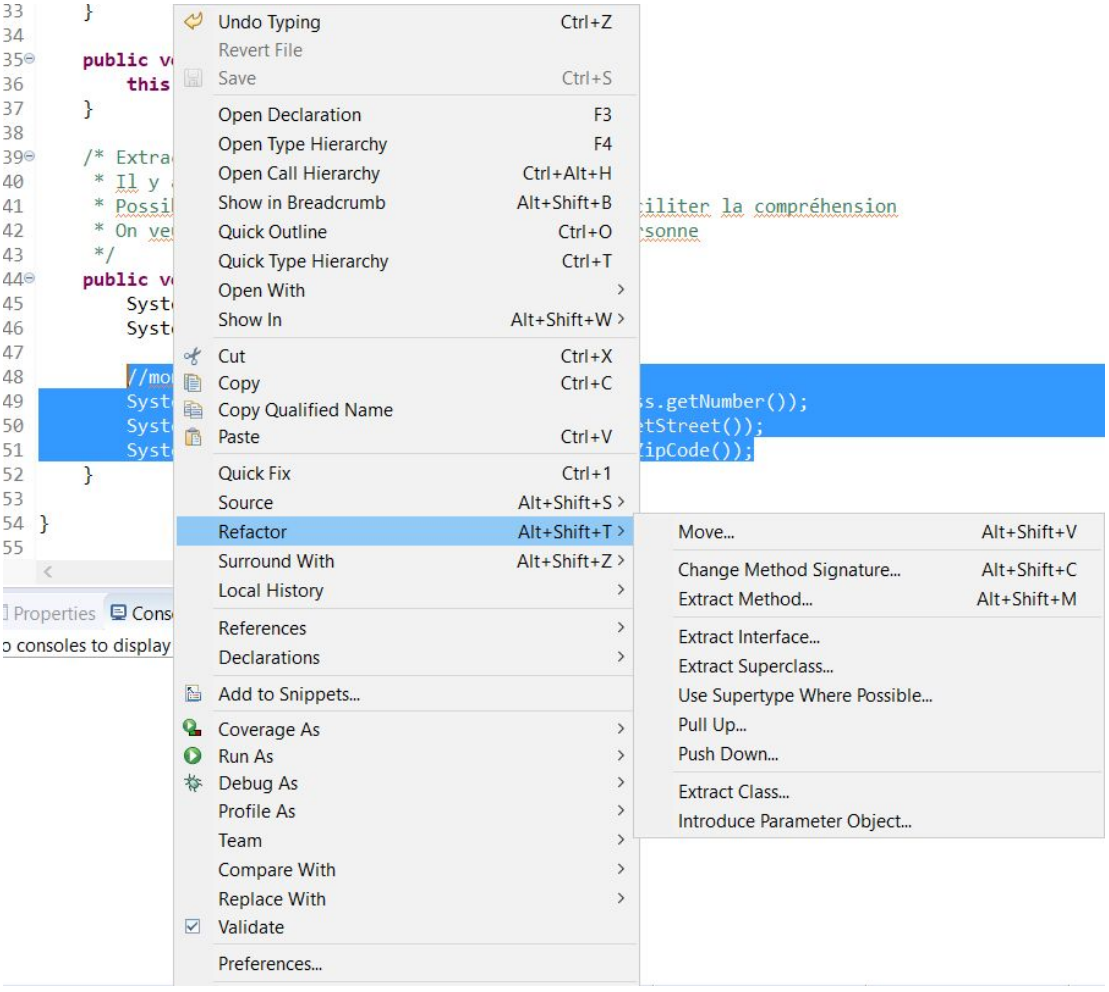
```java
public void info() {
    super.info();

    System.out.println("Grade : " + grade);

    printFriends();
    printEnemies();
    printMyCourses();
}

private void printMyCourses() {
    for(Course course : courses) {
        System.out.println("Course name : " + course.getName());
        System.out.println("Course duration : " + course.getDuration());
        System.out.println("Course teacher : " + course.getTeacher().getName());
    }
}

private void printEnemies() {
    System.out.print("Mes enemies : ");
    for(Person person : enemies) {
        System.out.print(person.getName() + ", ");
    }
}
```

2) Procédure appliqué

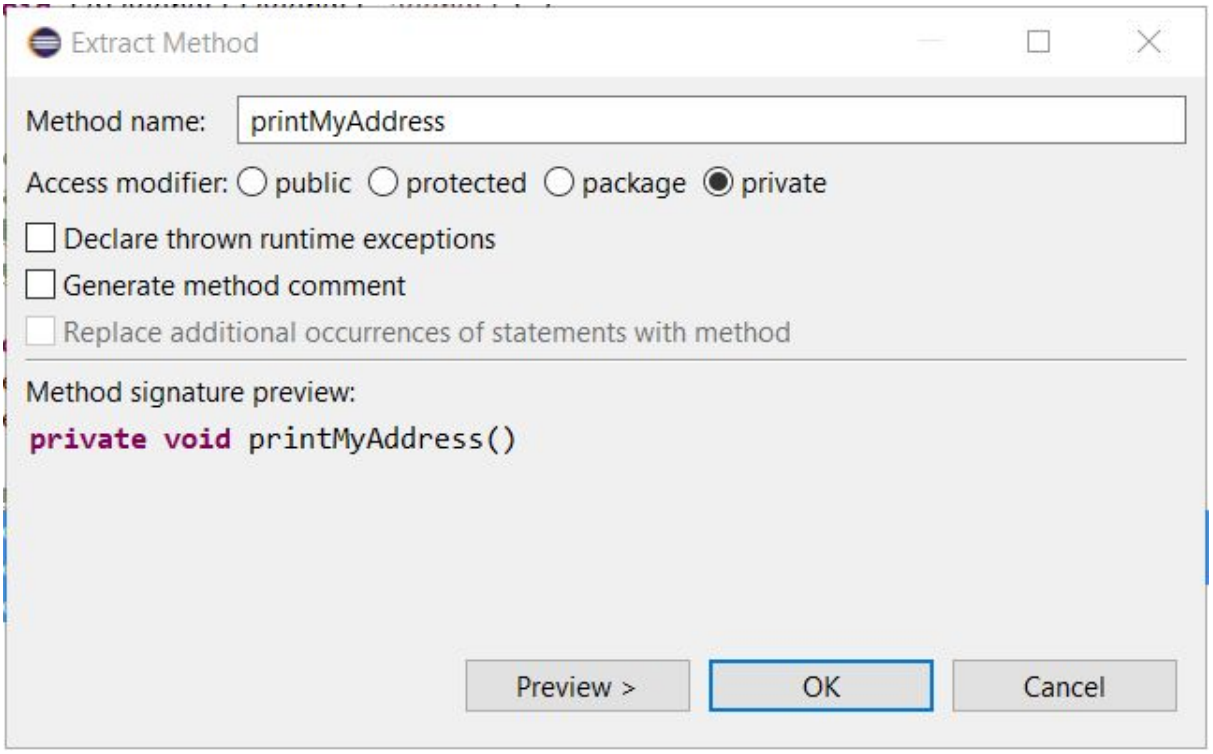
Etape 1 : Sélectionner la partie à extraire dans une méthode

```
39      /* Extract Method
40       * Il y a les duplications du code
41       * Possibilité d'extraire dans une méthode pour faciliter la compréhension
42       * On veut afficher tous les informations de la personne
43       */
44      public void info() {
45          System.out.println("Name : " + name);
46          System.out.println("Age : " + age);
47
48          //my address
49          System.out.println("Address number : " + address.getNumber());
50          System.out.println("Street name: " + address.getStreet());
51          System.out.println("Zip code : " + address.getZipCode());
52      }
```

Etape 2 : Cliquer droit sur la partie sélectionnée et choisir “extract method”



Etape 3 : Nommer la nouvelle méthode et cliquer sur “Ok”



Etape 4 : Résultat

```
39      /* Extract Method
40       * Il y a les duplications du code
41       * Possibilité d'extraire dans une méthode pour faciliter la compréhension
42       * On veut afficher tous les informations de la personne
43       */
44      public void info() {
45          System.out.println("Name : " + name);
46          System.out.println("Age : " + age);
47
48          //mon adresse
49          printMyAddress();
50      }
51
52      private void printMyAddress() {
53          System.out.println("Address number : " + address.getNumber());
54          System.out.println("Street name: " + address.getStreet());
55          System.out.println("Zip code : " + address.getZipCode());
56      }
57
58  }
```

3) Intérêt

Grâce à l'extraction de la méthode, j'ai pu rendre le code plus compréhensive. Les méthodes ne sont plus inondés des lignes ainsi que les conditions "if" sont plus faciles à comprendre avec le nom de la méthode que des morceaux de code compressé (par ex: courses.contains(course);). Le coût de la maintenance serait aussi plus rapide car le code est séparé en plusieurs parties. Le code est plus facile à lire quand les tâches sont séparées et bien nommées.

4) Avis sur la mise en oeuvre sous eclipse

La manoeuvre sous Eclipse était simple et rapide. Il est nécessaire seulement de faire quelques clics pour extraire un morceau du code. De plus, les partie, où il y a la même signature du code que celle qu'on a sélectionné, seront aussi changé en nouveau écriture ce qui est très pratique.

5) Test s'est bien déroulé ou nécessite les modifications

Le résultat suite à la modification reste pareil. Il n'y a pas de modification à faire.

6) Comparaison avec le site refactoring.com/catalog

Sur le catalog, l'auteur a extrait le morceau du code vers une méthode de la méthode actuelle alors que Eclipse propose d'extraire vers une méthode privée dans la classe (voir la partie 1). Les façons sont tout à fait correctes mais avec la façon du catalog, nous n'allons pas pouvoir rappeler cette méthode extraite dans une autre méthode.

```
function printOwing(invoice) {
  printBanner();
  let outstanding  = calculateOutstanding();
  printDetails(outstanding);

  function printDetails(outstanding) {
    console.log(`name: ${invoice.customer}`);
    console.log(`amount: ${outstanding}`);
  }
}
```