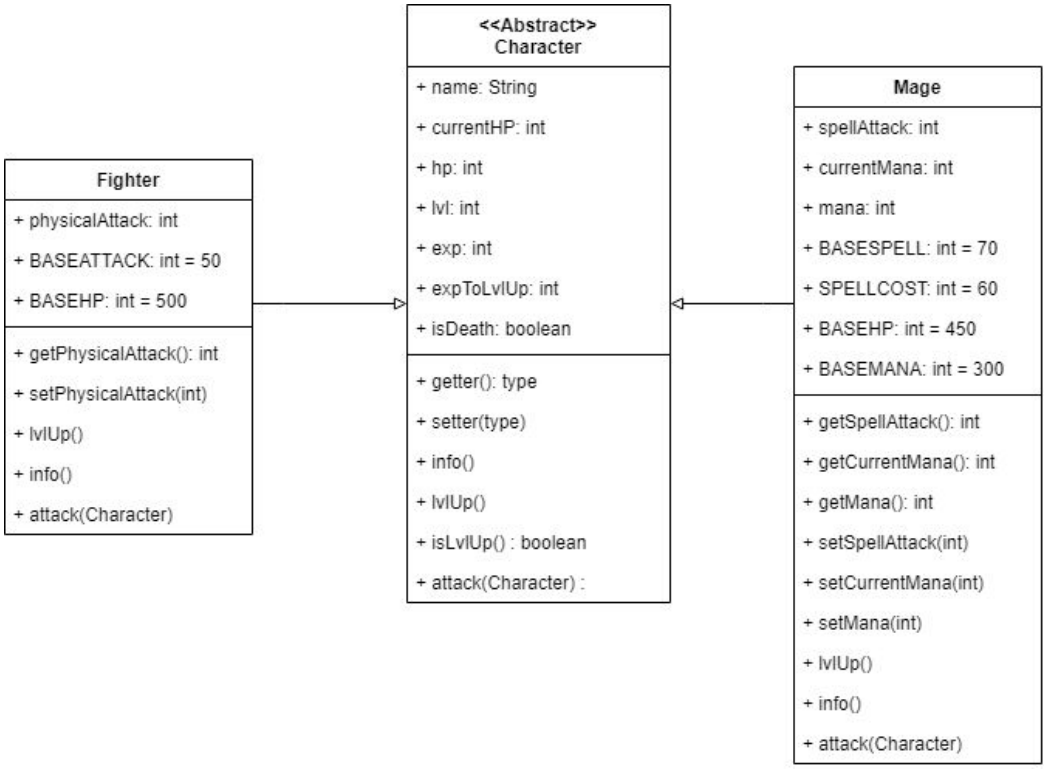
TP 4 : Refactoring

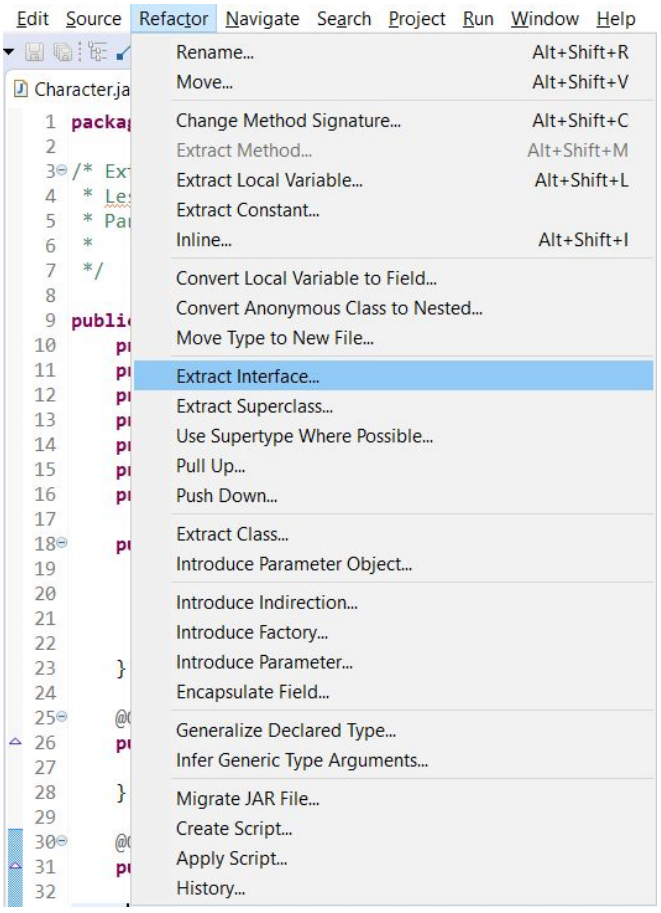


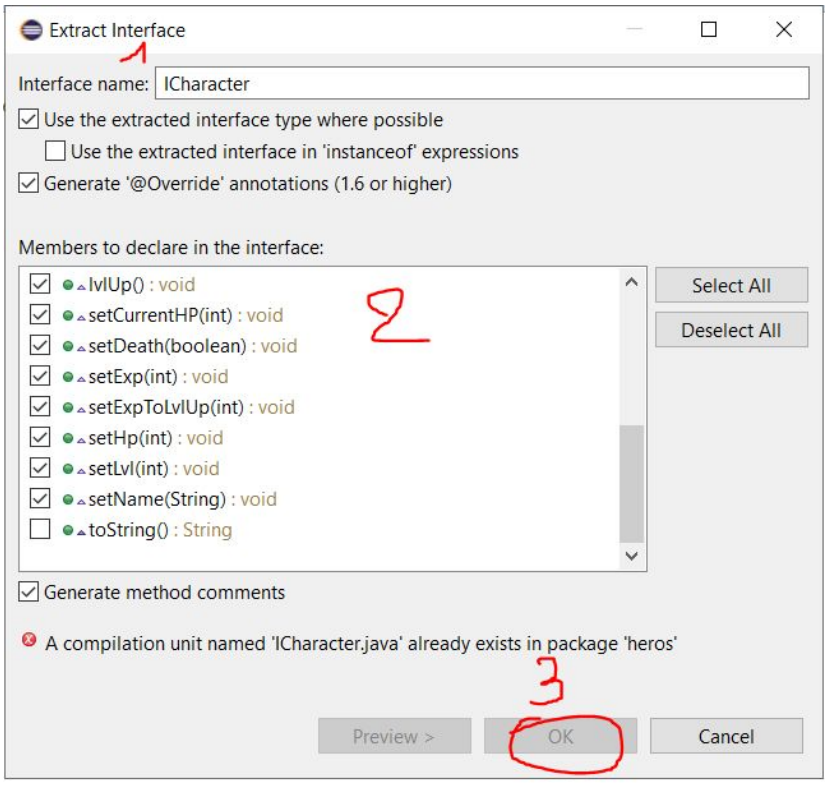
1) Refactoring réalisé

```java
package heros;

public interface ICharacter {
    int getExpToLvlUp();
    void setExpToLvlUp(int expToLvlUp);
    String getName();
    void setName(String name);
    int getCurrentHP();
    void setCurrentHP(int currentHP);
    int getHp();
    void setHp(int hp);
    int getLvl();
    void setLvl(int lvl);
    int getExp();
    void setExp(int exp);
    boolean isDeath();
    void setDeath(boolean isDeath);
    void info();
    void lvlUp();
    boolean isLvlUp();
    void attack(ICharacter character);

}
```

Comme je constate dans le schéma UML de l'application, il y a des opérations en commun que je peux mettre dans une interface afin de définir le comportement des héros. J'ai mis tout d'abord toutes les méthodes accesseurs de la classe abstraite qui contiennent les déjà les codes duplicants des sous-classes dans l'interface.

2) Procédure appliquée sous eclipse





Une fois appuyé sur le bouton ok, nous allons obtenir une classe interface comme vu dans la partie 1).

3) Intérêt

La fonctionnalité “refactor” de code permet de retravailler le code source sans ajouter les nouvelles fonctionnalités ou les erreurs. Ici, nous avons utilisé la fonctionnalité “extract interface” sur la classe abstraite. Celle-ci, nous a permis de définir le comportement de la classe character sans y toucher le code source car cela peut causer une dysfonctionnement de l’application. La maintenabilité de l’application augmente aussi grâce à l’ajout d’une interface. De plus, si nous regardons dans la classe Main, nous voyons que les types statiques de Character ont été changés en type interface ICharacter. On constate que la fonctionnalité “extract interface” gère aussi les déclarations liés à cette interface.

```
ICharacter fighter = new Fighter("Un fighter");
ICharacter mage = new Mage("Un mage");
```

4) La mise en oeuvre

La mise en oeuvre sous eclipse est bien réalisée et plutôt simple à utiliser.

5) Comparaison

Sur le site “refactoring.com/catalog”, il n’y a pas le type refactoring de type interface. Il est possible que l’auteur considère que “extract interface” contient la même fonctionnalité que “extract super-class”. Cela consiste d’extraire les parties en commun vers une super-classe car dans la 1er édition du livre, il y a toujours un exemple de “extract interface”.