TP03 - Création d'un composant logiciel en Java

Contexte

- SEG - Syndicat des Eaux de Gévaudan

L'eau potable a toujours été l'un des premiers objets de coopération intercommunale. La sécurité de l'alimentation face à une ressource rare, difficile à mobiliser ou de mauvaise qualité, a poussé les municipalités à regrouper leurs moyens pour obtenir une distribution de qualité. Le **Syndicat des Eaux de Gévaudan (SEG)** s'est ainsi donné pour mission le captage, le traitement et la distribution de l'eau potable pour satisfaire les usagers répartis sur le territoire des communes regroupées au sein d'un syndicat de communes.

- Gestion de la consommation d'eau

L'eau est un bien de consommation courante et son utilisation doit être contrôlée pour éviter tout gaspillage. La communauté de communes a donc décidé de mettre en œuvre un projet permettant une gestion dynamique et économe de cette ressource, afin d'en limiter les pertes, qui s'expliquent par plusieurs facteurs : débordement des réservoirs, volumes détournés, utilisations non comptabilisées (réseau incendie, eaux de lavage du domaine public, purges des réseaux après travaux, etc.), fuites dues à l'état des canalisations, etc.

Le service des eaux doit donc disposer de moyens pour connaître, puis localiser les origines des pertes. La sectorisation des réseaux est un outil d'aide à la gestion optimale des pertes. Chaque commune est donc divisée en secteurs géographiques (quartiers) qui sont alimentés en eau par des vannes reliées au réseau général. Ces vannes, équipées de compteurs, permettent de mesurer le volume d'eau utilisé dans chaque quartier. L'eau est ensuite distribuée aux usagers via le réseau de canalisations du quartier.

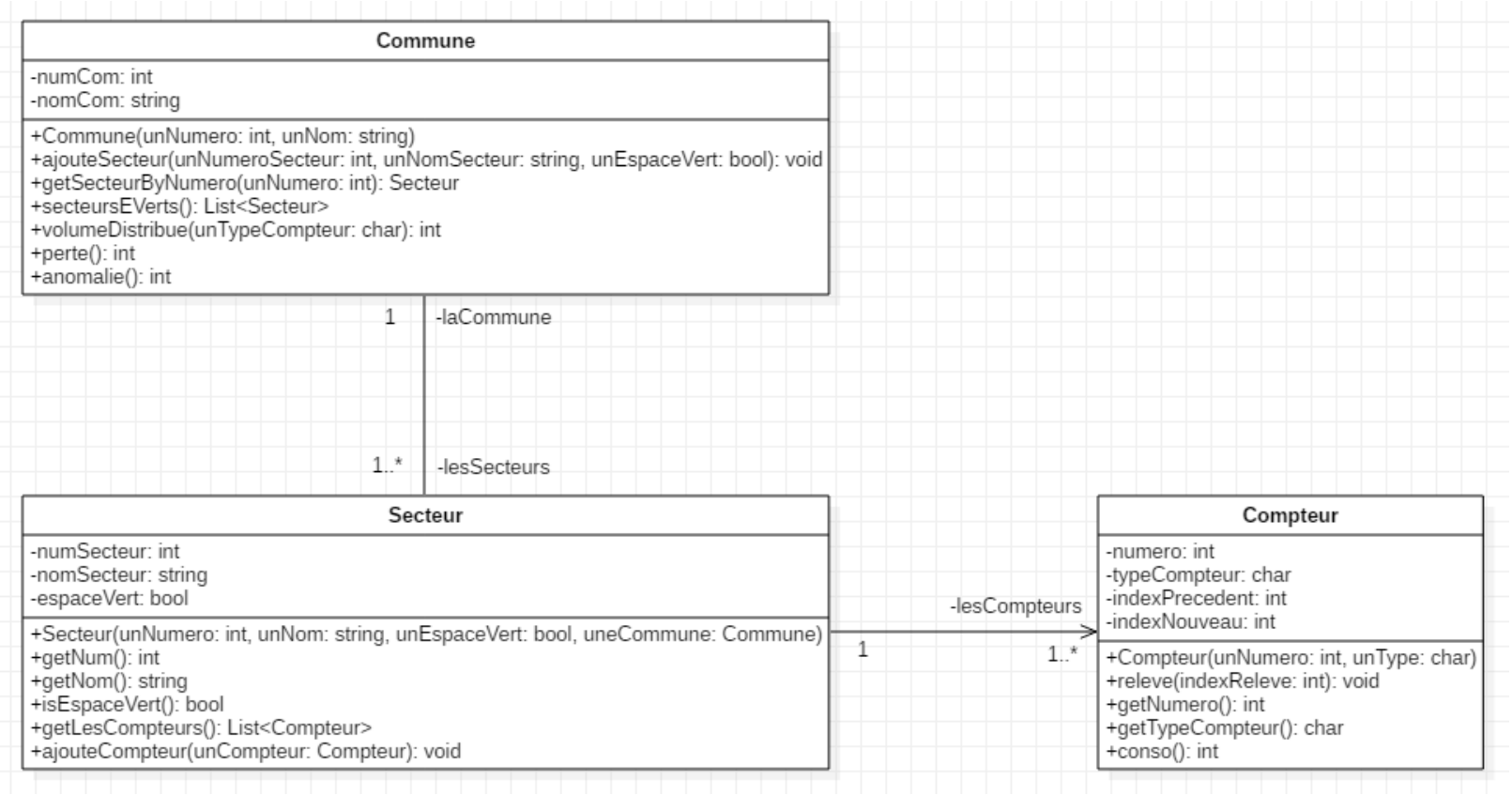
Une fois dans l'année, au mois de mars, les agents communaux relèvent les consommations d'eau de chaque secteur en consultant le compteur des vannes et les compteurs individuels de chaque abonné du quartier. Sur chaque compteur (de vanne ou d'abonné), on relève le nouvel index (en m^3) ; la consommation est calculée en y soustrayant l'index précédent.

Spécifications techniques

Vous intégrez l'équipe chargée du développement d'une application concernant la gestion des relevés, on vous confie le développement du composant logiciel métier en Java. Le diagramme de classes est fourni **(Doc. 1).**

TRAVAIL A FAIRE	
1.	**Créer un nouveau projet Java puis créer un *Package* de nom** `sio.seg.metier` *Pour créer un package ⇒ click droit sur le projet puis menu* `New/Package`
2.	**Dans ce package, créer le prototype des classes correspondant au diagramme de classes métier** (Doc. 1)
3.	**Implémenter les méthodes des différentes classes** *Conventions de codage du langage Java :* *https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html*
4.	**Documenter les classes puis générer la documentation Java** (Doc. 2)
5.	**Générer un fichier JAR de nom** `SEGMetier.jar` (Doc. 3)
6.	**Afin de tester les méthodes implémentées, créer un nouveau projet Java** `TestSEG` **dans lequel vous aurez importé la librairie externe** `SEGMetier.jar` (Doc. 4)

Document 1 - Diagramme de classes



Classe `Commune`

Méthode `ajouteSecteur` : crée le secteur et l'ajoute à la collection `lesSecteurs`

Méthode `getSecteurByNumero` : retourne l'objet `Secteur` dont le numéro est passé en paramètre ou `null` s'il n'existe aucun secteur portant ce numéro

Méthode `secteursEVerts` : retourne les secteurs possédant un espace vert

Méthode `volumeDistribue` : retourne le volume distribué par les compteurs de la commune pour le type de compteur spécifié (A pour abonné, V pour vanne)

Méthode `perte` : retourne la différence entre le volume total distribué par les vannes et la consommation des abonnés

Méthode `anomalie` : calcule le pourcentage des pertes par rapport au volume distribué par les vannes et retourne : 1 si ce pourcentage est inférieur à 10%
2 s'il est entre 10 et 15% inclus
3 au-dessus de 15%

Classe `Secteur`

Attribut `nomSecteur` : nom du quartier

Attribut `espaceVert` : indique la présence ou non dans le quartier d'un espace vert municipal à arroser

Classe `Compteur`

Attribut `typeCompteur` : A pour abonné ou V pour vanne

Attributs `indexPrecedent` et `indexNouveau` : index avant le relevé et après le relevé

Méthode `releve` : met à jour les index. L'index relevé sur le compteur est passé en paramètre

Méthode `conso` : calcule et retourne la consommation en m^3 pour le compteur

Document 2 - Génération documentation Java sous Eclipse

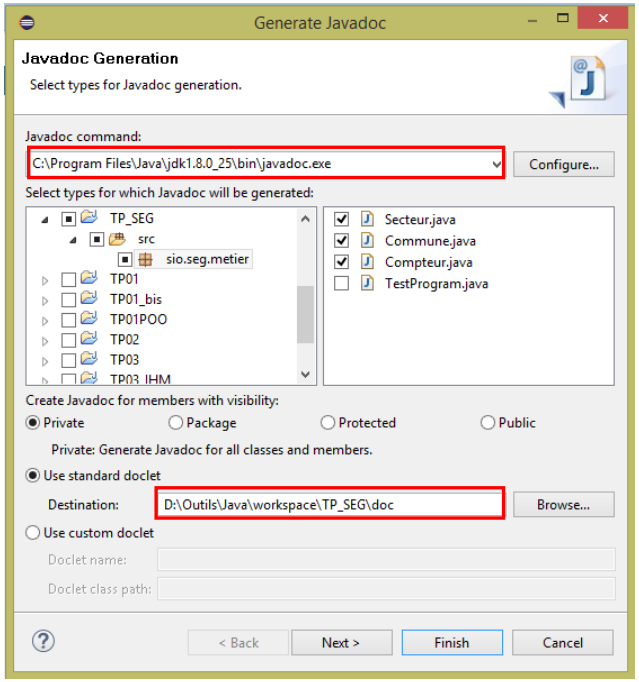
- *Générer les tags de documentation*

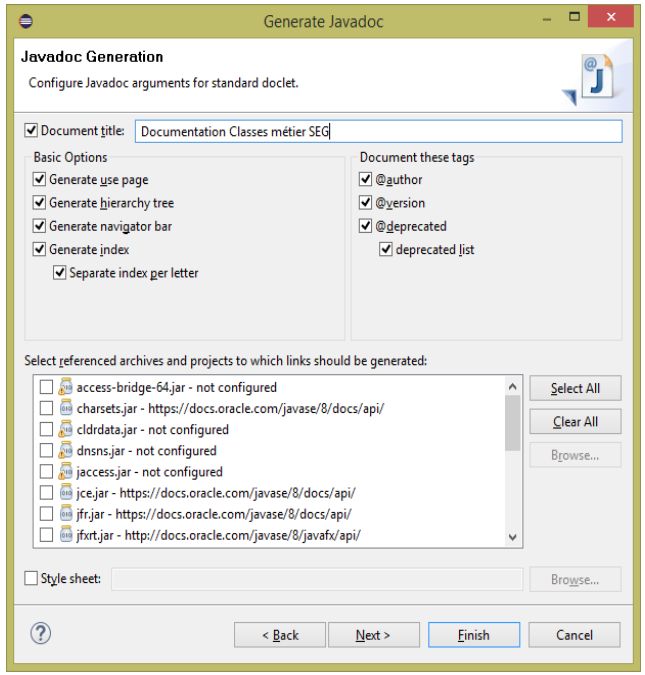
Pour générer les tags de documentation
⇒ taper les caractères ***/***** *puis touche entrée*

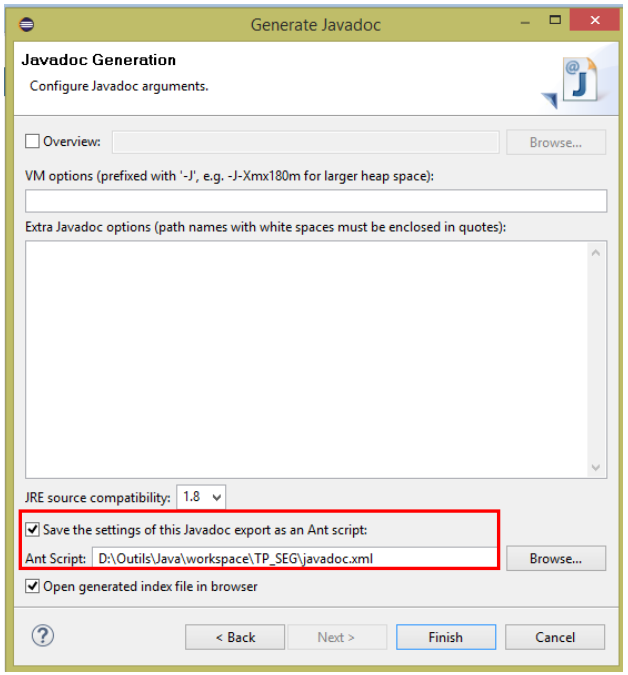
```
/**
* Constructeur de la classe Compteur
* @param numero      numéro du compteur
* @param typeCompteur      type du compteur
*/
public Compteur(int numero, char typeCompteur) {...}
```

- *Générer la documentation Java*

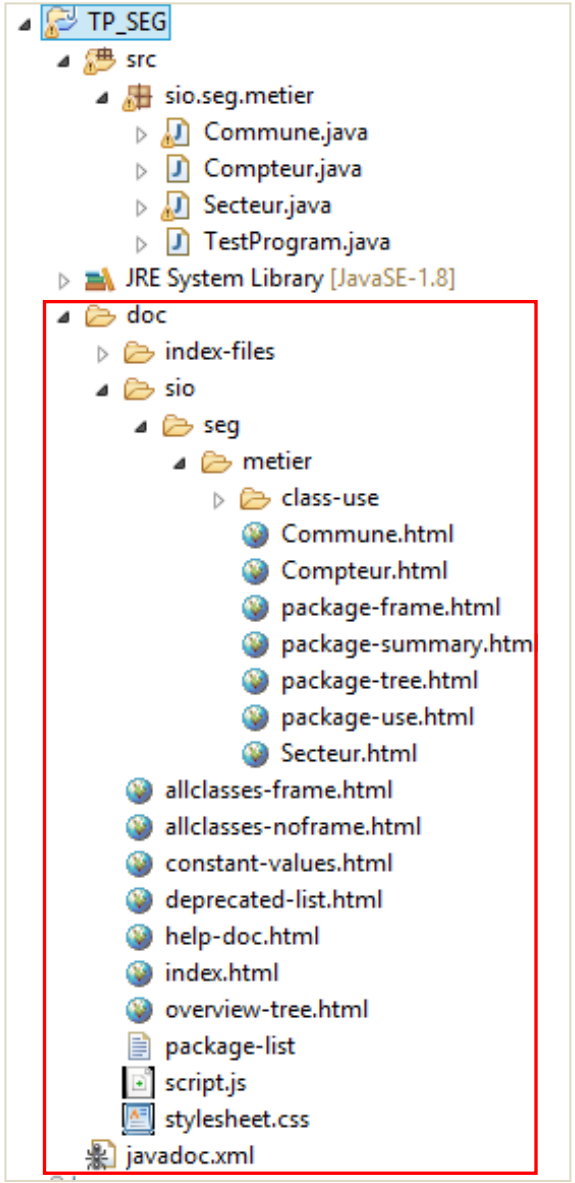
Pour générer la documentation Java ⇒Menu `Project / Generate Javadoc`

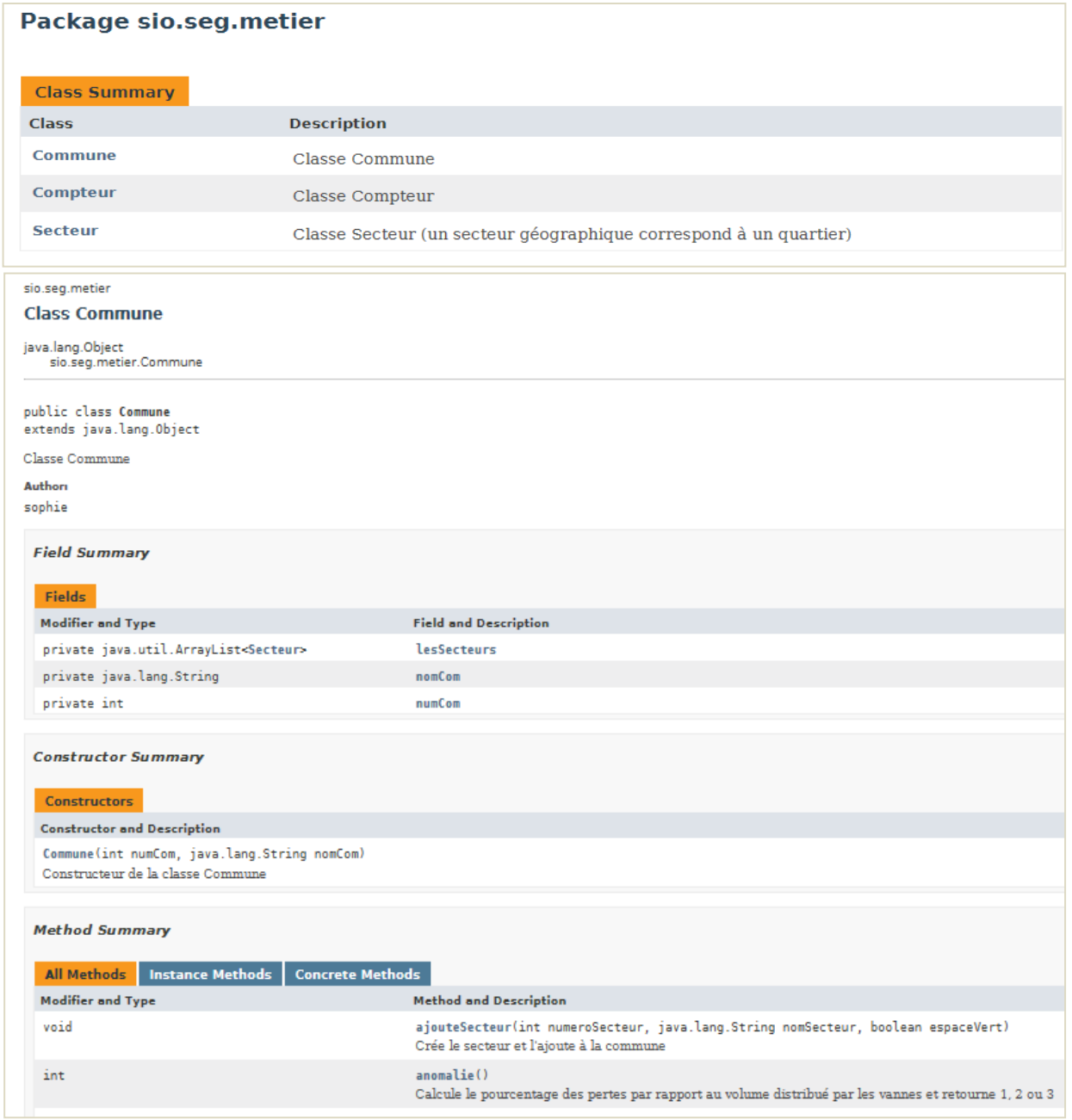






La documentation Java a bien été générée :





Package sio.seg.metier

Class Summary

Class	Description
Commune	Classe Commune
Compteur	Classe Compteur
Secteur	Classe Secteur (un secteur géographique correspond à un quartier)

sio.seg.metier

Class Commune

java.lang.Object
  sio.seg.metier.Commune

```
public class Commune
extends java.lang.Object
```

Classe Commune

Author
sophie

Field Summary

Fields

Modifier and Type	Field and Description
private java.util.ArrayList<Secteur>	lesSecteurs
private java.lang.String	nomCom
private int	numCom

Constructor Summary

Constructors

Constructor and Description
Commune(int numCom, java.lang.String nomCom) Constructeur de la classe Commune

Method Summary

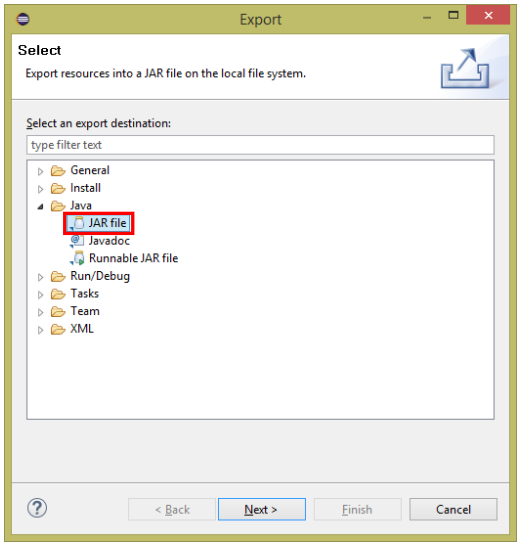
All Methods | **Instance Methods** | **Concrete Methods**

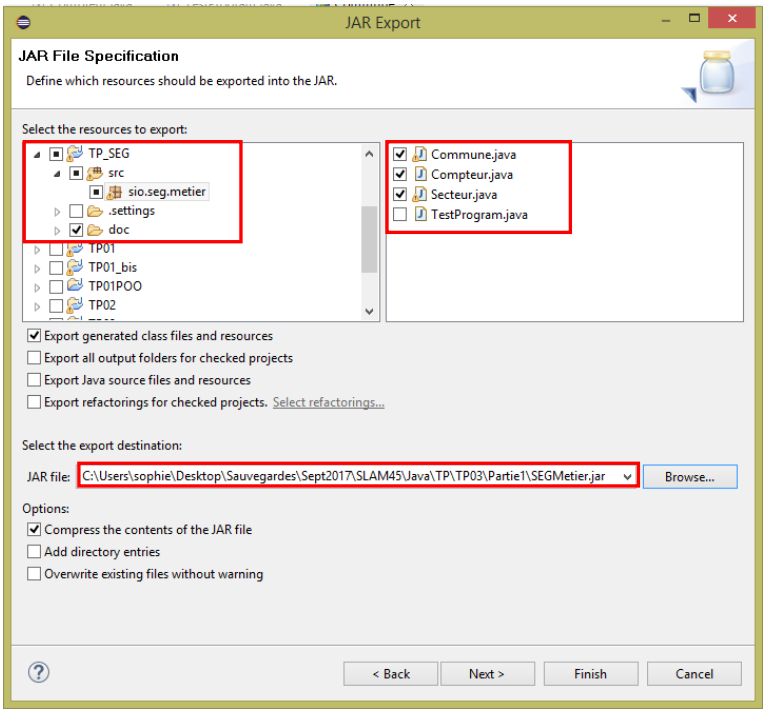
Modifier and Type	Method and Description
void	ajouteSecteur(int numeroSecteur, java.lang.String nomSecteur, boolean espaceVert) Crée le secteur et l'ajoute à la commune
int	anomalie() Calcule le pourcentage des pertes par rapport au volume distribué par les vannes et retourne 1, 2 ou 3

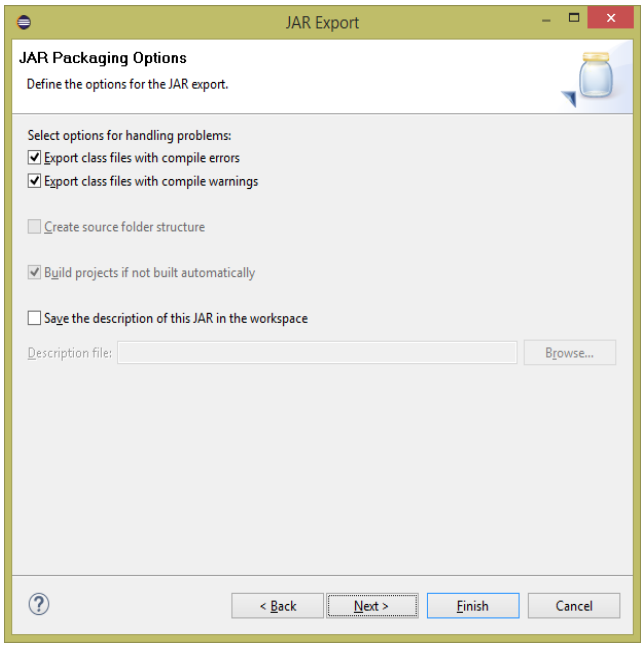
Document 3 - Génération d'un fichier JAR

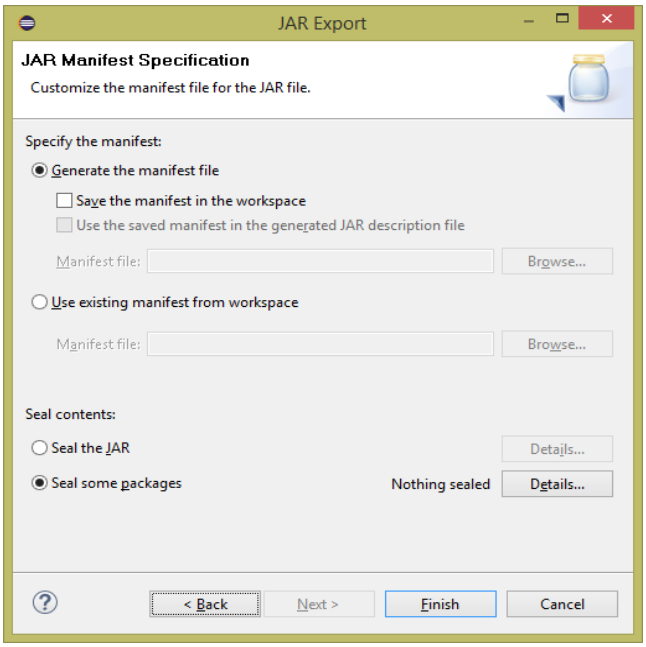
*Un fichier **JAR** (Java archive) est un fichier compressé utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker les définitions des classes ainsi que des métadonnées (fichier manifeste)*

Pour générer un fichier Jar ⇒ click droit sur le projet puis Menu `Export`







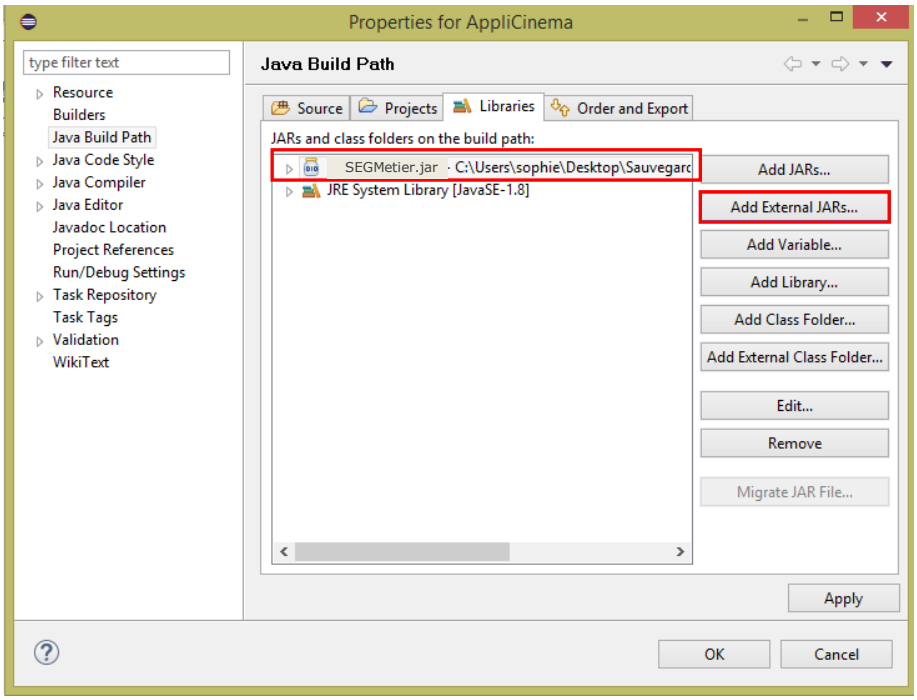


Le fichier *Jar* a bien été généré, il peut maintenant être utilisé dans un projet.

Document 4 - Ajout d'une librairie externe (fichier .jar)

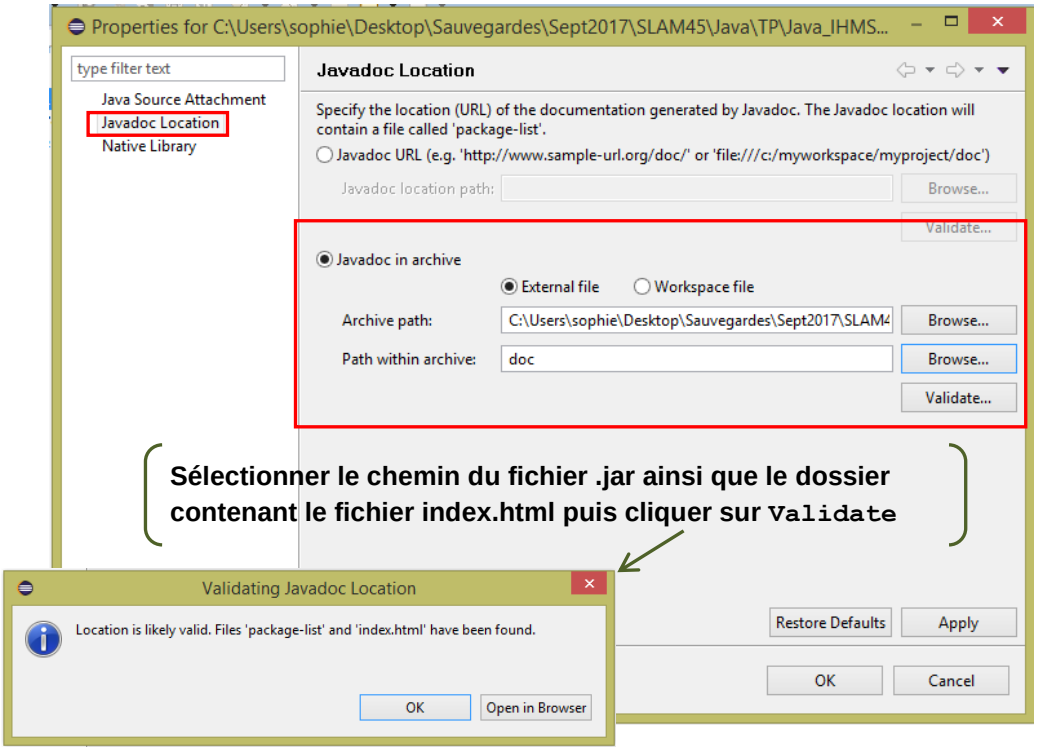
2 façons pour y parvenir :

- *Lors de la création du projet, onglet Librairies / Add External JARs*
- *Click droit sur le projet puis Properties/Java Build Path / Add External Archives*



La librairie a bien été ajoutée à notre projet, il faut maintenant **associer la documentation JavaDoc**

Dans l'explorateur, faire un click droit sur le fichier SEGMetier.jar / Properties et sélectionner Javadoc Location



Afin d'utiliser dans notre code les classes contenues dans cette librairie, il faut inclure le package :

```
import sio.seg.metier.*;
```