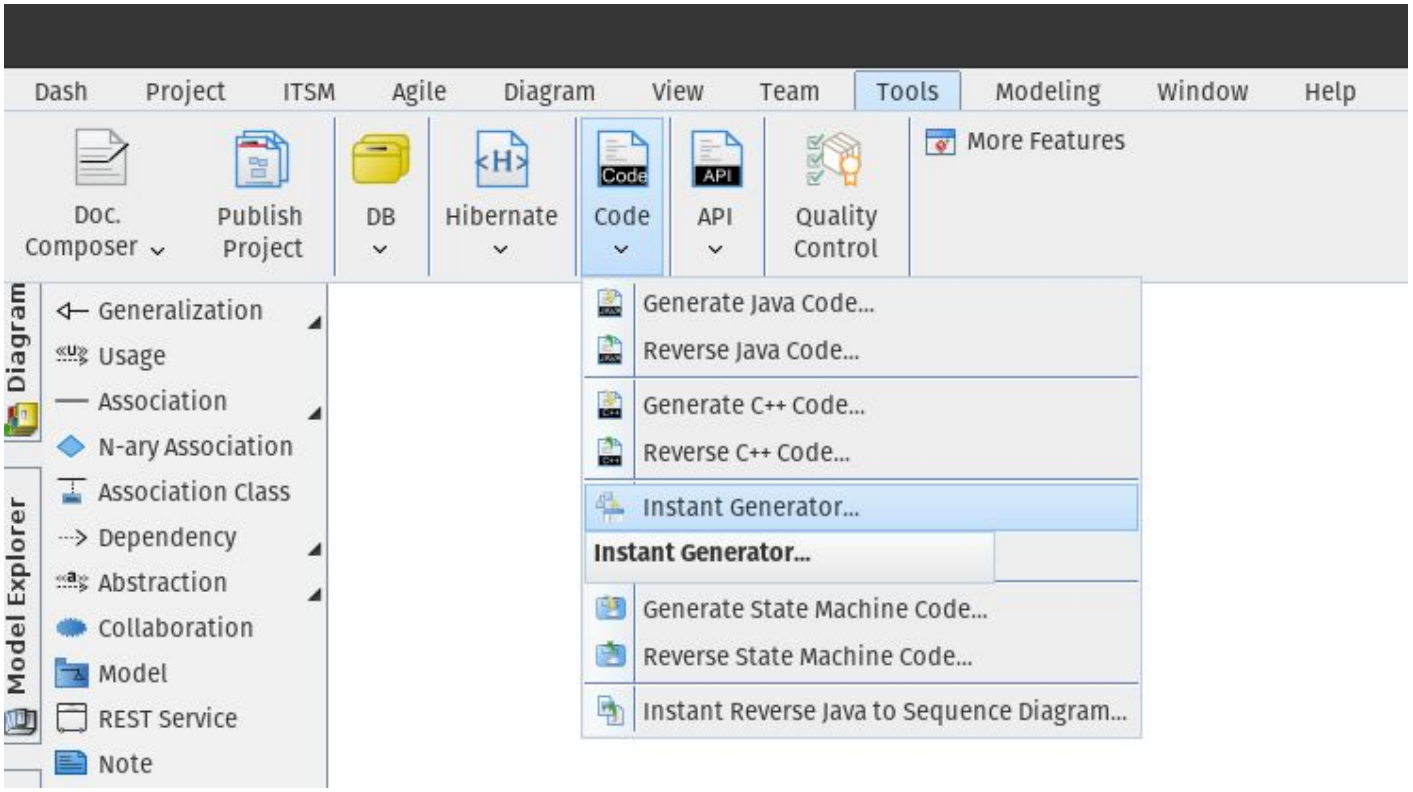
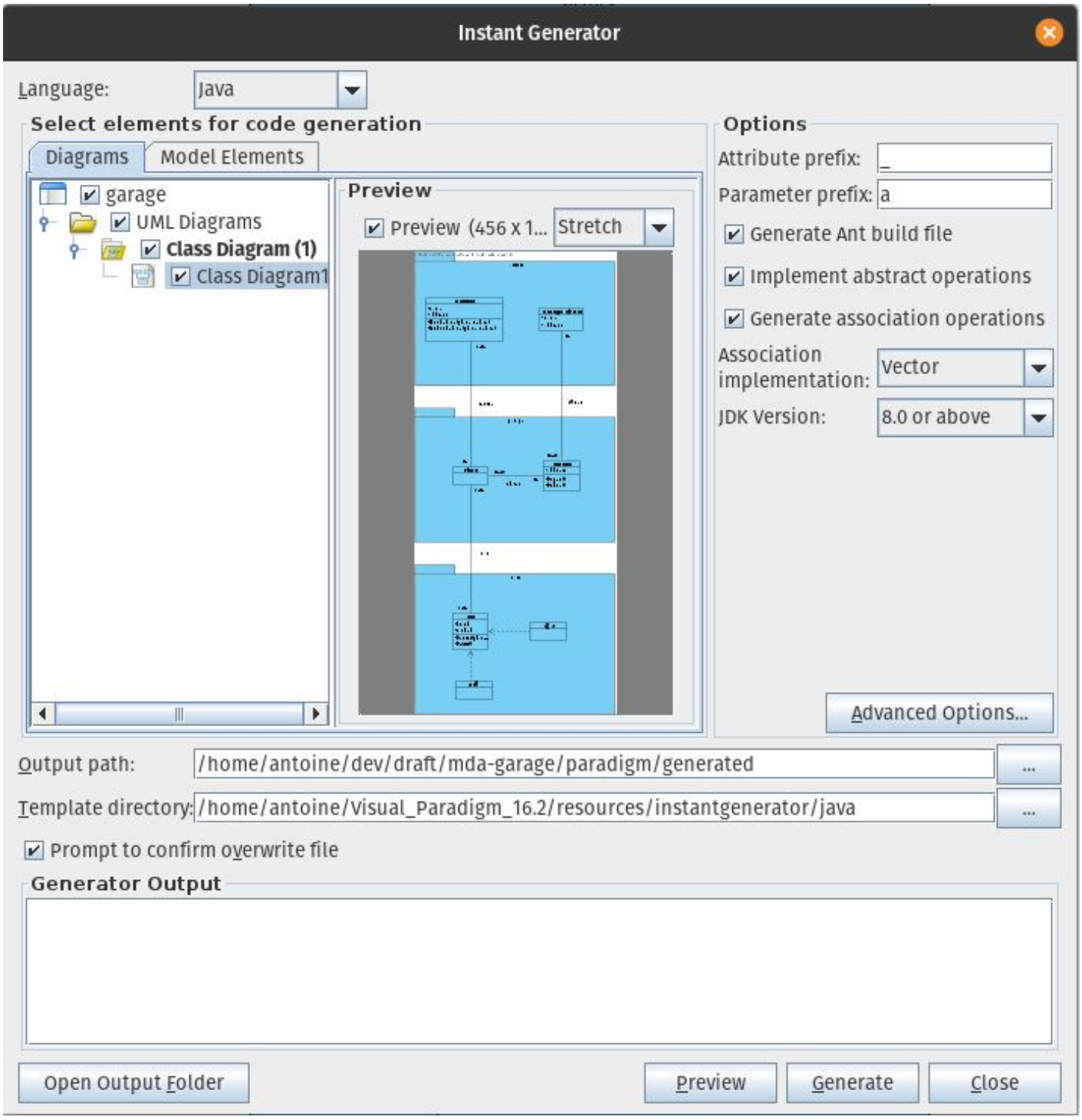
Génération de code java à partir d'un diagramme de classe

Pour générer le code, il suffit de se rendre dans tools > Code > Instant Generator



Dans la fenêtre qui s'ouvre, on sélectionne les diagrammes à partir desquels générer le code, et on choisit un dossier de sortie.

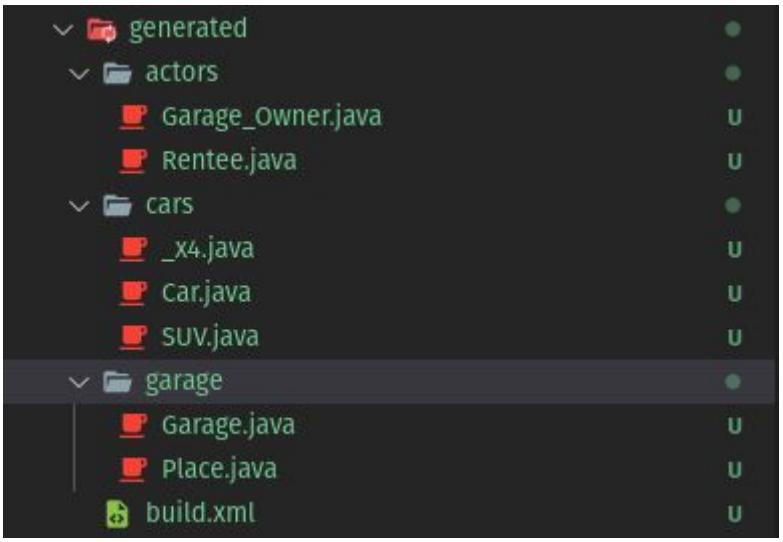


Cette fenêtre permet aussi de régler pas mal de paramètres pour la génération du code. On a le choix entre beaucoup de langages on peut également choisir la classe à utiliser pour les associations. Par défaut, il s'agit de la classe Vector qui est un peu comme une liste java.

On peut également choisir le préfixe des attributs et la version de JDK à utiliser.

Ici on laisse tout par défaut et on clique sur *Generate*.

Voici les fichiers générés:



On retrouve bien nos packages avec tous nos modèles à l'intérieur.
Il y a juste eu un petit soucis avec la classe 4x4 qui s'est transformée en _X4. Rien de très grave.

À l'intérieur de nos classes, on retrouve bien tous les attributs et les méthodes que l'on avait définis dans le diagramme de classe. Par exemple, voici la classe Garage:

paradigm > generated > garage > Garage.java > Garage

```java
package garage;

import actors.Garage_Owner;
import java.util.Vector;
import garage.Place;

public class Garage {
    private Object _address;
    public Garage_Owner _owner;
    public Vector<Place> _places = new Vector<Place>();

    public void open() {
        throw new UnsupportedOperationException();
    }

    public void close() {
        throw new UnsupportedOperationException();
    }
}
```

Les méthodes sont prêtes à être implémentées, mais pour l'instant elles renvoient une *UnsupportedOperationException* pour dire qu'elles ne sont pas encore implémentées.
On remarque également l'utilisation des prefix d'attribut _ et de la classe *Vector* comme configuré plus haut.

Pour conclure, on peut dire que la fonction pour générer du code depuis des diagrammes Visual Paradigm est extrêmement pratique.
Elle permet d'être sûr que notre code correspond à 100% à notre UML. Il faut ensuite prendre soin de mettre à jour les deux en même temps lors d'un changement dans le code pour rester cohérent.
C'est également un gros gain de temps d'utiliser cette fonction, et en plus cela encourage le développeur à créer des diagrammes avant de commencer un projet.
C'est une très bonne découverte et je vais essayer d'utiliser cette fonctionnalité plus souvent, surtout qu'elle est disponible sur pas mal de langages.