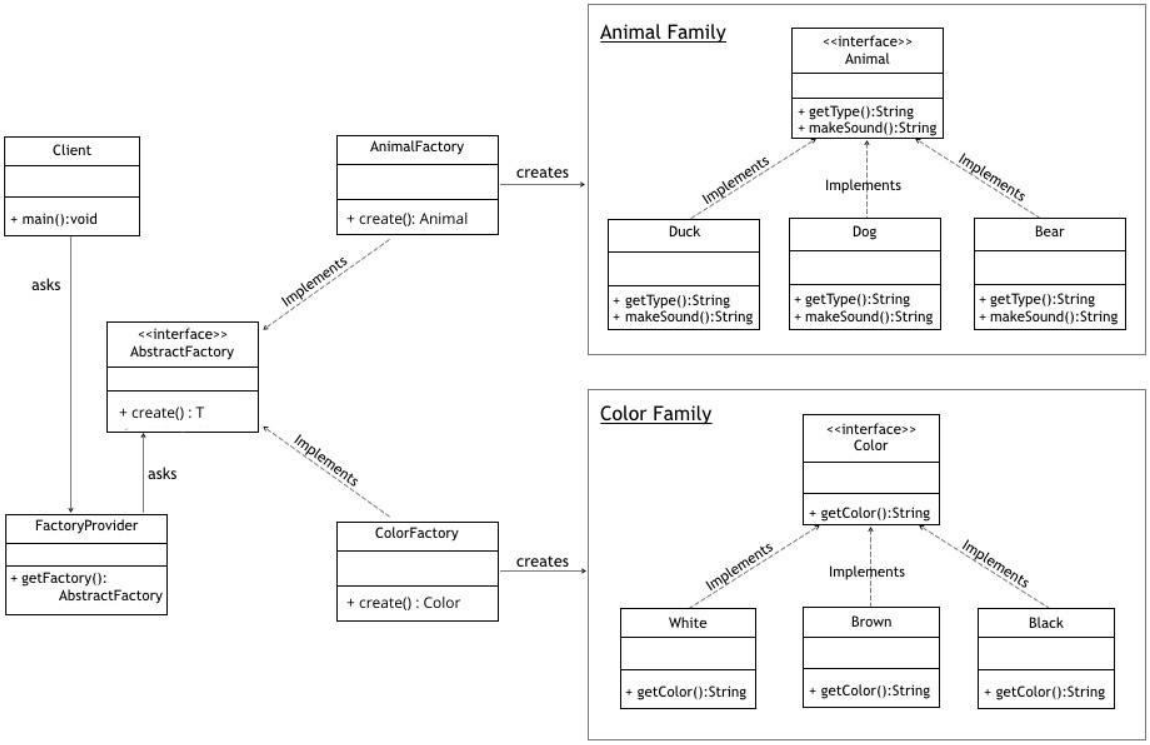
Abstract Factory

Exemple de modèle de conception d'usine abstraite

Dans cet exemple, nous allons créer deux implémentations du modèle Factory Method Design : *AnimalFactory* et *Color Factory*.

Après cela, nous en gérerons l'accès à l'aide d'une Abstract Factory *AbstractFactory*



Tout d'abord, nous allons créer une famille de classe *Animal* et l'utiliserons plus tard dans notre Abstract Factory.

Voici l'interface *Animal :*

```
Public interface Animal {
      String getAnimal();
      String makeSound();
 }
```

Et une implémentation concrète *Duck* :

```
Public class Duck implements Animal {
    @Override
    Public String getAnimal() {
            return "Duck";
    }
     @Override
      Public String makeSound () {
            Return "Squeks";
    }
}
```

De plus, nous pouvons créer des implémentations plus concrètes de l'interface *Animal (comme Dog, Bear,* etc.) exactement de cette manière.

L'usine abstraite traite des familles d'objets dépendants. Dans cet esprit, nous allons introduire une autre famille *Color* en tant qu'interface avec quelques implémentations (White, *Brown,*).

```
Public interface AbstractFactory<T> {
     T create (String animalType) ;
}
```

Ensuite, nous allons implémenter une *AnimalFactory* en utilisant le modèle de conception Factory Method dont nous avons parlé dans la section précédente :

```
Public class AnimalFactory implements AbstractFactory<Animal> {
        @Override
         Public Animal create (String animalType) {
               If ("Dog".equalsIgnoreCase(animalType)) {
                     return new Dog();
                } else if ("Duck".equalsIgnoreCase(animalType)) {
                     return new Duck();
                } return null;
          }
}
```