

## EXAMEN FINAL

Avril/Juin 2020

**BRANCHE : ALGORITHMIQUE ET  
PROGRAMMATION OBJET (APO)**

**CLASSE : ESIG2 Classe A  
(Groupes 1 et 2)**

**DATE : 10 juin 2020**  
initialement programmé le 1<sup>er</sup> avril 2020

**NOM : .....**

**PRÉNOM : .....**

**PROFESSEUR : Eric Batard**

**N° du poste de travail : ESIG-PB.....**

**N° de clé USB : .....**

## Modalités

- Durée : 240 minutes.
- Travail individuel.
- Documentation personnelle (livres, papiers) : Autorisée.  
Documentation électronique (disquettes, CD, clé USB, ...) : Autorisée si elle a été recopiée dans un répertoire sur **C:\ESIGUsers** *avant* le début de l'épreuve.
- Tout partage de ressources de votre poste de travail avec le réseau ainsi que toute tentative de communication seront considérés comme fraude et sanctionnés comme tels par la note minimale.

## Démarrage

- Connectez-vous au réseau sur le poste de travail qui vous a été attribué.
- Copiez dans **C:\ESIGUsers\** le dossier réseau qui vous sera indiqué au début de l'épreuve, normalement  
**G:\ESIG\Distribution\2019\_2020\ESIG-2\APO\Eléments Exa APO 10-06-20**
- Les éléments fournis dans ce répertoire sont l'énoncé en .pdf et un répertoire de projet Exa20 pour IntelliJ IDEA. C'est ce répertoire de projet qu'il faut ouvrir, après copie, dans IntelliJ IDEA, par exemple par glisser-déposer.

## Travail à faire

- Lisez tous les documents fournis.
- Complétez les procédures/fonctions/méthodes ou classes demandées de manière à ce qu'elles répondent aux spécifications de l'énoncé.
- Vous rendrez les fichiers correspondants sur la clé USB qui vous sera remise quand vous serez près à rendre.
- Informations complémentaires sur l'évaluation : l'affichage de résultats dans le log quand un affichage dans une fenêtre est demandé ne rapportera au mieux qu'une fraction des points de même que l'utilisation d'une liste quand le parcours d'un arbre est demandé.

C'est à vous de vérifier que les fichiers enregistrés et rendus contiennent bien la dernière version de votre travail

**Le nom du répertoire de projet doit être préfixé par vos initiales**

-----

**A ajouter juste avant la reddition**

Quand les deux cas se présentent, le masculin est utilisé dans cet énoncé de façon générique.

# Classification À PO issons

## Contexte

Un usage déjà ancien des structures arborescentes est la classification du vivant, des multiples espèces de la faune et de la flore. Même la classification des poissons est complexe (une illustration est insérée mais à la fin de cet énoncé pour ne pas vous effrayer).

Nous allons nous limiter aux poissons plutôt courants du Lac Léman, complétés par une famille supplémentaire (que vous pouvez deviner mais c'est hors-sujet).

L'objectif est de construire une interface permettant d'illustrer une partie de la classification des poissons et d'afficher des informations selon les demandes de l'utilisateur.

A noter que la construction de l'interface se fera en deux temps.

## Première étape : construire l'interface de base (dans la classe `FenExa20`)

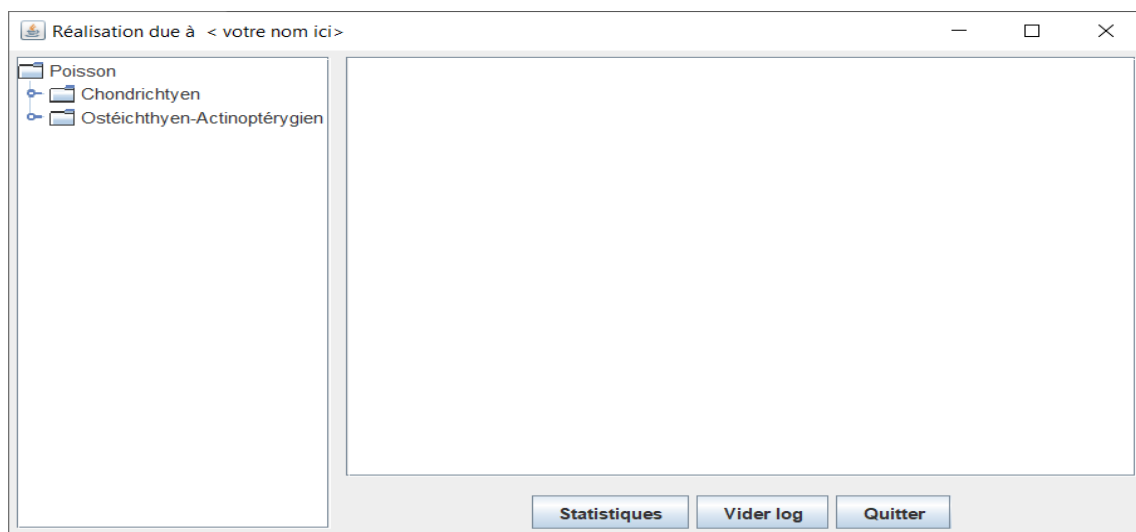


Figure 1 – le « look » initial (un peu aplati) après affichage

Comment obtenir cette première version de l'interface ?

On constate qu'il y a deux zones côte à côte :

- À gauche, un `JTree`. On verra plus loin comment obtenir les données qu'il affiche.
- À droite, une zone constituée d'un `JTextArea` (de 20 lignes sur 50 colonnes et non modifiable interactivement) et d'une sous-zone avec 3 boutons.

De plus, le `JTree` et le `JTextArea` doivent être dotées de barres de défilement apparaissant quand c'est nécessaire.

Pour remplir le `JTree`, il faudra utiliser la méthode/fonction fournie `initRacine()` qui renvoie une instance de la classe `DefaultMutableTreeNode`. Cette instance correspond à la racine de l'arbre que doit afficher le `JTree`.

Pour votre information, la méthode `initRacine()` trouve ses informations dans l'arborescence de répertoires `Poisson` qui est fournie dans le répertoire de projet. Il ne faut évidemment ni la modifier, ni la supprimer !

Il faudra bien sûr qu'un clic sur la case de fermeture ferme proprement votre programme (= arrêt du processus).

## Deuxième étape : afficher l'interface (dans la classe `RunExa20`)

Vous ajouterez les instructions permettant l'affichage d'une instance de la classe de fenêtre `FenExa20`. Et n'oubliez pas d'indiquer votre prénom et votre nom dans le paramètre du constructeur, et d'afficher ce titre.

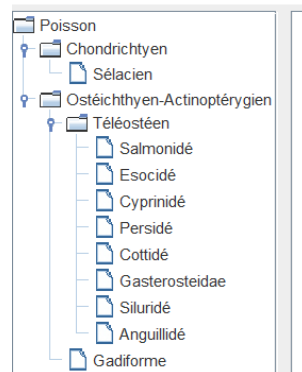


Figure 2 – le « look » initial quand l'arbre est totalement ouvert

## Troisième étape : compléter l'interface de base (dans la classe `FenExa20`)

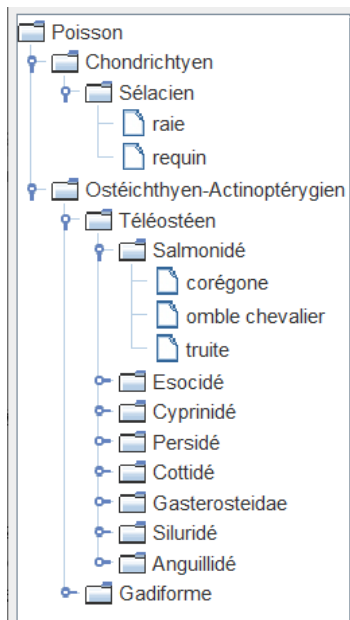


Figure 3 – le « look » final  
avec quelques exemples dans deux familles (Sélacien et Salmonidé)

Si vous avez eu la curiosité de regarder dans l'arborescence de répertoires fournie, vous avez constaté que seuls les répertoires sont repris dans l'arborescence affichée actuellement par le `JTree`. Il manque les fichiers (tous des `.txt`). Il faut savoir que le dernier répertoire, celui qui contient des fichiers, correspond à ce qu'on appelle en classification du vivant une *famille*.

Vous avez peut-être aussi remarqué qu'une `HashMap` appelée `exemples` était fournie. Pour l'initialiser, il est indispensable que vous appelez `initExemples()` dans le constructeur de la fenêtre. Qu'y a-t-il dans `exemples`? Pour chaque famille, on associe une liste d'exemples (c'est-à-dire de poissons).

L'objectif de cette étape est d'utiliser les informations dans `exemples` pour compléter l'arbre en mémoire affichée par le `JTree`. Il faut les ajouter et cela, *avant* l'affichage du `JTree` (NB : il n'est pas impossible de le faire après coup, ce serait donc accepté mais c'est sensiblement plus technique !)

Pour compléter l'arbre en mémoire, il faut procéder ainsi :

- Pour chaque famille, c'est-à-dire pour chaque clé dans la `HashMap` `exemples`, il faut trouver le nœud correspondant. La recherche du nœud correspondant à une famille est assurée par la méthode fournie, `trouverFamille()`, qui prend en paramètre le nom de la famille et la racine de l'arbre en mémoire.
- Une fois obtenu le nœud qui porte le nom de la famille, grâce à `trouverFamille()`, il faut ajouter comme fils de ce nœud un nouveau nœud correspondant à chacun des exemples dans la liste d'exemples associée à la famille (et qu'on trouve dans la `HashMap` `exemples`).

C'est à vous de faire cette partie b.

## ***Quatrième étape : rendre opérationnels les boutons***

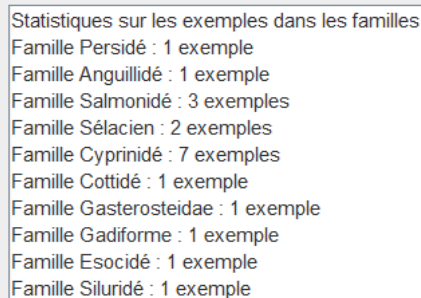
### ***Etape 4.1 – Vider le log***

Comme d’habitude on appelle *log*, le `JTextArea` où on affichera divers résultats. Vider le log revient à effacer tout ce qui est déjà affiché.

### ***Etape 4.2 – Stat***

Il s’agit simplement d’afficher dans le log un récapitulatif des nombres d’exemples pour chacune des familles. Le plus simple est d’utiliser la `HashMap` `exemples` mais toute méthode de *calcul* (donc pas une simple recopie de résultats spécifiques) sera acceptée.

Il y a une méthode fournie `afficheDansLog()` qui affiche dans le `JTextArea` donné en première paramètre le message donné en second paramètre sur une ligne à part.



```
Statistiques sur les exemples dans les familles
Famille Persidé : 1 exemple
Famille Anguillidé : 1 exemple
Famille Salmonidé : 3 exemples
Famille Sélacien : 2 exemples
Famille Cyprinidé : 7 exemples
Famille Cottidé : 1 exemple
Famille Gasterosteidae : 1 exemple
Famille Gadiforme : 1 exemple
Famille Esocidé : 1 exemple
Famille Siluridé : 1 exemple
```

Figure 4 – affichage des statistiques dans le log

### ***Etape 4.3 – Quitter***

A la suite d’un clic sur ce bouton (ou de la case de fermeture comme déjà indiqué), l’application se termine proprement, c’est-à-dire sans processus restant en mémoire.

## ***Cinquième étape : récupérer les données d’informations***

Les données d’informations sont les données qui seront affichées quand l’utilisateur cliquera sur le `JTree` (cf. l’étape suivante pour la mise en œuvre).

Ces données se trouvent dans le fichier texte `infos.txt` fourni (dans la racine du répertoire de projet et listé en Annexe). Sa structure est simple : chaque ligne contient deux chaînes de caractères séparées par un point-virgule (format csv européen).

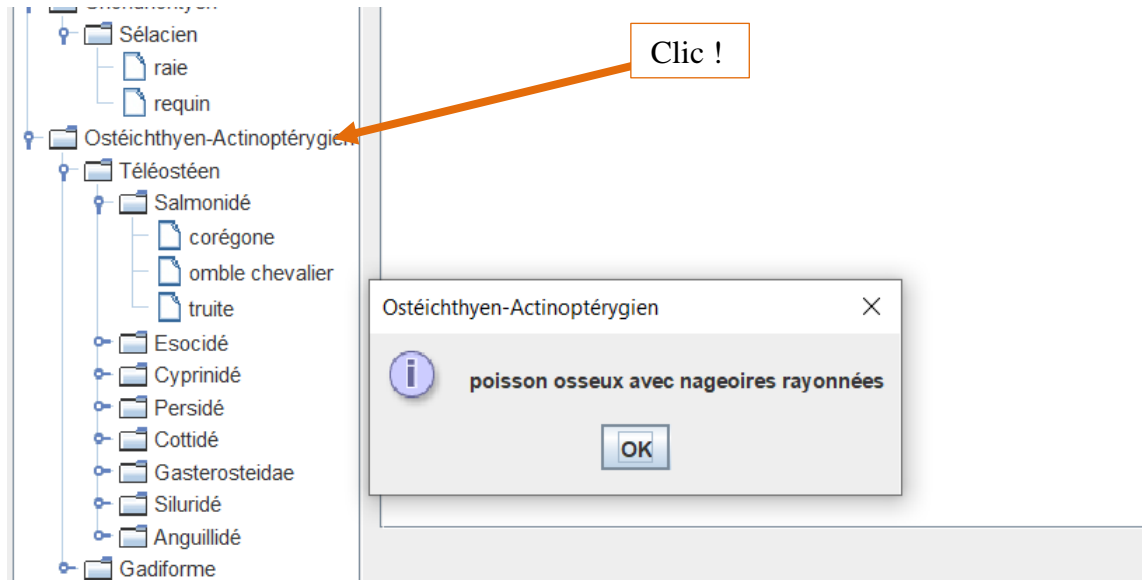
Il y a deux questions à se poser :

1. Comment stocker ces informations en mémoire ? Le plus efficace est une nouvelle `HashMap` car la première chaîne de chaque ligne correspond au texte d’un nœud affiché sur le `JTree` mais toute méthode est acceptée du moment qu’elle dépende des données.
2. Comment découper la ligne lue en deux chaînes ? Là encore il y a plusieurs approches, toujours toutes acceptées du moment qu’elles dépendent des données. On peut utiliser un deuxième `Scanner` avec `useDelimiter(";")`. On peut combiner deux méthodes de la classe `String` `substring()` et `indexOf(";")`, cette dernière méthode donnant la position du point-virgule. Dans ma solution, j’ai utilisé une autre méthode de la classe `String`, `split(";")`, qui renvoie sous la forme d’un tableau de chaînes, ici donc un tableau à deux éléments, les morceaux d’une chaîne séparés par le point-virgule.

### ***Sixième étape : rendre le JTree réactif***

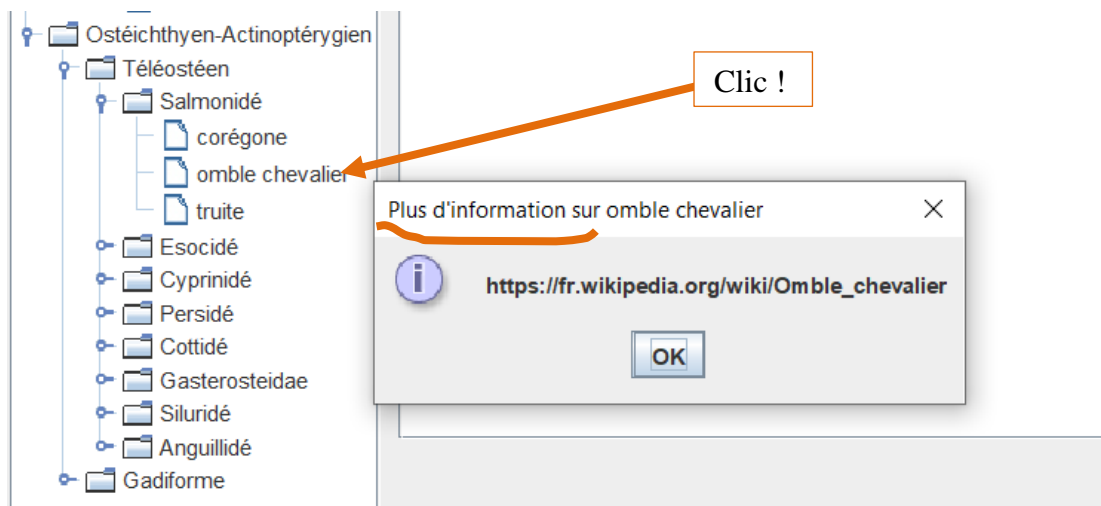
Il s'agit de présenter à l'utilisateur certaines informations quand il clique sur le JTree. Il faut déjà gérer l'événement correspondant (n'oubliez aucun des 3 piliers !).

Si on clique sur un nœud dont le texte apparaît dans les informations lues dans l'étape précédente, on se contente d'afficher une boîte de dialogue où le titre correspond au nœud et le texte du message au texte associé. Ci-dessous un exemple et tous les cas différents sont listés en Annexe.



*Figure 5 – exemple de clic sur Ostéichthyen-Actinoptérygien*

Une amélioration supplémentaire : si on clique sur un poisson-exemple, c'est-à-dire une feuille de l'arborescence affichée, on aimerait compléter l'affichage de l'URL avec un petit message dans le titre comme ceci :



*Figure 6 – exemple de clic sur omble chevalier*

Au cas où il n'y a pas d'information associée, le nœud cliqué correspond alors à une famille, et on veut voir la liste de tous les exemples dans le log présenté comme ceci :

Voici des exemples de poissons dans la famille Cyprinidé

- tanche
- chevaine
- ablette
- vengeron (gardon)
- brème
- carpe
- goujon

Figure 7 – Log après clic sur Cyprinidé

Notez qu'une méthode est fournie, `afficheDansLog()` qui affiche dans le `JTextArea` donné en première paramètre le message donné en second paramètre sur une ligne à part.

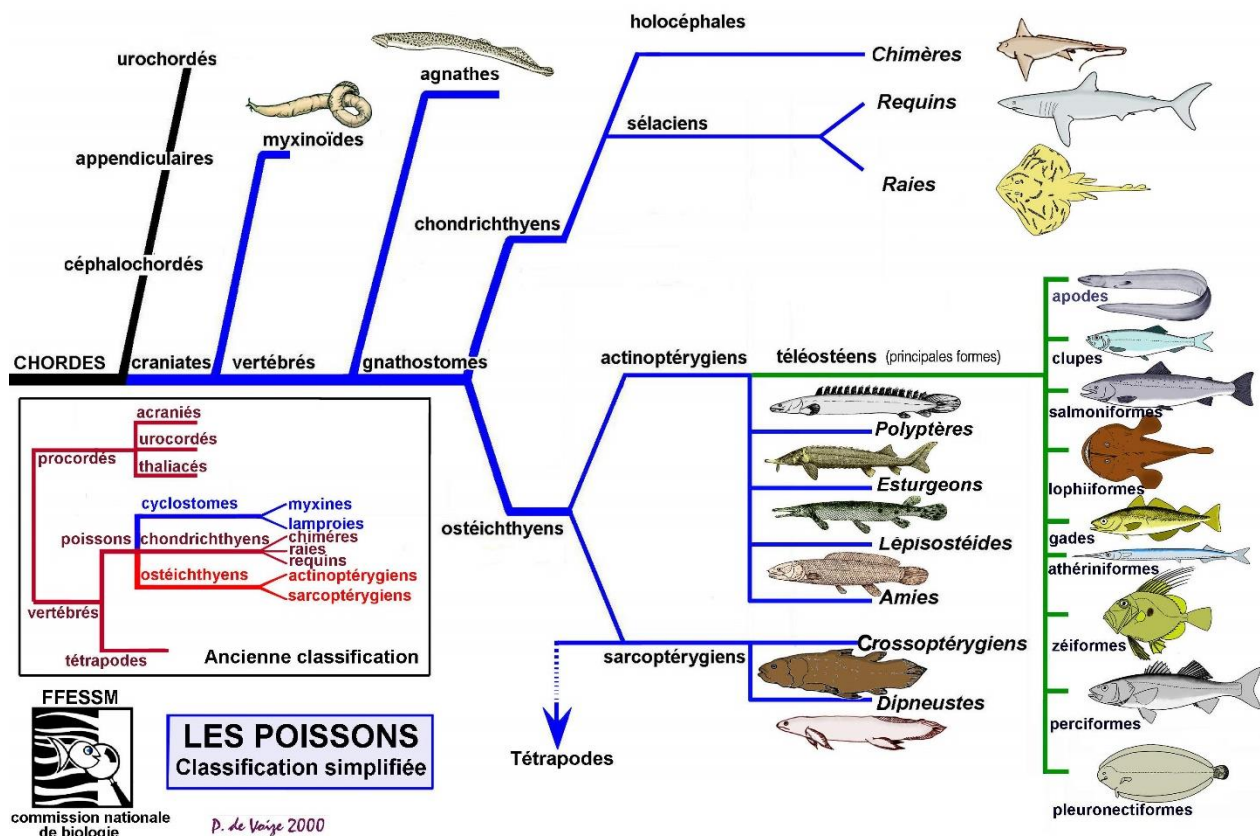
## Indication

Comment savoir si une clé est déjà présente dans une `HashMap` ? On peut utiliser la méthode booléenne `containsKey()` dans `HashMap` qui prend en paramètre la valeur de la clé en question.

## Avertissement

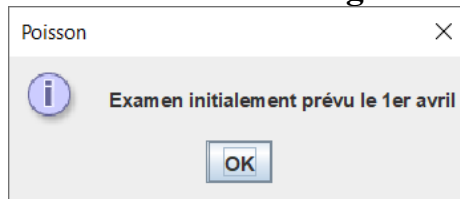
Certains points ont été simplifiés dans la hiérarchie comme en témoigne l'image ci-dessous tirée de <https://www.pageconcept.org/fishes.htm> (URL de l'image : <http://plongee.cours.free.fr/bio/images/poissons/F.88.Ppoissclass.P91JPG.JPG>). On notera qu'elle est elle-même présentée comme « simplifiée » ...

Et on a préféré l'ancien nom « sélacien » au lieu de *Elasmobranchii*...

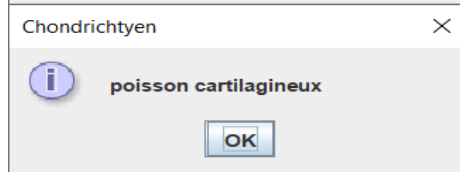
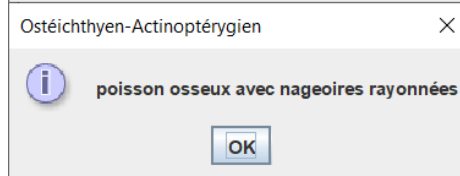


## Annexe : le contenu de infos.txt et l'affichage correspondant

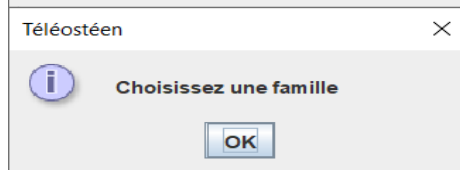
Poisson



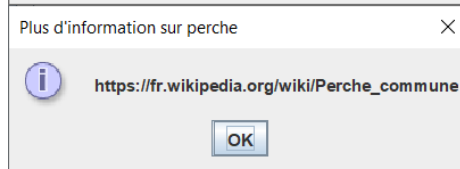
Chondrichtyen

Ostéichthyen-  
Actinoptérygien

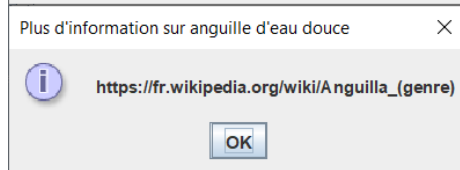
Téléostéen



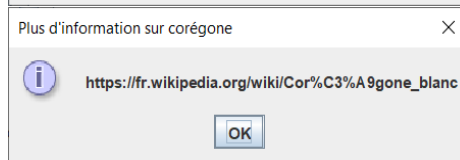
perche



anguille d'eau douce



corégone



omble chevalier

[https://fr.wikipedia.org/wiki/Omble\\_chevalier](https://fr.wikipedia.org/wiki/Omble_chevalier)

truite

<https://fr.wikipedia.org/wiki/Truite>

raie

<https://fr.wikipedia.org/wiki/Raie>

requin

<https://fr.wikipedia.org/wiki/Requin>

tanche

<https://fr.wikipedia.org/wiki/Tanche>

chevaine

<https://fr.wikipedia.org/wiki/Chevesne>

ablette

<https://fr.wikipedia.org/wiki/Ablette>

vengeron (gardon)

[https://fr.wikipedia.org/wiki/Gardon\\_\(poisson\)](https://fr.wikipedia.org/wiki/Gardon_(poisson))

brème

[https://fr.wikipedia.org/wiki/Br%C3%A8me\\_commune](https://fr.wikipedia.org/wiki/Br%C3%A8me_commune)

carpe

[https://fr.wikipedia.org/wiki/Carpe\\_commune](https://fr.wikipedia.org/wiki/Carpe_commune)

goujon

[https://fr.wikipedia.org/wiki/Goujon\\_\(poisson\)](https://fr.wikipedia.org/wiki/Goujon_(poisson))

chabot

[https://fr.wikipedia.org/wiki/Chabot\\_\(poisson\)](https://fr.wikipedia.org/wiki/Chabot_(poisson))

épinoche

[https://fr.wikipedia.org/wiki/Gasterosteus\\_aculeatus](https://fr.wikipedia.org/wiki/Gasterosteus_aculeatus)

lotte

[https://fr.wikipedia.org/wiki/Lotte\\_\(poisson\)](https://fr.wikipedia.org/wiki/Lotte_(poisson))

brochet

[https://fr.wikipedia.org/wiki/Grand\\_brochet](https://fr.wikipedia.org/wiki/Grand_brochet)

silure

<https://fr.wikipedia.org/wiki/Silure>