Travaux dirigés de Perl n°4/6

Modules, objets natif/Moose

—Université Gustave Eiffel—

Le but de ce TD est de finir l'apprentissage du langage *Perl* par des exemples de modules et de programmation objet.
Toutes les fonctions écrites devront être testées.

▸ **Exercice 1.**

Afficher la variable `@INC` *avec la commande* `perl -V` *Que conclure du dernier répertoire ?*

▸ **Exercice 2.** ***(Écriture d'un module)***

Le but de cet exercice est l'écriture un module et d'un script l'utilisant.

Écrivez un module Perl nommé `MonModule` *qui contiendra des fonctions que vous avez déjà écrites (par exemple, les fonctions* `TableMult` *du premier TD). Faites usage d'*`Exporter`*.*

▸ **Exercice 3. (Objet natif)**

1. *Nous allons définir une classe* `Disque`*; une instance de cette classe sera la représentation mémoire d'un disque dans le plan. Un tel objet contiendra les champs suivants :* `X` *et* `Y` *coordonnées du centre du disque et* `R` *le rayon de ce disque.*
   Définissez un constructeur pour cette classe acceptant trois paramètres pour initialiser chacun des champs. Faites en sorte d'avoir des valeurs par défaut (testez si vos paramètres sont définis) : par défaut un disque aura pour centre `(0,0)` *et un rayon* `1`*.*
   Affichez un objet de cette classe avec `Data::Dumper`
2. *Définissez une méthode* `surface` *donnant la surface du disque (la valeur de* π *se trouve dans le module* `Math::Trig`*, consultez sa documentation avec* `perldoc`*).*
3. *Écrivez une méthode* `dump` *renvoyant une représentation de ce disque sous la forme d'une chaîne de caractères, par exemple* `Disque:0,0,1`
   Optionnel : surchargez l'opérateur `""` *pour que la méthode* `dump` *soit utilisée.*
4. *Nous allons définir une classe* `Anneau` *comme héritant de la classe* `Disque`*. On considère pour cela qu'un anneau est défini comme un disque avec un rayon supplémentaire, dit interne, qui définit le sous-disque intérieur manquant à un disque pour définir un anneau. Ainsi, le seul champ à définir explicitement et à gérer dans* `Anneau` *est le champ* `RI` *(rayon interne).*
   Définissez un constructeur pour `Anneau` *acceptant quatre paramètres (les deux coordonnées, le rayon et le rayon interne). Par défaut, un anneau aura les caractéristiques d'un disque par défaut (appel au constructeur de la classe* `Disque`*) et un rayon interne valant zéro.*
5. *Redéfinissez les méthodes* `surface` *et* `dump` *dans la classe* `Anneau`*. Vous ferez en sorte d'utiliser la méthode correspondante de la classe mère pour rédiger ces deux méthodes.*

▸ **Exercice 4. (Objet avec Moose)**

1. *Définissez une classe* `Personne` *comportant le champ* `nom` *chaîne de caractères non modifiable.*
2. *Définissez une classe* `Soiree` *comportant les champs* `capacite` *(capacité, nombre maximal de personne) entier obligatoire non modifiable et* `potes` *tableau de* `Personne` *initialement vide et auto-déréférencé.*
3. *Via le trait '*`Array`*' de l'attribut* `potes`*, déléguez lui les méthodes* `entrer` *(fonction* `push`*),* `expulser` *(*`pop`*) et* `nbPotes` *(*`count`*).*
4. *Écrivez une méthode* `fete` *qui affiche le nom de tous les potes de la soirée.*
5. *Ajoutez un traitement avant la méthode* `entrer` *affichant le nom de la personne ajoutée.*
6. *Ajoutez un traitement après la méthode* `entrer` *vérifiant que le nombre de potes ne dépasse pas la capacité. Le cas échéant, elle expulse la dernière personne entrée et affiche son nom, sinon lui souhaite la bienvenue.*
7. *Écrivez un rôle Moose nommé* `Fetard` *qui dispose d'un attribut* `boisson` *(la boisson préférée du fêtard) chaîne de caractères obligatoire sans valeur par défaut, qui comporte une méthode* `boire` *affichant la boisson préférée et qui exige une méthode* `delirer`*.*
8. *Modifiez la classe* `Personne` *pour qu'elle implémente le rôle* `Fetard` *: la méthode* `delirer` *pourra afficher le nom et la chaîne* `'délire!'`*.*
   NB : La création des objets `Personne` *nécessite maintenant une* `boisson`*.*
9. *Modifiez la classe* `Soiree` *pour que le champ* `potes` *soit un tableau de* `Fetard`*.*
   La méthode `fete` *appellera sur chaque potes ses méthodes* `boire` *et* `delirer`*.*
   Vérifiez qu'une soirée ne peut plus comporter que des instances de classes implémentant `Fetard`*.*

▸ **Exercice 5.** ***(Usage de modules)***

Le but de cet exercice est la sauvegarde sur disque d'une page web avec ses images, en modifiant en conséquence les attributs `src` *des tags* `img` *afin d'utiliser les images sauvegardées en local.*

Pour simplifier, toutes les images du document HTML auront des URL absolues et auront un nom de fichier différent. Faites vos essais sur `https://formation-perl.fr/t/lwp.html`

Vous pourriez procéder par étapes de la manière suivante :

- *Téléchargement du contenu de la page HTML,*
- *Écriture d'une fonction* `getFileName` *prenant en paramètre une URL et renvoyant le nom du fichier (qui pourra être utilisé pour la sauvegarde en local) ; dans le cas de l'URL indiquée ci-dessus, elle reverra* `lwp.html` *Vous pouvez utiliser successivement les modules :*
  - `URI` *méthode* `path` *pour extraire de l'URL le chemin demandé (ici* `/t/lwp.html`*)*
  - `File::Basename`*, NB : la fonction* `basename` *de ce module n'a pas forcément besoin de second paramètre,*
- *Extraction des URL des images (utilisation de la méthode* `extract_links` *de la classe* `HTML::Element` *dont* `HTML::TreeBuilder` *hérite),*
- *Téléchargement du contenu des images,*
- *Sauvegarde des images sur disque (vous devrez utiliser la fonction* `binmode`*) ; la sauvegarde sur disque se fera dans le fichier dont le nom sera renvoyé par la fonction du deuxième point,*
- *Modification des attributs* `src` *des tags* `img` *dans l'arbre renvoyé par le parser (la méthode* `attr` *des objets* `HTML::Element` *permet aussi de modifier les attributs) avant la sauvegarde du HTML sur disque.*
- *Génération du HTML depuis l'arbre et sauvegarde sur disque.*

Vérifiez dans votre navigateur que les images de votre copie locale de la page s'affichent correctement et qu'elles sont locales elles-aussi.