TP N°5

HERITAGE ET REDEFINITION

Objectif du TP

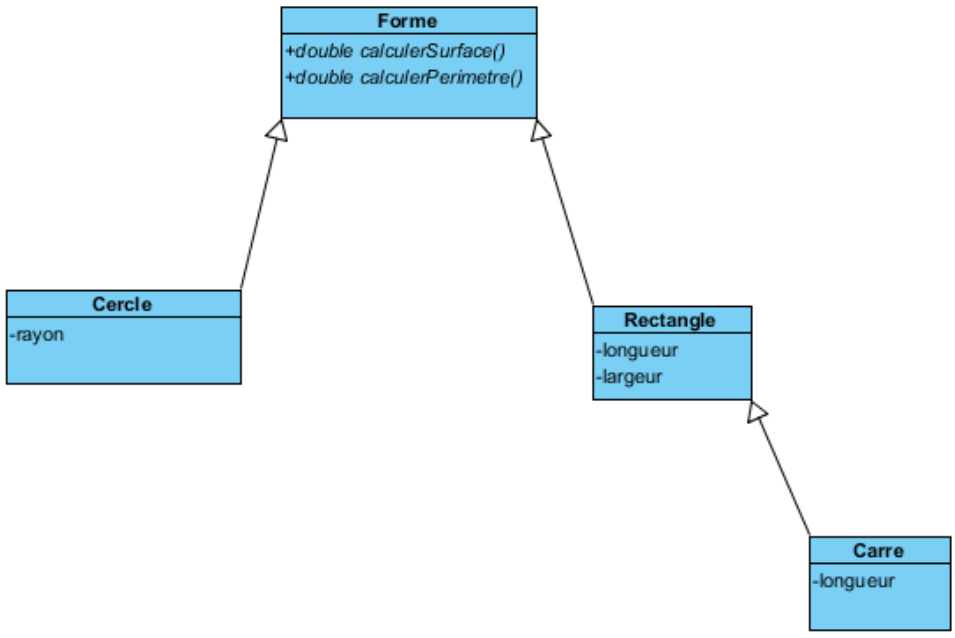
L'objectif de cet TP est de mettre en place un mécanisme d'héritage ainsi que la redéfinition de méthode.

Attention : dans tous les exercices qui vont suivre, vous devrez respecter les régles de l'encapsulation et mettre en place les constructeurs appropriés pour chaque classe.

Projet approche-objet

Exercice Forme

- La classe **Forme** va représenter la classe mère de diverses formes géométriques.
  - cette classe est abstraite
  - elle possède une méthode abstraite **calculerSurface**
  - elle possède une méthode abstraite **calculerPerimetre**
- Voici les autres classes à mettre en place avec leurs attributs :



- Implémentez toutes les classes de ce modèle objet
- Créez une classe **AffichageForme** :

- cette classe possède une méthode **afficher**. Cette méthode prend en paramètre une variable de type **Forme** et affiche le périmètre de la forme ainsi que sa surface.

- Créer une classe **TestForme** :
  - Créer une variable de type cercle, une de type rectangle et une de type carré et tester la méthode **afficher** avec ces diverses variables.
- **CONCLUSION** : comme vous le constatez la méthode **afficher** peut prendre en paramètre n'importe quelle instance d'une classe qui hérite de Forme. C'est l'essence même du **polymorphisme**.

Exercice CalculSalaire

Dans une application de gestion de la paie d'un journal quotidien « La Voix de Saint-Herblain », on a une hiérarchie d'objets suivants : Intervenant (classe mère), Salarié (classe fille de Intervenant) et Pigiste (classe fille de Intervenant).

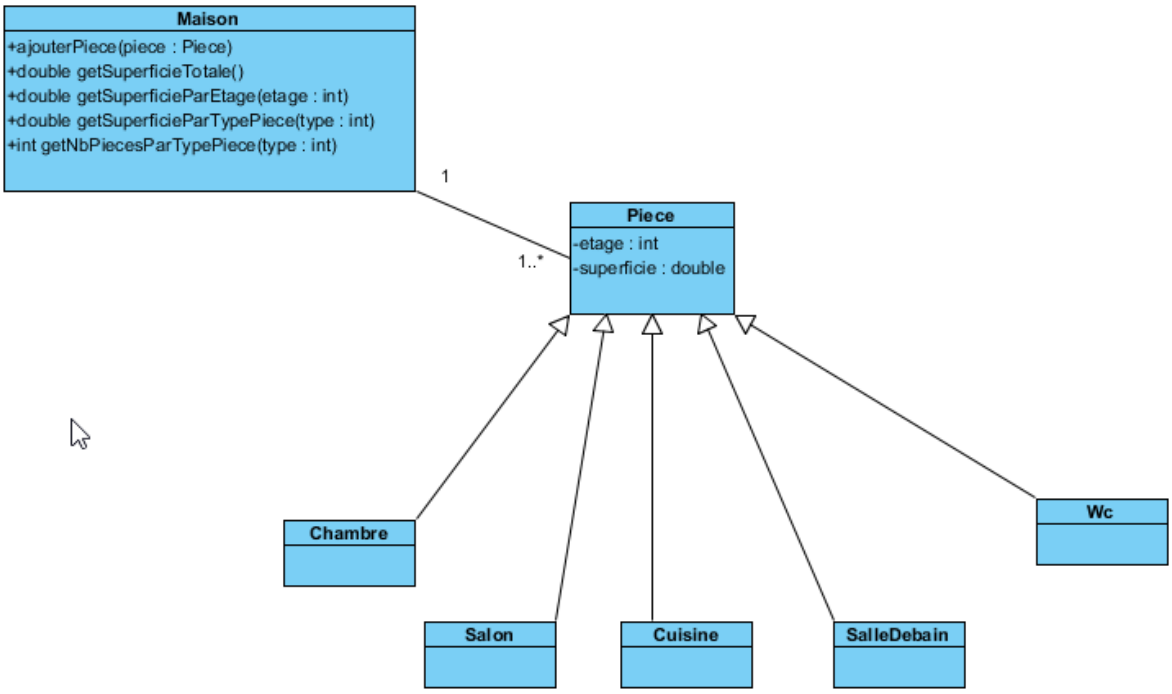
Etape 1 : implémentation des classes

- La classe **Intervenant** va représenter la classe mère des diverses presonnes travaillant pour le journal
  - cette classe a 2 attributs : **nom** et **prénom**
  - cette classe a une méthode **abstraite getSalaire**
- La classe **Salarie**:
  - est une classe fille de la classe **Intervenant**.
  - représente un salarié de la société, i.e. ceux qui ont un contrat
  - a un attribut d'instance : le montant du salaire mensuel
- La classe **Pigiste** représente les personnes payées à la journée. Cette classe a 2 attributs :
  - un attribut qui représente le nombre de jours travaillés pour la société durant le mois
  - un attribut qui représente le montant journalier de rémunération.
- Implémentez la méthode **getSalaire** pour les 2 classes : **Pigiste** et **Intervenant**.
- Développez une classe **TestIntervenant** :
  - Créez une instance de **Salarie** et affichez le résultat retourné par la méthode getSalaire
  - Créez une instance de **Pigiste** et affichez le résultat retourné par la méthode getSalaire

Etape 2 : la méthode afficherDonnees

- La classe **Intervenant** a une nouvelle méthode:
    - Cette méthode s'appelle **afficherDonnees** et affiche toutes les données concernant un intervenant : son nom, son prénom et son salaire
- Dans la classe **Salarie**:
    - Redéfinissez la méthode **afficherDonnees** afin que cette méthode affiche toutes les données du salarié : nom, prénom, salaire et « Statut : salarié »
    - **Important : évitez la duplication de code**
- Dans la classe **Pigiste**:
    - Redéfinissez la méthode **afficherDonnees** afin que cette méthode affiche toutes les données du pigiste : nom, prénom, salaire et « Statut : pigiste »
- Développez une classe **TestIntervenant** :
    - Pour chacune des instances de salarié et de pigiste invoquez ces méthodes afin de vérifiez qu'elles fonctionnent

Exercice Immobilier



Dans cet exercice nous allons modéliser une **maison** avec ses **diverses pièces** qui peuvent être de types différents : Chambre, Cuisine, Salon, Salle de bain, WC

- On va commencer par créer une classe abstraite **Piece**, qui a 2 attributs :
    - la superficie

- le numéro de l'étage. On considerera par convention que l'étage 0 désigne le RDC, 1 le 1er étage, et ainsi de suite..

- La classe **Piece** a un constructeur avec 2 paramètres permettant d'initialiser les variables d'instance superficie et etage.
- Comme le montre le diagramme de classes, la classe **Piece est la classe mère** de toutes les pièces de la maison. Cette classe mère a **5 classes filles** :
  - Chambre
  - Cuisine
  - Salon
  - SalleDeBain
  - WC
- La classe **Maison** va représenter une maison avec un unique attribut : un tableau d'objets de type **Piece**.
  - cette classe possède une méthode **ajouterPiece** qui permet d'ajouter une pièce à la maison.
  - cette classe possède une méthode qui retourne la superficie totale de la maison
  - cette classe a une méthode qui retourne la superficie d'un étage donné.
- Créer une classe **TestMaison** qui permet de tester la création d'une maison. Ajoutez des pièces de diverses natures à différents étages et vérifiez que toutes vos méthodes fonctionnent.
- **Plus difficile** : l'écriture des 2 méthodes suivantes demande un peu de réflexion
  - Dans la classe Maison, écrivez une méthode qui **prend en paramètre un type de pièce donné** et retourne la superficie globale pour ce type de pièce donné : par exemple, la superficie globale des chambres.
  - Dans la classe Maison, écrivez une méthode qui retourne le nombre de pièces d'un type donné : par exemple le nombre de chambres.

(Facultatif) Exercice JeuDeRole

Pour ce TP nous allons développer un jeu assez simple inspiré des jeux de rôles.

Dans ce jeu il va y avoir une interaction avec l'utilisateur qui sera donc amené à saisir des actions au clavier.

Pour simplifier le jeu, les actions se feront grâce à un système de menu.

Conseils avant de commencer le développement de ce jeu : imaginez les classes dont vous allez avoir besoin pour créer ce petit jeu.

Menu du jeu :

- Créer le personnage
    - lorsqu'on choisit cette option, un personnage va être créé avec 3 attributs :
        - force (entre 12 et 18) tirée aléatoirement
        - points de vie (entre 20 et 50) tirés aléatoirement
        - score (à 0)
- Combattre une créature
    - Le choix de cette option va permettre au personnage d'engager un combat. Ce combat n'est possible que si votre personnage a un nombre de points de vie > 0, sinon un message est affiché : « Votre personnage est décédé. Il a obtenu le score de X points. Veuillez créer un nouveau personnage ».
    - lorsqu'on choisit cette option, le personnage doit combattre une créature parmis les créatures suivantes :
        - Un loup : force (entre 5 et 10) et points de vie (entre 5 et 10)
        - Un gobelin : force (entre 7 et 12) et points de vie (entre 10 et 15)
        - Un troll : force (entre 12 et 20) et points de vie (entre 20 et 30).
    - le combat dure jusqu'à ce que votre personnage ou la créature soit victorieuse. Le combat se déroule de la manière suivante :
        - A chaque tour, on calcule l'attaque des 2 protagonistes. l'attaque est calculée de la manière suivante : force + nombre aléatoire entre 1 et 10.
        - Celui dont l'attaque est la plus forte remporte le tour
        - Celui qui remporte le tour inflige une quantité de dégats égale à la différence entre les 2 valeurs d'attaque calculées précédemment.
        - Cette quantité de dégats se soustrait au nombre de points de vie de celui qui a perdu le tour.
        - Si votre personnage perd, la partie est finie et le score du joueur est affiché.

        - Si votre personnage gagne le combat, son score augmente de : 1 si c'est un loup, 2 si c'est un gobelin et 5 si c'est un troll. Un message affiche alors l'issue du combat avec le nouveau score.
    - Vous pouvez engager un nouveau combat tant que votre personnage est encore en vie. L'objectif du jeu étant d'avoir un score maximum.
- Afficher score
    - cette méthode affiche le score.
- Sortir

(Facultatif) Exercice Parser

Dans cet exercice nous allons écrire un parseur capable à partir d'une chaine de caractères de construire une expression mathématique simple.

Le parseur doit être capable de parser une expression mathématique du type :

- x+3.5
- 3/x
- 2.5-x
- y+2
- etc…

Une fois le parsing effectué, vous devez être capable d'évaluer l'expression pour une valeur donnée de votre variable. Exemple

- l'évaluation de 3/x avec la valeur x=6 fournit le résultat 0.5
- l'évaluation de y+2 avec la valeur y=2 fournit le résultat 4

Spécifications :

- Le parseur est capable de parser n'importe quelle chaine de caractères contenant une variable (x dans l'exemple), un opérateur (+, -, * ou /) et une constante.
- Le **parseur** retourne un objet de type **Expression**, qui a 3 attributs
    - un membre gauche qui peut être une variable (x par exemple) ou une constante (par exemple 3)
    - un opérateur à choisir parmi +, -, *, /
    - un membre droit qui est également soit une variable soit une constante
- L'objet **Expression** a une méthode **evaluer** qui prend 2 paramètres :
    - le nom de la variable, par exemple x
    - la valeur de cette variable, par exemple 2.5
    - Pour évaluer la valeur de l'expression il doit être fait appel aux méthodes évaluer des membres.

Exemple de mise en œuvre :

String chaine = « x+3 » ;

Expression expr = Parser.parse(chaine) ;

double resultat = expr.evaluer(« x », 2.5) ; //Signifie que l'expression est évaluée avec x=2.5

System.out.println(expr.evaluer(« x », « 2.5)) ; // affiche 5.5

Commitez vos développements sur GitHub