C. Sempé

HERITAGE DE CLASSE

TP

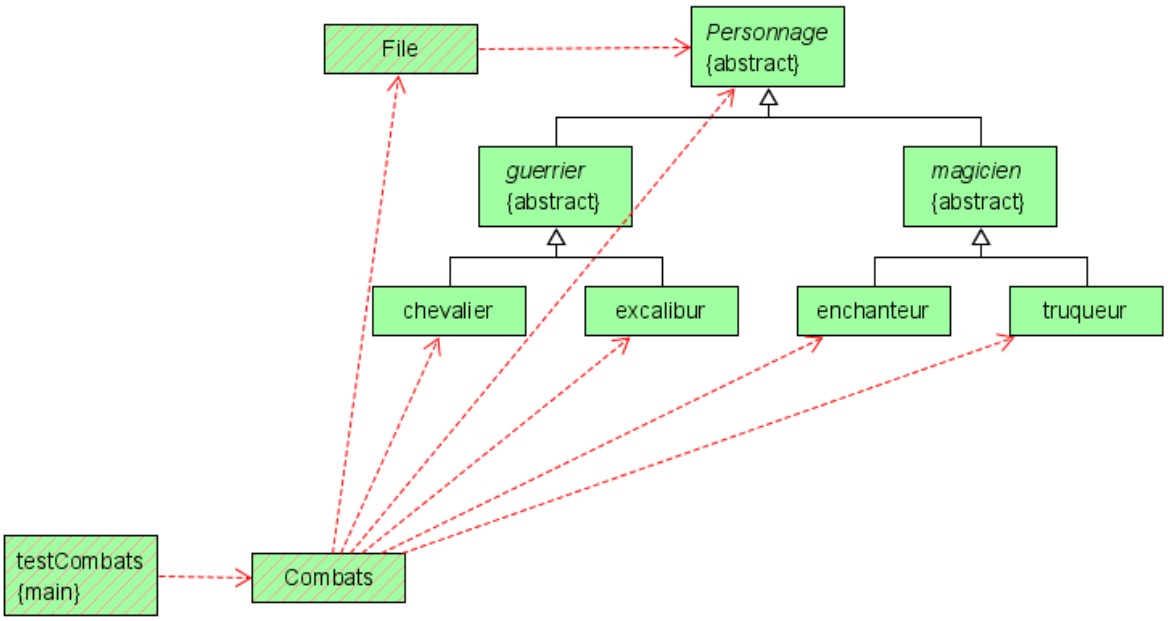
Nous voulons organiser des combats entre deux équipes. Chaque équipe pourra être constituée de plusieurs types de personnages ayant chacun leurs potentiels :

Chevalier = { vie, force, courage }
Excalibur = { vie, force, agressivité }
Enchanteur = { vie, magie, aura }
Truqueur = { vie, magie, malice }

Tous les personnages auront la possibilité de déclencher une *attaque* mais aussi pourront être *blessés.*

Implémentation :

Le logiciel pourra suivre la structure suivante, décrite par ce diagramme de classes :



La classe Personnage et ses sous-classes

Comme on peut le voir sur le diagramme de classe, on peut regrouper les personnages en plusieurs familles.
Ceci dans un souci de factorisation des caractéristiques. On pourra aussi utiliser cette hiérarchie afin de diffuser les méthodes attaque() et blesse() communes à tous les personnages.

- Ouvrez le fichier Personnage.java

```
public class Personnage
{   //CONSTANTES ---------------------------------------

   protected int vie;
   protected String typePers ;

   public int getVie()
   {
      return this.vie;
   }

   public int getType()  // renvoie le type de personnage selon les constantes
   {
      return this.typePers;
   }

   protected int Alea(int min, int max)
   {
      return  min +(int)(Math.random()*(max - min + 1));
   }

   public abstract int attaque();
   public abstract void blesse();
   public abstract String toString();
   public abstract String tuEsQui();      //retourne le type exact du personnage

}
```

Cette classe contient le dénominateur commun à tous les personnages : vie, typePers
Elle contient aussi des méthodes incomplètes pour le moment (abstract)
Enfin une méthode implémentée Alea (min, max) permettant de tirer au sort un entier entre min et max.

- Compilez et corrigez l'erreur du programme.
- Concevez la hiérarchie des caractéristiques selon le diagramme.

Précisions :

- méthode attaque()
  Elle retournera un entier A = Alea[0, vie] + Alea[0, ??] + Alea[0, ??]
  Chaque personnage fera participer ses jauges dans le calcul de l'attaque.

- méthode blessé()
  Elle change les caractéristiques en enlevant des points de potentiel
  vie = vie – Alea(20, 40) .
  Vous pouvez sophistiquer ce calcul si vous le souhaitez.

- Constructeurs
  Ils permettront d'initialiser les potentiels de chaque personnages grâce aux paramètres
  public chevalier(int vie, int force,int courage)
  La variable typePers devra elle aussi être initialisée

- toString()→ String permet de renvoyer le texte donnant les caractéristiques du personnage.

Test des personnages

Testez dans un main() à part, l'instanciation des différents personnages ainsi que attaque() et blesse() de chacun.

La classe combats

Cette classe va permettre de constituer les équipes et de faire s'affronter les personnages deux à deux.
On pourra utiliser des Files pour modéliser les équipes.

Dans une première version, les adversaires lancent chacun leur attaque. Selon la valeur, le perdant sera blessé.
Les personnages dont la vie sera inférieure à 0, seront considérés comme morts et exclus des équipes.

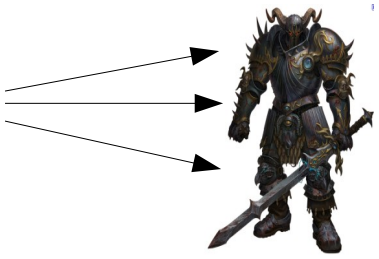
Le perdant est l'équipe qui ne possède plus de combattant.

Personnage avec intelligence artificielle

- Testez la classe combat dans sa version la plus élémentaire.

Pour rendre le combat plus réaliste chaque personnage pourra être atteint à trois endroits différents et posséder une mémoire afin d'acquérir de l'expérience.

tête, ventre, jambes.



Chaque personnage a son point faible mais inconnu des autres personnages

Potentiels possibles :

Chevalier = (tête : 40 % de dégâts), (ventre : 70 % de dégâts), (jambes : 100 % de dégâts)
Excalibur = (tête : 40 % de dégâts), (ventre : 100 % de dégâts), (jambes : 70 % de dégâts)
Enchanteur = (tête : 100 % de dégâts), (ventre : 70 % de dégâts), (jambes : 40 % de dégâts)
Truqueur = (tête : 100 % de dégâts), (ventre : 40 % de dégâts), (jambes : 70 % de dégâts)

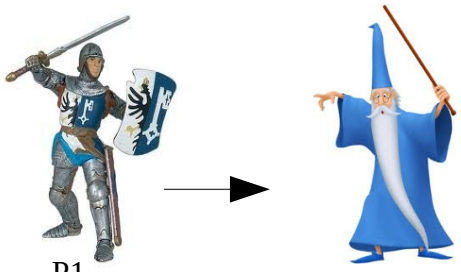
ex : si un Chevalier est attaqué au ventre avec une attaque de 100 alors ces dégâts seront de 100*70%=70

Modification des personnages

Comme vous le feriez vous même, un personnage peut acquérir de l'expérience en connaissant mieux son adversaire.
Au début un personnage ne connaît pas les points faibles de son adversaire. Au fur et à mesure des combats chaque personnage peut noter le lieu de l'attaque et y associer le dommage occasionné.
Au bout de trois attaques sur un même type de personnage, il devrait être capable de savoir ou est le dommage le plus important et ainsi viser toujours au même endroit à l'avenir.

Éléments techniques pour vous aider:

Un chevalier voulant attaquer un magicien devra procéder ainsi



P1 P2

1 P1.**choixCible**(typePersonnageP2) → cible
2 P2.**blesse**(cible, P1.attaque()) → degats
3 P1.**apprendre**(typePersonnageP2, cible, degats)

Mémoire

Chaque personnage devra posséder une mémoire qui s'enrichira au fur et à mesure des combats.

Exemple de la mémoire de dégâts d'un personnage a un moment donné:

La valeur -1 indique la non exploration

Exploration incomplète pour Excalibur la prochaine attaque devra se faire sur les jambes pour connaître définitivement le point faible

	1 Tête	*2 Ventre*	*3 Jambes*
1 Chevalier	-1	-1	-1
2 Excalibur	20	30	-1
3 Enchanteur	10	20	30
4 Truqueur	35	12	22

Si un combat doit se faire avec un Truqueur, on sait exactement où frapper : Max(35, 12, 22) -> Tête

- Modifiez judicieusement les classes afin que chaque personnage puisse garder la mémoire.

- Pensez à faire afficher à l'écran la mémoire du personnage.
- Test : dans un premier temps choisissez un seul personnage, faites en sorte qu'il affronte une série d'adversaires. Observez l'évolution de la mémoire.

Générer une documentation

- Faites un nouveau dossier Doc proche de vos fichiers
- Placez des commentaires /** blablabla **/ dans Personnage.java. Enregistrez
- Clic droit sur le nom du projet ->Generate documentation
- Choisissez le lieu de stockage de vos documentations (Doc)
- Appuyez sur le bouton Generate
- Consultez la doc html en double cliquant sur les fichiers