TD 7 — DevObj
Combat de Créatures

Contexte

Chaque joueur dirige une créature possédant des statistiques (points de vie, attaque, défense) et une capacité spéciale. Le dernier survivant remporte la partie. L'exercice doit illustrer les concepts suivants : encapsulation, abstraction, héritage et polymorphisme.

Règles principales

1. Le jeu se déroule *tour par tour*. Au début de son tour, un joueur choisit :
   — d'attaquer une créature adverse ;
   — de se défendre (réduit de moitié *tous* les dégâts subis jusqu'à son prochain tour) ;
   — d'activer sa capacité spéciale (si elle n'est pas passive).
2. Lorsqu'une créature atteint 0 point de vie, elle est éliminée.
3. Le combat continue dans l'ordre fixe des joueurs : on parcourt la liste, puis on revient au premier joueur. C'est un *cycle* tant qu'il reste au moins deux créatures vivantes.

Statistiques de départ

Créature	Points de vie	Attaque	Défense	Capacité spéciale
Dragon	100	20	10	Souffle de feu : attaque toutes les créatures (25 % de réussite par cible)
Loup	80	15	5	Rage (passive) : 50 % de chances d'enchaîner une seconde attaque
Gobelin	60	10	3	Esquive : 50 % d'éviter la prochaine attaque subie
Vampire	90	18	8	Drain de vie : inflige 14 dégâts et récupère la moitié en points de vie
Golem	120	10	15	Peau de pierre : réduit de moitié les dégâts subis ce tour

La **défense** est une *réduction fixe* soustraite aux dégâts *avant* l'application d'autres effets. Exemple : une attaque brute de 20 contre une créature ayant une défense de 5 inflige 15 dégâts, qui peuvent ensuite être divisés par 2 si la cible s'est défendue pendant ce tour.

Travail à faire

1. **Classe abstraite Créature**
   - Implémentez, en respectant les principes d'encapsulation, une classe `Créature` avec les attributs et les méthodes suivants :
     - `attaquer(cible : Créature)`
     - `seDefendre()`
     - `utiliserCapacite(cibles : Créature[], joueur ? : Joueur)`
     - `recevoirDegats(montant : nombre)`
   - Réfléchissez à l'intérêt de l'abstraction dans ce contexte.
2. **Sous-classes de Créature**
   - Créez les cinq classes filles : `Dragon`, `Loup`, `Gobelin`, `Vampire`, `Golem`.
   - Redéfinissez les méthodes si nécessaire.
   - Attention : certaines capacités sont *passives* (exemple : `Loup`), d'autres doivent être activées explicitement (exemple : `Dragon`).
3. **Classe Joueur**
   - Chaque joueur possède :
     - un nom ;
     - une créature ;
     - une méthode `jouerTour()` pour choisir l'action à effectuer.
4. **Classe Combat**
   - Créez une classe `Combat` qui gère :
     - la liste des joueurs ;
     - l'ordre cyclique des tours ;
     - la suppression des joueurs éliminés ;
     - la boucle principale du combat.
   - Ajoutez des messages pour afficher les actions réalisées, les dégâts infligés, les points de vie restants, etc.
5. **Diagramme UML**
   - Réalisez le diagramme de classes correspondant à votre solution.