Fantaisie Travaux Pratiques n°1

« Implémentation de diagramme de classes,
généricité, classe interne, interface Iterator, varag»

I. Rappel : environnement de développement Eclipse

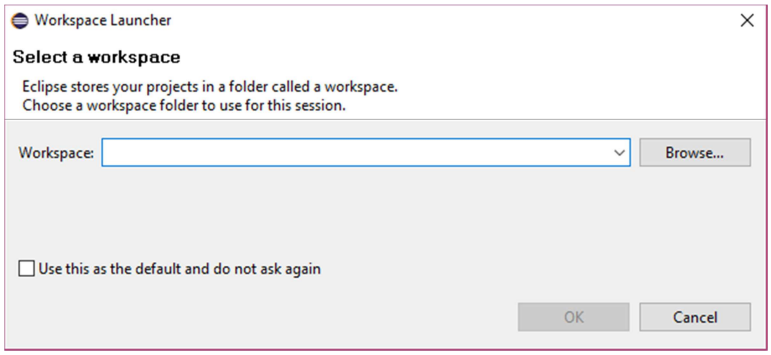
Au cours des séances de travaux pratiques nous utiliserons, comme l'année dernière, le logiciel Eclipse.
Le logiciel Eclipse est un environnement intégré de développement (IDE) pour le langage JAVA. Ce logiciel est gratuit, vous pourrez le télécharger chez vous sur le site www.eclipse.org. L'intérêt d'un tel outil, quand il est bien utilisé, est de permettre au programmeur de se concentrer sur l'essentiel de son travail, de se dégager de problèmes techniques de peu d'importance et de fournir des outils facilitant et encourageant la bonne écriture du code et la bonne conception d'applications.

1. Créez un projet

Lorsque vous lancez Eclipse vous devez **impérativement** choisir votre espace de travail.

Deux cas de figure :

- La fenêtre ci-contre apparaît :



Avec le bouton « Browse… » rechercher le disque qui vous est propre (disque Z:) puis noter le dossier dans lequel vous voulez vous placer (ex : Z:\ <<numEtudiant>>\Licence3\Semestre1\POO2\) puis compléter **OBLIGATOIREMENT** par \workspace.

- Eclipse s'ouvre directement :

Sélectionner File>Switch Workspace> Other…
Puis reprendre l'explication du premier cas de figure.

Eclipse fonctionne par projet. Pour créer un nouveau projet on sélectionne « file>new>project ». Après avoir choisi « java project », une fenêtre permet de créer un nouveau projet.
Il n'est pas recommandé d'utiliser le paquetage par défaut dans Eclipse. Pour chaque nouveau projet on créera donc un paquetage. Un clic droit

sur le projet fait apparaître un menu contextuel, le sous-menu « new » propose notamment de créer un nouveau paquetage.
a) Créez un nouveau projet nommé « *fantaisieTP1* ».
b) Créez un paquetage nommé « *attaque* » dans le dossier « src ».

2. Créez une nouvelle classe

On crée une nouvelle classe de la même manière qu'un nouveau paquetage : Un clic droit sur le paquetage « *attaque* » fait apparaître un menu contextuel, le sous-menu « new » propose notamment de créer une nouvelle classe. Le champ nom contient le nom de votre classe, **il doit commencer par une majuscule**. La case « public static void main » permet de générer automatiquement l'entête de la méthode main.
Créer la classe nommé **ForceDeCombat**.

II. Les Forces d'attaques – *Implémentation de diagramme de classes*

Implémenter l'ensemble des classes du diagramme de classes donné en page 4. Vous devez rigoureusement respecter les noms donnés aux classes et attributs sans chercher à les « réduire ».

N'oubliez pas qu'Eclipse est là pour vous aider. Utiliser les touches ctrl + espace pour :

- générer getter / setter (écrire get ou set puis ctrl + espace),
- générer un main (écrire main puis ctrl + espace),
- écrire System.out.println (écrire syso puis ctrl + espace)
- compléter un nom d'attribut ou de variable locale préalablement déclaré,
- toutes autres choses qui viendront au fur et à mesure du développement.

Utilisez aussi les générations automatiques sous « Sources », notamment : « Generate Constructor using Fields » permettant de créer un constructeur permettant d'initialiser les attributs choisis.

Ci-dessous un tableau récapitulatif des noms des forces d'attaque et le nombre de dégât correspondant.

Lance-pierre	10		Lave	80
Boomerang	20		Pics de glace	50
Arc (point par flèche)	50		Tornade	50
Epée	80		Griffe	20
Boules de feu	20		Lames d'Acier	50
Eclair	50		Morsure	?

Le nombre de dégât d'une morsure, dépend du monstre qui vous l'a faite …

III. Les Monstres – *Généricité, vararg*

Les monstres sont des êtres vivants. Chaque être vivant possède comme attributs :
- nom de type chaîne,
- forceDeVie de type entier.

Les monstres possèdent en plus comme attribut un tableau attaques contenant les attaques qu'ils peuvent produire. Mais attention, les monstres sont soit des êtres de Feu, soit des êtres de Glace soit des êtres Tranchants : ils ne peuvent en aucun cas être des êtres hybrides (avec un mélange d'attaque de feu, de glace et d'attaque coupante). Le type du tableau dépend donc du type de Monstre.

1. Créer le paquetage « *protagoniste* » dans lequel vous créerez les classes « EtreVivant » et « Monstre » en sachant que :
- Il ne peut pas y avoir d'objet instance de la classe « EtreVivant », que ses attributs sont initialisés par le constructeur, et qu'il existe des getters sur les attributs,
- La classe « Monstre » est générique, paramétrée par son pouvoir, et représentera donc soit un monstre de glace, soit un monstre de feu soit un monstre tranchant.

2. Dans le paquetage « *protagoniste* » créer l'Enuméré « ZoneDeCombat », contenant *aérien*, *aquatique* et *terrestre*.
   Ajouter au monstre l'attribut zoneDeCombat, qui sera initialisé par le constructeur. Créer aussi un getter sur cet attribut.
   AIDE ECLIPSE : pour créer l'attribut, écrire `private` puis Z ctrl+espace, Eclipse complètera avec le type « ZoneDeCombat » et importera l'énuméré. Puis tapez z ctrl+espace, Eclipse donnera à l'attribut le même nom que le type. Avec 2 lettres Eclipse a écrit pour vous :
   ```
   import enumere.ZoneDeCombat;
   private ZoneDeCombat zoneDeCombat;
   ```

3. Dans le paquetage « *protagoniste* » créer l'Enuméré « Domaine », contenant *feu*, *glace* et *tranchant*.
   Ajouter au monstre l'attribut domaine, qui sera initialisé par le constructeur. Créer aussi un getter sur cet attribut.

4. Le nombre d'attaques par monstre n'est pas fixe, certains n'en possèdent qu'une alors que d'autres en possèdent 3 … Utilisez un vararg afin de transmettre les attaques au constructeur pour qu'il puisse initialiser l'attribut attaques qui est un tableau.
   Un warning apparait sur la ligne du constructeur : « Type safety: Potential heap pollution via varags parameter attaques ».
   Cliquer 1 fois sur l'ampoule et sélectionner la correction :
   « Add @SafeVarargs »

5. Générer les méthodes *toString* dans les classes « EtreVivant », « Monstre » et « ForceDeCombat ».

6. Dans un nouveau paquetage « *testsfonctionnels* » créer la classe « TestGestionAttaque » contenant un `main`. Créez le monstre dragotenebre ayant une force de vie de 200 et préférant une zone de combat aérienne. Son domaine est le feu et ses attaques sont : la lave, les éclairs et les boules de feu. Affichez-le !

IV. Les attaques des Monstres – *Classe interne, interface Iterator*

Pour cette partie n'hésitez pas à reprendre l'exemple du cours numéro 2, plus précisément les diapositives dont le titre est ***Classe Iterable*** et comportant la classe « Panier » et sa classe interne « Iterateur ». Nous suivrons à peu près le même schéma.

Pour gérer les attaques des monstres il vous faut créer la classe « GestionAttaque » interne à la classe « Monstre » qui implémentera l'interface ***Iterator***.

Cette classe contiendra 2 attributs :
- attaquesPossibles : initialisé avec l'attribut attaques de la classe englobante,
- nbAttaquesPossibles : initialisé avec le nombre de case du tableau attaques de la classe englobante.

Les méthodes à implémenter obligatoirement sont :
- *hasNext()* : réorganise le tableau d'attaques avant de pouvoir répondre.
  Elle effectue un parcours du tableau en regardant si l'attaque est toujours opérationnelle (attribut operationnel de la classe « Pouvoir ») Si ce n'est pas le cas alors elle prend le pouvoir contenu dans la case n° nbAttaquesPossible -1 et le range dans la case en cours, puis diminue le nombre d'attaques possibles.
  Elle retourne un booléen indiquant si le nombre d'attaques possibles est non nul.
- *next()* : retourne une attaque aléatoire dans le tableau attaquesPossibles allant de l'indice 0 à l'entier nbAttaquesPossibles - 1.

Dans la classe englobante « Monstre » ajouter les méthodes :
- *entreEnCombat()* : régénère tous les pouvoirs contenus dans le tableau attaques, puis affecte un nouvel objet à l'attribut gestionAttaque.
- *attaque()* : retourne l'attaque suivante s'il en reste, sinon retourne null.

Dans la classe « TestGestionAttaque », faire combattre le monstre et demander plusieurs fois l'attaque suivante. Les attaques n'étant pas encore réellement codées, le pouvoir du Monstre ne faiblit pas et il peut en lancer indéfiniment !

Diagramme de classes