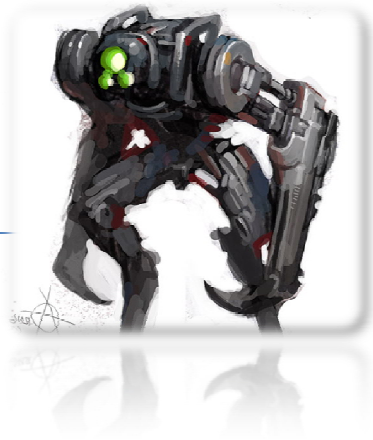

TD n°3 – les robots

Le but de ce TP sera de faire un programme pour contrôler un robot, et le faire se déplacer dans un environnement…

La classe Element

Un peu à la manière du jeu de rôle fait pendant le devoir, on va créer une classe Element. Cette classe sera la mère de tout ce qui existera dans le jeu final (mur, ennemi, bonus…). Elle comportera les informations suivantes :

- **`type`** : de type string, cet attribut permet de savoir quel est le type de l'élément…
- **`description`** : de type string, décrit l'élément…
- **`getDescription`**( ) qui affiche la description de l'élément.
- **`setDescription`**(**`string s`**) qui met à jour la description d'un objet .
- **`getType`**( ) qui retourne une chaine de caractère indiquant le type de l'élément.

La classe Robot

Écrivez une classe **`Robot`** capable de simuler le déplacement d'un robot sur un plan orthonormé (4 directions : gauche, droite, haut, bas). La classe **`Robot`** aura les attributs suivants :

- un **`nom`**
- une **`position`** (reprenez la classe Point2D faite au TP 7 que vous modifierez au besoin)

Outre les constructeurs adéquats, définissez les méthodes suivantes :

- **`allerAGauche`**`()` : qui déplace vers la gauche de `1`;
- **`allerADroite`**`()` : qui déplace vers la droite de `1`;
- **`allerEnHaut`**`()` : qui déplace vers le haut de `1`;
- **`allerEnBas`**`()` : qui déplace vers le bas de `1`;
- **`getPosition`**`()` : renvoie la position courante du robot (retourner la position);
- **`sePresenter`**`()` : affiche un texte indiquant le nom et la position du robot.

Testez chacune des méthodes dans un main…

Énergie

Ajouter à la classe `Robot` :

- un attribut `niveau` définissant le niveau de combustible c'est à dire le nombre de déplacements possibles (`0` à `100`).
- une méthode `getNiveau()` qui renvoie le niveau de combustible. S'il vaut `0`, le robot ne peut plus se déplacer.

Pensez à modifier les méthodes de déplacements pour mettre à jour le niveau de carburant !

Testez si les modifications fonctionnent en faisant bouger le robot plus de 100 fois…

Case libre

On veut pouvoir faire bouger le robot sur un plateau… Il faut donc falloir avoir des cases libres. On va donc créer une classe **`CaseLibre`** avec les informations suivantes : le **`type`**, c'est « case libre » et la **`description`**, c'est le caractère espace (" "). Il n'y a donc pas besoin de faire grand-chose pour cette classe…

Mur

On veut aussi pouvoir empêcher le robot de bouger… On va donc créer une classe **`Mur`** avec les infos suivantes : le **`type`**, c'est « mur » et la **`description`**, c'est le caractère dièse ("#"). Il n'y a donc pas besoin de faire grand-chose non plus pour cette classe…

Jeu

C'est avec cette classe que l'on va faire le lien entre tous ces objets. Pour ne pas trop compliquer les choses, on va supposer que le plateau est toujours de la même taille : 20*20 cases. Donc il faut avoir un tableau à deux dimensions (pour les lignes et les colonnes) de taille 20*20.

Ensuite, réfléchissons au type… Si on fait un tableau de type **`string`**, on ne pourra mettre que des chaines de caractères dedans, et pas des éléments du jeu… Donc pas de type **`string`**, ni **`int`**…

De type **`Element`** ? Presque… Le type du tableau sera de type pointeur sur **`Element`**. Grâce à ça, on va pouvoir mettre dans ce tableau tous les éléments et leurs dérivés que l'on veut ! Par exemple, pour mettre dans la case en haut à droite un mur, on fait : **`plateau[0][0]=new Mur();`** Donc pour résumer, le tableau sera déclaré comme ça : **`Element *plateau[20][20]`**.

Le jeu contient aussi un robot… Ajouter donc un attribut `protected` de type pointeur sur **`robot`**.

Enfin, ajoutez ces méthodes :

- **`void initJeu()`** : qui demande à l'utilisateur un nom pour le robot, qui le créé avec comme position (0,0) et qui initialise le plateau avec des cases vides partout (ou avec quelques murs, c'est mieux…)
- **`void affiche()`** : qui affiche les descriptions de toutes les cases. Faites attention à ne pas afficher une case dont la valeur est à **`NULL`** !
- **`void bougeRobot()`** : qui demande à l'utilisateur dans quel direction il veut déplacer le robot. Avant d'effectuer le déplacement demandé, vérifiez que le robot n'ira pas sur une case avec un mur, mais aussi qu'il ne sorte pas du plateau !
- **`void videJeu()`** : qui vide le plateau (pensez **`delete`**) ainsi que le robot.

Testez toutes ces fonctions dans un main…