MANUEL UTILISATEUR

IRHack

Ingénieur 2000 IR1
Jean- Baptiste Renaudi – Laurent Barbisan

Sommaire

Documentation utilisateur

Informations générales concernant le logiciel

But et généralités

Ce logiciel jeu de rôle mettant en scène Pingou le pinguin, au milieu de l'antartique. Ce monde recelle de milliers de créatures maléfiques et anciens pinguins morts de froid hantant ce monde infini. Afin de ne pas finir comme ces derniers, il est indispensable qu'il retrouve le chemin de son igloo.

Configuration recommandée

Toute plaforme ayant la machine virtuel java d'installée (de préférence une version superieur ou égal à la 1.3.1)

Matériel nécessaire a l'installation

Il est necessaire d'avoir les fichiers fournis avec ce manuel utilisateur c'est à dire:

- Lanceur.java
- Package Game
- Package Interface
- Package Lien
- Package Utils

Pour l'installé, il est nécessaire d'avoir un utilitaire d'extraction au format ZIP (ex:WinZip). Après avoir extrait les fichiers du zip, il suffit de les placer dans n'importe quel répertoire de votre machine.

Mode d'emploi

Démarrage du logiciel

Pour lancer le programme, suivez la procédure suivante :

1. Ouvrez une console MS-Dos (Windows) ou Xterm (Linux)
2. Placez vous dans le répertoire contenant le fichier Lanceur.class
3. Si le repertoire ne contient pas les fichiers executables (*.class), voir *section Compilation des sources* ci-après.
4. Puis enfin executez la commande suivante :

```
java Lanceur
```

Compilation des sources

Si le repertoire classes n'existe pas dans ses sources, il est alors nécessaire de compiler le logiciel pour le faire fonctionner.

Pour cela, se placer dans le repertoire racine au source (c'est à dire le repertoire contenant le fichier Lanceur.java)

Créer le répertoire classes (mkdir classes sous Linux ou md classes sur Windows).

Exécuter enfin la commande suivante :

```
javac 'find ./ -name "*.java"' -d classes
```

Alors tous les fichiers .class seront créer dans le répertoire classes. Le logiciel est enfin prêt à fonctionner.

Commande et réponses du logiciel

Lorsque vous avez démarré le programme, une fenêtre graphique de jeu apparait. Cette fenêtre contient toutes les fonctionnalités permis par le jeu.

Dés l'apparition de cette derniere, un pinguin se place dans la salle visible. Vous êtes ce dernier, la partie commence.

Deplacement

Via les touches du clavier

Par l'intermédiaire du clavier, nous vous offrons les interactions de déplacement, à savoir :

Haut	=>	haut	ou	8
Bas	=>	bas	ou	2
Droite	=>	droite	ou	6
Gauche	=>	gauche	ou	4

Via le PopUpMenu (menu dynamique)

Par l'intermédiaire de la souris, nous vous offrons les interactions de déplacement. Il suffit de faire un clique droit sur Pingou (votre personnage). Une menu listant les actions pouvant être effectué par lui sur lui, comme les fonctions de déplacement.

ACTION

Toutes les actions sur le jeu sont disponibles à travers le **PopUPpMenu**. Ce dernier apparait après un clique droit de la souris sur un objet du jeu (panel graphique). De suite, la liste d'actions associées à cet objet (créature, fontaine, etc...) est affiché à ce même emplacement.Une simple selection de l'action permettra ensuite son execution.

Par défaut la source de l'action sera Pingou.

Ex: *Pour boire dans une fontaine, un simple clique droit de la souris sur cette fontaine affichera le menu des actions possibles de Pingou sur cette fontaine, BOIRE par exemple.*

Si on veut faire interagir 2 objets (dont la source est autre que Pingou), un premier clique gauche de la souris sur la source est nécessaire d'abord, puis ensuite, un clique droit sur l'objet destination fera apparitre les actions.

Ex: *Pour faire interagir 2 objets de la sacoche. On clique gauche sur le premier objet, puis clique droit sur le second.*

Malheureusement, le module d'affichage du contenu de la socoche n'est pas implémenté, donc cet fonctionnalité est obsolèteà à ce moment.

DEFINITION DES ACTIONS POSSIBLES

Aventurier sur Aventurier

Action	Cible	Consequence
Monter	Pingou	Deplacement d'une case vers haut
Descendre	Pingou	Deplacement d'une case vers droite
Gauche	Pingou	Deplacement d'une case vers gauche
Droite	Pingou	Deplacement d'une case vers droite
Attaquer	Pingou	S'attaque lui-même (Kamikaze)
Description	Fontaine	Affiche une description de la fontaine

Aventurier sur Fontaine (Jade ou Bénite)

Action	Cible	Consequence
Boire	Pingou	Afflige Bonus ou Mallus
Changer*	Pingou	Change sa profession
Description	Fontaine	Affiche une description de la fontaine

Aventurier sur Pile (Objets)

Action	Cible	Consequence
Prendre*	Objets	Prend le premièr objet de la pile
Description	Fontaine	Affiche une description de la fontaine

* Fonction apparaissante mais non implémenté ou causant des problèmes

DEFINITION DES ELEMENTS DU JEU

Ce jeu est constitué de nombreux objets, chacun ayant un comportement ou des actions spécifiques à sa catégorie :

Case

Cet objet est n'a aucune intéraction avec les autres composants, il peut être seulement occupé par un meuble et/ou une créature.

Mur

Cet objet est un obstable non déplacable et indestructible. En effet, une fois placé dans la salle lors de sa création, il ne peut subir aucune action.

Créature :

La créature possède plusieurs caractèristiques:

Vitesse	vitesse de déplacement (nombre d'action réalisable par la créature durant un tour).
Attaque	dégat infligé à la créature adverse
Défense	Protection contre les dommages infligés
Endurance	Resistance de la créature
Point de Vie	Nombre de point avant que la créature ne meurt(4 *Endurance
Experience	Nombre de créature tuée (permet d'augmenter de niveau)
Niveau	Le niveau représente la puissance du personnage

Créature : Monstre

Le Monstre est une créature gérer par l'ordinateur

Créature : Aventurier

L'aventurier est la créature gérée par le joueur.

L'aventurier peut changer de profession, c'est à dire qu'il peut acquerir des spécialitées :

Sans	L'aventurier est une créature sans bonus.
Guerrier	L'aventurier acquière : L'action 'Charger' *Lors d'une charge, le bonus d'attaque double pendant un round.* Modificateur sur Valeur d'attaque *VA = base + niveau* Modificateur sur l'Endurance *E = base + (3*Niveau)/2*
Prêtre	L'aventurier acquière: L'attribut L'attribut 'Mana' *Nombre de point de magie* Le Sort 'Soigner' *redonne n point de vie pour n Mana*
Sorcier	L'aventurier acquière: Modificateur sur 'Endurance' *E = base + (niveau+1)/2* Modificateur sur 'Vitesse' *V = base + niveau/3* L'attribut L'attribut 'Mana' *Nombre de point de magie* Le Sort 'Blaster” *inflige n point de dégats suité à d cases pour un cout de n+d Mana*

Meuble :

Ceci est la définition d'un objet altérable mais non déplacable.

Meuble : Fontaine :

Ceci est une définition particulière de Meuble. Elle ajoute l'action boire à ses decendants. Une fontaine ne peut être bue qu'une seule fois.

Meuble : Fontaine : Jade

Cet objet est une fontaine maléfique. Les créatures peuvent y boire mais elle inflige des points de blessure aléatoirement de 1 à 10.

Meuble : Fontaine : Bénite

Cet objet est une fontaine bienfaisante. Les créatures peuvent y boire et ainsi annule leurs points de blessure aléatoirement de 1 à 10. Si la créature n'a aucun point de blessure, alors cette dernière n'a aucun effet sur lui.

Meuble : Fontaine : Souvenirs

Cet objet est une fontaine bienfaisante. Elle donne une profession à celui qui boit son eau. (cf. *Créature:Aventurier).*

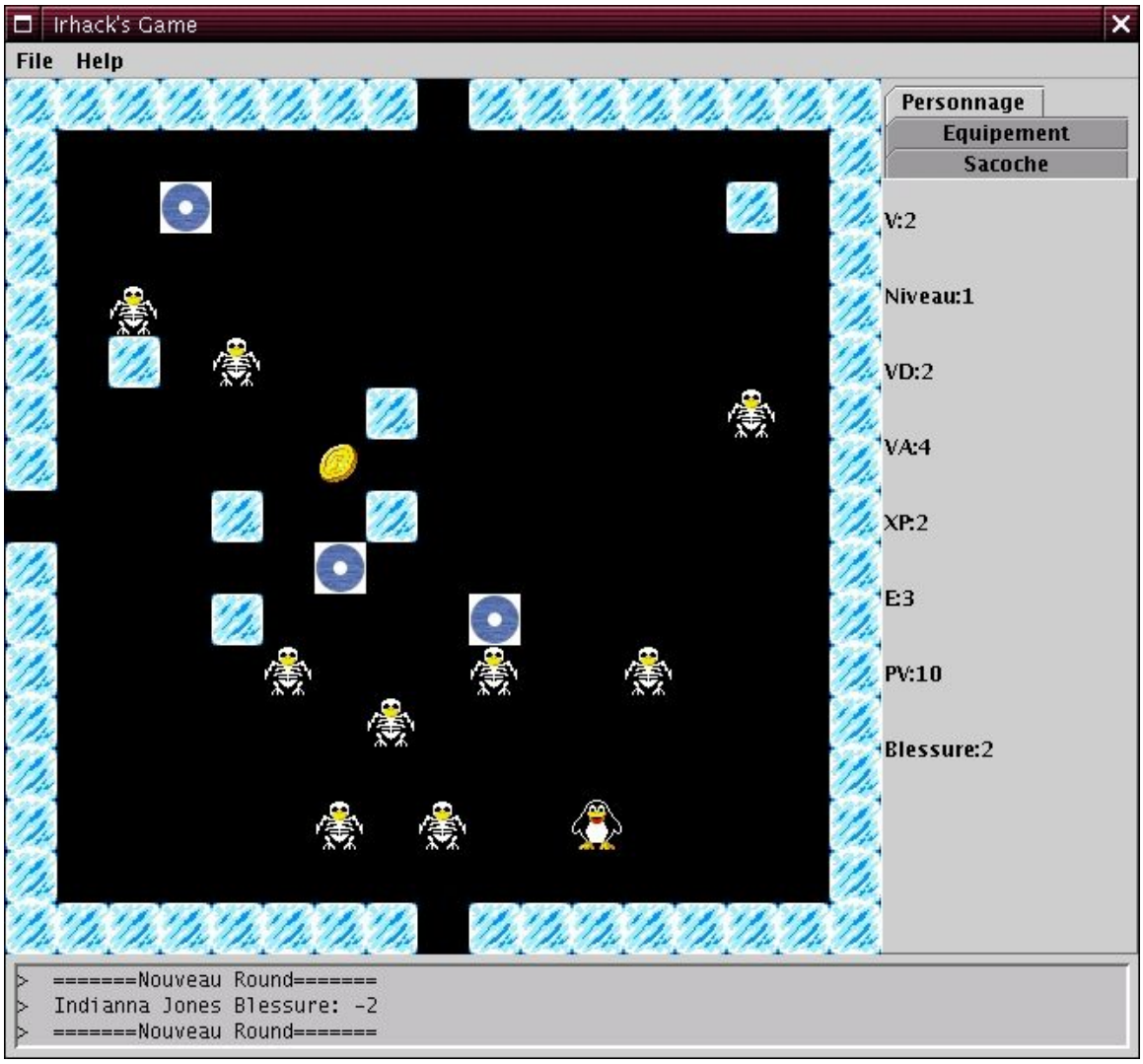
Meuble : Pile

Cet objet contient une pile d'objet, l'aventurier peut prendre l'objet du dessus.

L'aventurier peut aussi déposer des objets dans cette pile.

Interface Utilisateur

Voici l'interface utilisateur:



Carte

La carte montre une salle avec tout ces elements. Lors du franchissement d'un e porte, la salle dans la quelle le joueur entre est affichée.

Onglet

Les onglets permettent de voir les caractéristiques de l'aventurier, il y a trois onglets :

- Personnages
- Equipement
- Sacoche

Personnage

L'onglet Personnage affiche les attributs du personnage (c'est à dire, point de vie, point d'endurance, etc...)

Equipement

L'onglet Equipement affiche l'equipement actuellement porté par l'aventurier

- Bouclier
- Arme
- Armure

Sacoche

Le sacoche permet de stocker jusqu'a 10 objets.

Console

La console permet de visualiser les actions effectuées dans le jeu:

- Ajout de modificateur sur le joueur
- Action efffectuer par les Elements du jeu
- Etc...

Personnalisation du jeu de rôle

Gràce à la modularité il est facile d'implementer de nouveau éléments

Création d'un nouvel element

Les Elements du jeu de rôle sont stockés dans le package 'Game.Elements'

Un element est en fait une classe. Le nouvel objet doit hériter de la classe, ou d'une sous- classe de Occupant.

Un Element est composé de trois choses:

- Les actions liée à cette elements
- Les contenaires (sacoche, poche...)
- Les modificateurs (Point de Vie, Endurance, Mana,...)

Fonction nécessaire dans la classe

Constructeur avec en paramètre une Classe Cellule

Ajout d'action (facultatif)

Ajout d'action À l'élément, écriture d'une méthode :

```
public type Action(<type> Source, <type> Destination)
```

L'objet destination sera envoyer par la class ActionManager, suivant le type de classe de <type>.

Remarque:
La plupart du temps, l'objet source est l'element que vous créer.
Les Elements dérivé de Créature ont une action “p_jouer”, qui leur donne l'ordre de jouer quand c'est leur tour.

Ajout de contenaire (facultatif)

Le contenaire est ce qui va permetre de stocker tous les objets autorisés.

1. Definir un conténaire de la façon suivante :

```
private Contenaire nom_contenaire;
```

2. Dans le constructeur de l'element ajouter:

```
nom_contenaire = new Contenaire(“Classe_de_objet”, Nomb_e Stockable);
```

Remarque :

Si Nombre_Stockable est égal à zéro le nombre d'objet est infini.

Classe_de_objet: est egal au type d'element qui sera stocké, par exemple, “Game.Elements.Objet” stockera toutes les classes Game.Elements.Objet et ces dérivés.

Ajout de Modificateur(facultatif)

Les Modificateurs sont calquer sur le même principe que les contenaires, ainsi :

1. Definir un conténaire de la façon suivante :

```
private Attribut nom_attribut;
```

2. Dans le constructeur de l'element ajouter:

```
nom_contenaire = new Attribut(Valeur_de_base);
```

Remarque :

Imaginons que l'on veuille ajouter 3 Attributs:

- Niveau = 1
- Endurance = Niveau *2
- Vitesse = Endurance / 3

Declaration :

```
private Attribut Niveau, Endurance, Vitesse;
```

Initialisation :

```
Niveau = new Attribut(1); /* Valeur de base */
Endurance = new Attribut(0); /* Valeur de base */
Endurance.Add_Mod(new ModificateurMult(Niveau, new Modificateur(2);
Vitesse.Add_Mod(new ModificateurDiv(Endurance, new Modificateur(3);
```

Ajout à l'interface graphique

Il faut à présent implémenter l'image lié au nouvelle élément

Aller dans la classe Link.Linktable
Dans la méthode `private void init()`
Rajouter la ligne suivante :

```
this.addlink("Nom_de_la_classe",Image.getImage("ULRImage", kit);
```

Ajout dans le déroulement du jeu

Le Elements sont ajouté au hasard lors de la création d'une salle (Salle.java), il suffit d'ajouter suivant le type de classe ce code :

```
New nom_Element(cellule);
```

Pour les Classes héritant de Meuble, c'est à mettre dans la méthode Placemeuble.

Pour les Classes héritant de Moble, c'est à mettre dans la méthode PlaceMobile.

Développement

La conception et la modularité ont été les points clefs du projet, c'est pour cela que le projet suit le modèle MVC (Modèle Vue Controleur)

- Modèle : Irhack
- Vue : Interface Graphique
- Controleur : Protocole de transfert de donnée

Irhack => Interface Graphique

Algorithme générale

La Classe controleur, initialise IrHack et l'interface graphique.

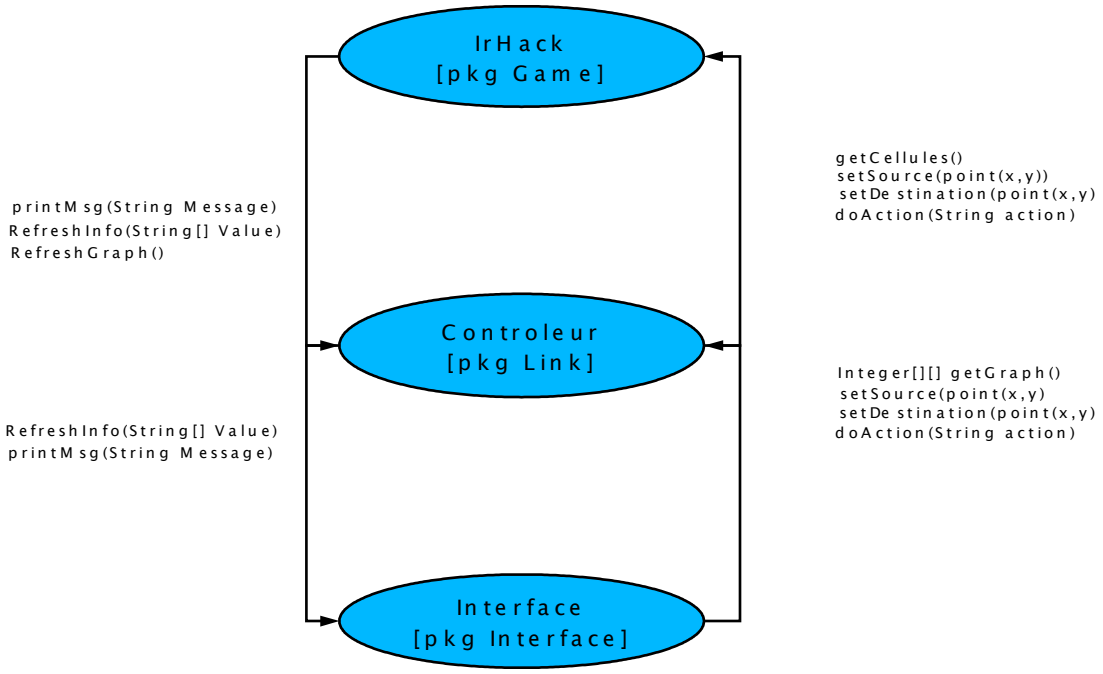
La Classe Donjon initialise une Salle qui initialise ses élements.

Donjon lance ensuite une boucle qui fait jouer tout les Créatures tour à tour (l'aventurier compris) par ordre de vitesse.

Si l'aventurier meurt le jeu s'arrête.

Interface MVC

Voici le schema de la structure MVC:

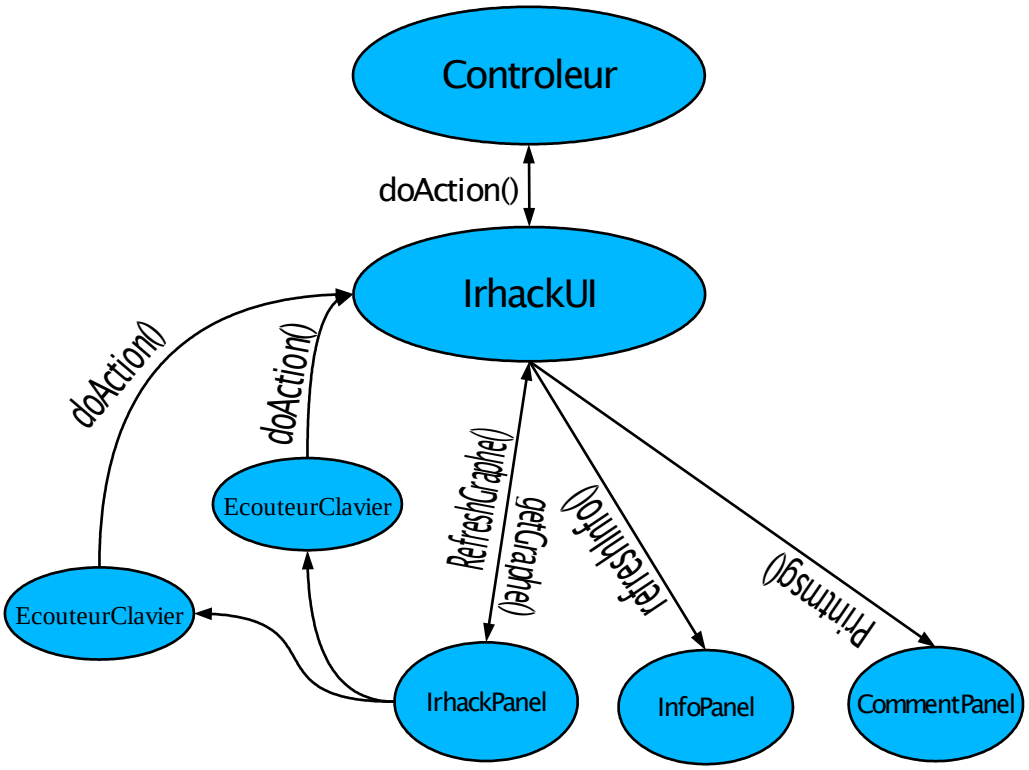


Le principe du MVC est de séparer les classes métiers et les classes servant d'interfaces.

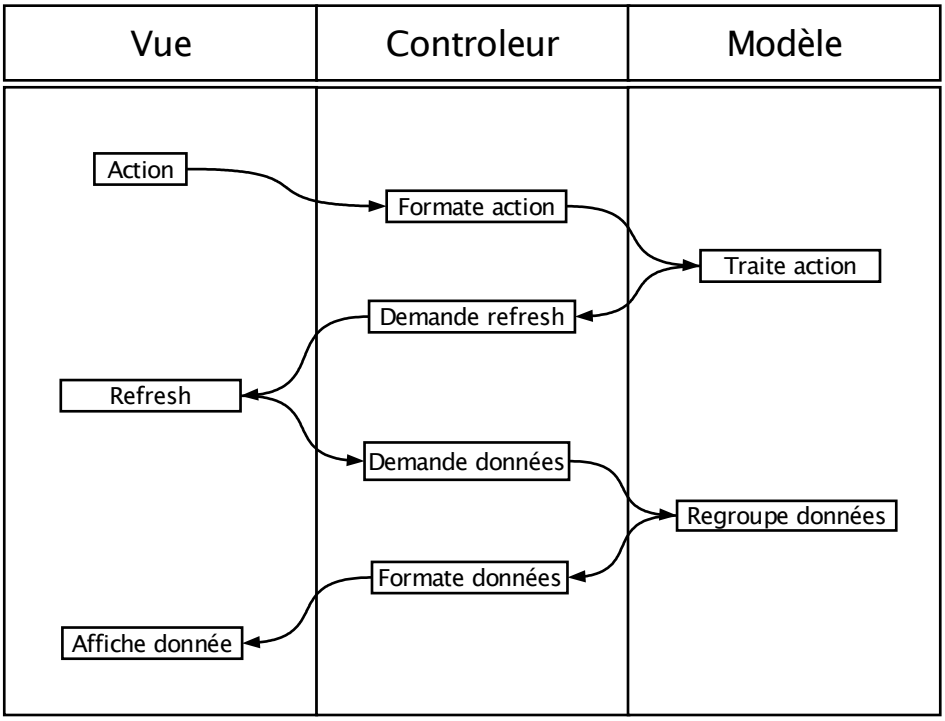
Un objet intermédiaire, appelé controleur fait la liaison entre ces deux groupe de classes.

Des méthodes standards sont attribuées au controleur permettant ainsi au deux groupes de classes de dialoguer. Bien sûr, un reformatage des données est effectué lors d'un dialogue (protocole de transfert des données).

Interface Graphique



Comme vous pouvez le voir sur le schema ci-dessus, l'objet IrhackUI centralise toutes les méthodes de dialogue avec le controleur. Pour mieux expliquer ceux-ci nous allons détailler une séquence de dialogue entre l'interface et le Jeu.



Explication textuelle de la séquence:

1. Le joueurappuie sur la touche 'flèche droite' du clavier.
2. L'écouteur clavier détecte l'évènement.
3. L'écouteur clavier transmet l'action à IrhackUI.
4. IrhackUI transmet l'évènement au controleur.
5. Le controleur formate l'évènement.
6. Le controleur transmet l'évènement formaté au jeu.
7. Le jeu traite l'évènement.
8. Le jeu transmet l'ordre de rafraichissement de l'ecran au controleur.
9. Le controleur le retransmet à IrhackUI.
10. IrackUI transmet l'ordre de rafraichissement a IrhackPanel(Ecran).
11. IrhackPanel ordonne a IrhackUI de récupérer les informations écran.
12. IrhackUI retransmet l'ordre au controleur.
13. Le controleurtransmet l'ordre au jeu.
14. Le jeu retourne les données écran.
15. le controleur les formate.
16. Le controleur les transmet a IrhackUI.
17. IrhackUI transmet les données à IrhackPanel.
18. IrhackPanel affiche les données reçues.

Gestion de Elements

Les éléments sont basés sur trois choses:

- Actions
- Contenaires
- Modificateurs

Chacune des ces caractéristiques sont gérées par des “Managers” qui par introspections se chargent de récupérer, la liste des actions, des contenaires, des modificateurs.

Gestion des Actions

L' ActionsManager est capable de récupérer la liste des méhodes assimilées à des actions et est capable de les éxécuter.

Les actions d'un élément sont en fait des méthodes qui ont pour signature :

```
public type Action(<type> Source, <type> Destination)
```

Nota : les méthodes dont le nom commence par p_ sont des actions 'privé' de sorte qu'elles ne seront pas visible dans le PopUpMenu.

Gestion des Modificateurs

L' AttributsManager est capable de récuperer la liste des attributs assimilés à des modificateurs et de retourner leur valeur.

Pour l'implémentation des caractéristiques des personnages, il était nécessaire de se souvenir de la forme du calcul

Exemple:

Vitesse = 4 * Endurance

Problème, si l'endurance change, il faut changer la valeur de la vitesse en

conséquence.

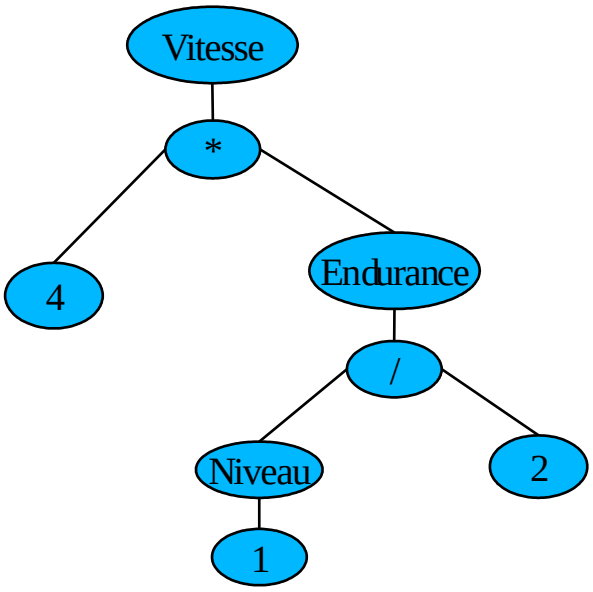
Pour se faire il a été implémenté un arbre d'expression.

Exemple:

Vitesse = 4 * Endurance

Endurance = Niveau / 2

Niveau = 1



Ainsi dès que le niveau est modifié la mise à jour se fait sur tous les autres modificateurs.

Implémentation en java

L'implémentation est fait avec une classe de base appellée modificateur qui est assimilé à un noeud,et qui permet de se chainer elle même. La classe chainée connait donc son successeur et son prédécesseur.

Chaque Classe Modificateur contient un lien vers un fils gauche et un fils droit.

La suppression et l'ajout de modificateur est donc rapide et facilement implémentable.

Pour récupérer la valeur final, il suffit de parcourir tout le liste.

Gestion des Contenaire

Le ContenairesManager est capable de récupérer la liste des contenaires assimilés à des Modificateurs et d'en retourner un, suivant son nom.

La classe contenaire permet de gérer tout ce que peut posséder un Element du jeu.

A la contrsuction on spécifie le type de classe à stocker dans le contenaire, ainsi que le nombre maximum stockable.

Nota: une taille de 0, équivaut à l'infini.

Exemple :

Une main gauche qui ne contient que des Classes dérivées de bouclier
Une main droite qui ne contient que des Classes dérivées d'arme
Une Armure qui ne contient que des classes dérivées de Vetement
Une Sacoche qui peut contenir que des Classes dérivées d'objet

implémentation en java

On utlise simplement une collection indexée.

Une méthode Add est implémentée, elle se charge de vérifier si l'objet est de la bonne classe.

Gestion du changement de salle

La contrainte première était un nombre de salle infinie.

Le changement de salle a donc été implémenté à l'aide d'une table de hachage ayant pour clef les coordonées de la salle et comme elements la salle :

Imaginons le chemin du joueur :

Droite, Haut, Bas, Gauche, Gauche, Gauche

			Salle3(1,1)
Salle5(- 2,0)	Salle4(- 1,0)	Salle1(0,0)	Salle2(1,0)

la Table de Hachage donnerait :

coordonées	***Salle***
“0,0”	Salle1
“1,0”	Salle2
“1,1”	Salle3
“- 1,0”	Salle4
“- 2,0”	Salle5

L'avantage de cette méthode, est que l'on ne stocke pas un tableau, et donc nous n'avons pas à réallouer le table à chaque dépassement de bornes du tableau.

Bug Connues

Le changement de profession n'a pu être implémenté, le temps a manqué, néanmoins, l'idée était de créer des constructeur par copie, ainsi, un classe guerrier aurait pu héritée de Aventurier. De même pour les sorts

Si un monstre change de salle, le joueur disparait.