Architecture utilisée :

Package « baba.enumerate » :

L 'interface « EnumWords » : Englobe tous les mots du jeu.

On a 3 Enum qui implementent cette interface :

- **« EnumNouns »** : Contient tous les noms possibles du jeu (Baba,Wall etc...).

- **« EnumOperators »** : Contient tous les operateurs possibles du jeu (que « is » dans cette version).

- **« EnumProperties »** : Contient toutes les propriétés possibles du jeu (You,Stop etc...).

Ils ont tous les 3 une methode qui permet de savoir si une chaine de caractere décrit un nom pour « EnumNouns », un operateur pour « EnumOperators » ou une propriété « EnumProperties ».

Package « baba» :

Representation des elements du jeu :

Classe « Element » : Sert à représenter un element du jeu, elle contient les informations concernant la position (positionx et positiony) ainsi que le « type » de l'élement (element « Baba », un bloc de texte « You » etc..) qui correspond à un « EnumWords ». Elle contient les methodes qui servent à savoir si un élément est au dessus d'un autre, est juste apres ou à la meme position qu'un autre élément.
.

Representation des propriétés du jeu :

Interface « Property » : Englobe toutes les propriétés du jeu.

Classe abstraite « AbstractProperty » : Elle implementel'interface « Property », elle possède toutes les methodes qu'une propriétés doit avoir. Dx et dy represente des directions.
L'ArrayList « elements » contient tous les élements qui ont la propriétés.

Cette classe est étendue par 9 classes qui sont :

- **Classe « You » :** Représente la propriété **«you »** et possede tous les éléments qui ont cette propriété. On y trouve l'ensemble des methodes qui permettent d'effectuer un **déplacement**.

- **Classe « Win » :** Représente la propriété «**win** » et possede tous les éléments qui ont cette propriété. On y trouve la methode qui permet de savoir **si les conditions de victoire sont remplisent**.

- **Classe « Stop » :** Représente la propriété «**stop**» et possede tous les éléments qui ont cette propriété. On y trouve la methode qui permet de savoir **si on peut traverser un element ou pas**.

- **Classe « Sink » :** Représente la propriété «**sink**» et possede tous les éléments qui ont cette propriété.On y trouve la methode qui **permet de retirer les éléments « Sink ».**

- **Classe « Push » :** Représente la propriété «**push**» et possede tous les éléments qui ont cette propriété. On y trouve l'ensemble des methodes qui **permettent la pousser ou non d'un élément du jeu**.

- **Classe « Hot»** : Représente la propriété «**hot**» et possede tous les éléments qui ont cette propriété.

- **Classe « Melt » :** Représente la propriété «**melt**» et possede tous les éléments qui ont cette propriété. On y trouve la methode qui **permet de retirer les éléments « melt » qui sont à la meme position que les elements « hot ».**

- **Classe «Defeat» :** Représente la propriété «**defeat**» et possede tous les éléments qui ont cette propriété.On y trouve la methode qui **permet de retirer les éléments «You» qui sont à la meme position que les elements «Defeat».**

- **Classe «Breaker» :** Représente la propriété «**breaker**» et possede tous les éléments qui ont cette propriété. Elle contient l'ensemble des methodes qui **permettent de détruire l 'élement qui est sur le passage d'un élément contenu dans « you »**.

Representation d'un niveau de jeu :

Classe abstraite « AbstractLevel » : Contient toutes les methodes necessaire à la creation d'un niveau.
Elle possede 4 hashmap :

- « **elements** » : Contient tous les elements du jeu sauf les blocs textes.Toute modification d'un element se fera par son intermediaire.

- « **texts** » : Ne contient que les blocs de textes. Elle va nous servir à generer les regles d'un niveau.

- « **properties** » : Contient les propriétés du jeu.Toute modification d'une propriété se fera par son intermediaire.

- « **elementsImage** » : Contient tous les elements du jeu. Elle sert à afficher pour chaque image les elements qui lui correspondent.

Classe « Level » : Contient les methodes qui permettent de **gerer le parametrage de niveau à l'aide des commandes**, ainsi que de **generer les propriétés par defaut du niveau** (par exemple associé You à un élement si cela n'est pas parametré avec les commandes).

Amelioration et correction apportées apres la soutenance :

- Refonte complete du programme.
- Changement de la representation des elements par plusieurs classes en une seule classe.
- Ajout d'un 3eme parametre (type) dans la classe « Element » qui permet de differencié les elements entre eux.
- Utilisation d'une hashmap qui contient tous les elements au lieu d'avoir pour chaque element son ArrayList.
- Suppression de l'interface « Elements » et de la classe abstraite « AbstractElement ».
- Changement dans la representation des blocs de texte, suppression de la classe « Text » et creation d'une hashmap qui regroupe tous les blocs de textes.
- Changement dans la façon de generer les regles du jeu.
- Suppression de l'interface « Image » et de la classe « DrawImages », remplacement par une HashMap « elementsimage » qui contient toutes les images ainsi que les elements qui leurs sont associés, ainsi qu'une fonction « drawLevel » qui parcours cette hashmap et affiche ses elements.
- Generation des differents niveaux à l'aide de fichier textes, au lieu d'avoir une classe pour chaque niveau.
- Changement de la HashMap « elementsimage » en LinkedHashMap pour afficher les elements du decor en premier et ainsi ne pas avoir de probleme d'affichage (Baba derriere un element du decor par exemple)