Explications du diagramme de classe:

Package model :

- ### Enum GameState :

  L'énumération GameState permet de représenter les différents états du jeu.
  Elle peut comporter les valeurs suivantes :
  PLAY: L'état du jeu lorsqu'il est en cours de partie.
  WIN: L'état du jeu lorsqu'il est remporté par le joueur.
  LOSE: L'état du jeu lorsqu'il est perdu par le joueur.

- ### Enum Direction :

  L'énumération Direction permet de définir des directions possibles pour les objets dans le plateau de jeu. Elle contient cinq constantes: UP, RIGHT, LEFT, DOWN et NONE. Chaque constante est associée à un offset de déplacement représenté par les attributs privés deltaRow et deltaColumn. La méthode constructeur Direction prend deux paramètres, deltaRow et deltaColumn, pour initialiser les attributs de la constante. Elle a également deux méthodes publiques: getDeltaRow() qui retourne la valeur de l'attribut deltaRow, et getDeltaColumn() qui retourne la valeur de l'attribut deltaColumn. Ces méthodes permettent de récupérer les valeurs de déplacement associées à chaque constante.

- ### Enum Subject :

  L'énumération Subject est utilisée pour représenter différents types d'objets dans un jeu. Elle contient sept constantes: baba, lava, rock, flag, wall, bone et goop. Chaque constante représente un type d'objet spécifique dans le jeu. Par exemple, la constante baba peut représenter le personnage jouable du jeu, la constante lava peut représenter un obstacle dangereux, la constante rock peut représenter un objet que le joueur doit déplacer, la constante flag peut représenter un objectif à atteindre, la constante wall peut représenter un obstacle infranchissable, la constante bone peut représenter un ennemi et la constante goop peut représenter un obstacle destructible. Cette énumération permet de définir des

constantes faciles à utiliser pour représenter différents types d'objets dans le jeu.

- ## Enum Operator:
  L’énumération Operator contient une constante IS qui sert à etre un connecteur logique entre l’enum subject et l’enum complement pour former une règle.

- ## Enum Complement:
  L'énumération Complement est utilisée pour représenter les compléments associés aux différents types d'objets dans le jeu. Elle contient cinq constantes: kill, win, stop, push et you. Chaque constante représente un complément spécifique qui peut être associé à un type d'objet. Par exemple, la constante kill peut être associée à la constante lava pour indiquer que le joueur meurt s'il entre en contact avec la lave. La constante win peut être associée à la constante flag pour indiquer que le joueur gagne s'il atteint le drapeau. La constante stop peut être associée à la constante wall pour indiquer que les objets ne peuvent pas passer à travers le mur. La constante push peut être associée à la constante rock pour indiquer que le joueur peut pousser les rochers. Enfin, la constante you peut être associée à la constante baba pour indiquer que le joueur contrôle le personnage principal du jeu. Cette énumération permet de définir facilement les différents compléments qui peuvent être associés aux objets dans le jeu.

- ## Enum Icon:
  L'énumération Icon est utilisée pour représenter les différentes icônes associées aux objets du jeu. Elle contient onze constantes: baba_icon, rock_icon, flag_icon, wall_icon, metal_icon, bone_icon, grass_icon, empty_icon, lava_icon, goop_icon et water_icon. Chaque constante représente une icône spécifique qui peut être associée à un objet du jeu. Par exemple, la constante baba_icon peut être associée au personnage jouable, la constante rock_icon peut être associée aux rochers, la constante flag_icon peut être associée aux drapeaux, la constante wall_icon peut être associée aux murs, la constante metal_icon peut être associée aux objets métalliques, la constante bone_icon peut être associée aux ennemis, la constante grass_icon peut être associée à l’herbe, la constante empty_icon peut être

associée à un emplacement vide, la constante lava_icon peut être associée à la lave, la constante goop_icon peut être associée aux obstacles destructible et la constante water_icon peut être associée à l'eau. Cette énumération permet de définir facilement les différentes icônes qui peuvent être associées aux objets du jeu.

- ## Enum ObjectType:

  L'énumération ObjectType est utilisée pour représenter les différents types d'objets utilisés dans le jeu. Elle contient quatre constantes: subject, operator, complement et icon. Chaque constante représente un type d'objet spécifique dans le jeu. La constante subject représente les objets du jeu qui peuvent être contrôlés par le joueur, comme le personnage jouable ou les rochers. La constante operator représente les opérateurs du jeu, comme les verbes "est".La constante complement représente les compléments associés aux objets du jeu, comme "kill" ou "stop". Enfin, la constante icon représente les icônes utilisées pour représenter les objets dans l'interface graphique du jeu. Cette énumération permet de classer les différents types d'objets utilisés dans le jeu.

- ## Classe Position :

  La classe Position est utilisée pour stocker la position d'un objet dans un plateau de jeu. Elle a deux attributs: row, qui représente le numéro de ligne, et column, qui représente le numéro de colonne. La classe a une méthode constructeur Position qui prend deux paramètres, row et column, pour initialiser les attributs de la classe. Elle a également deux méthodes publiques: getRow() qui retourne la valeur de l'attribut row, et getColumn() qui retourne la valeur de l'attribut column. Enfin, elle a une méthode next() qui prend en paramètre une direction (Direction) et qui renvoie une nouvelle instance de Position qui correspond à la position suivante dans cette direction.

- ## Classe Rule:

  La classe Rule représente une règle du jeu composée d'un subject, d'un operator et d'un complement. Elle contient un constructeur Rule() qui prend en paramètre un objet de type Subject, un objet de type Operator et un objet de type Complement pour initialiser une

nouvelle instance de la règle. Cette méthode permet de créer facilement une nouvelle règle avec les objets spécifiés en paramètres. Par exemple, pour créer une règle qui dit que le personnage jouable est contrôlé par le joueur, on peut construire une règle grâce au constructeur Rule() avec l'objet Subject « baba », l'objet Operator « is » et l'objet Complement « you ». Cette règle sera ensuite utilisée pour déterminer le comportement du jeu.

Classe Tiles:

La classe Tiles représente une case dans le plateau de jeu. Cette case peut être constituée d'une ou plusieurs constantes parmi les quatre constantes de ObjectType: subject, operator, complement et icon. Elle contient un attribut listObject de type std::vector<ObjectType> qui permet de stocker les constantes associées à la case. La classe contient également une méthode Tiles() qui prend en paramètre un vecteur d'ObjectType pour initialiser une nouvelle instance de la case. La méthode getSizeList() permet de récupérer le nombre d'éléments présents dans la liste d'objets, tandis que la méthode getListObject() permet de récupérer la liste d'objets associés à la case. Cette classe permet de stocker les informations relatives à une case du plateau de jeu, ce qui est utile pour déterminer les règles applicables dans le jeu.

Classe Board:

La classe Board permet de déterminer le plateau de jeu. Elle contient les attributs board, player et rules qui représentent respectivement au plateau de jeu, le joueur et les règles du jeu.
Le plateau de jeu est représenté par un vecteur de vecteurs de type Tiles qui stocke les informations relatives à chaque case du plateau.
Le joueur est représenté par un objet de type ObjectType qui correspond à l'objet contrôlé par le joueur. Les règles du jeu sont stockées dans un vecteur de type Rule.
La classe Board contient plusieurs méthodes pour interagir avec le plateau de jeu. La méthode Board() permet de créer une nouvelle instance du plateau de jeu avec une taille spécifiée. La méthode getBoard() permet de récupérer le plateau de jeu. La méthode move() permet de déplacer le joueur dans une certaine direction. La méthode isValidMove() permet de vérifier si un mouvement est valide. La méthode dropTile() permet de supprimer une case du plateau de jeu.

La méthode setTile() permet de modifier une case du plateau de jeu en y ajoutant une nouvelle constante. La méthode addRules() permet d'ajouter des règles au jeu. La méthode removeRule() permet de supprimer une règle du jeu. La méthode isBrokeRule() permet de vérifier si une règle a été enfreinte. La méthode contains() permet de vérifier si une certaine position est présente sur le plateau de jeu. La méthode push() permet de déplacer un objet sur une case. Les méthodes isStop(), isWin(), isPush(), isKill() et isYou() permettent de vérifier si une certaine condition est remplie pour un objet donné. Enfin, la méthode getPlayer() permet de récupérer le joueur actuel. Cette classe permet de gérer le plateau de jeu, les règles et les interactions entre les différents éléments du jeu.

- ## Classe Level:

  La classe Level représente un niveau du jeu. Elle contient un attribut board de type Board, qui correspond au plateau de jeu du niveau.
  La classe Level possède plusieurs méthodes. La méthode getBoard() permet de récupérer le plateau de jeu associé au niveau. La méthode setBoard() permet de modifier le plateau de jeu associé au niveau.
  En résumé, la classe Level permet de stocker et de manipuler un plateau de jeu spécifique pour un niveau donné, ce qui permet de créer différents niveaux de difficulté pour le jeu.

- ## Classe LevelLoader:

  La classe LevelLoader permet de charger un niveau à partir d'un fichier texte qui décrit le plateau de jeu et les règles associées.
  La classe LevelLoader possède un attribut level de type Level, qui correspond au niveau chargé à partir du fichier texte.
  La méthode loadLevel(int numlevel) permet de charger le niveau correspondant au numéro numlevel à partir du fichier texte. Cette méthode lit le fichier texte et crée une instance de la classe Board à partir des informations fournies. Elle ajoute ensuite le Board créé à une instance de la classe Level.
  La méthode getLevel() permet de récupérer le niveau chargé à partir du fichier texte.
  En résumé, la classe LevelLoader permet de charger un niveau à partir d'un fichier texte et de le stocker dans une instance de la classe Level.

Classe ManagerCommand:

La classe ManagerCommand permet de gérer l'historique des commandes pour pouvoir les annuler ou les rétablir.
Cette classe utilise un attribut undoStack: une pile pour stocker les commandes qui ont été exécutées et qui peuvent être annulées et un autre attribut redoStack: une pile pour stocker les commandes qui ont été annulées et qui peuvent être rétablies.
La méthode addAllCommand(Command command): ajoute une commande à la pile undoStack pour la sauvegarder, la méthode undo(): annule la dernière commande en retirant la dernière commande de la pile undoStack et en l'ajoutant à la pile redoStack, et la méthode redo(): rétablit la dernière commande annulée en retirant la dernière commande de la pile redoStack et en l'exécutant. La commande sera ajoutée à la pile undoStack pour pouvoir être annulée à nouveau si besoin.

Classe BabaIsYou:

La classe BabaIsYou est une façade qui permet de simplifier l'interaction entre le joueur et le jeu.
Elle possède un attribut numLevel de type int qui permet de connaître le niveau courant, un attribut observers de type vector<Observer> qui permet de stocker les observateurs du jeu, et un attribut levelLoader de type LevelLoader qui permet de charger les niveaux à partir de fichiers textes et un attribut managerCommand, qui est une instance de la classe ManagerCommand qui permet de gérer les commandes undo et redo.

La méthode start(in numLevel:int) permet de lancer le jeu en chargeant le niveau correspondant au numéro numLevel à partir du fichier texte correspondant. Elle crée une instance de la classe LevelLoader avec le fichier texte correspondant et l'associe à l'attribut levelLoader.
La méthode getNumLevel() permet de récupérer le numéro du niveau courant. La méthode setNumLevel() permet de modifier le numéro du niveau courant.

La méthode BabaIsYou(in level:LevelLoader) permet de créer une instance de la classe BabaIsYou à partir d'une instance de la classe LevelLoader.
La méthode getLevel() permet de récupérer le niveau courant chargé à partir du fichier texte.
Les méthodes undo() et redo() permettent d'annuler ou de rétablir le dernier coup joué.
Les méthodes addObserver(in Observer observer), removeObserver(in Observer observer) et notifyObservers() permettent de gérer les observateurs du jeu.
En résumé, la classe BabaIsYou permet de lancer le jeu, d'annuler ou rétablir des coups, de gérer les observateurs et de récupérer des informations sur le niveau courant.