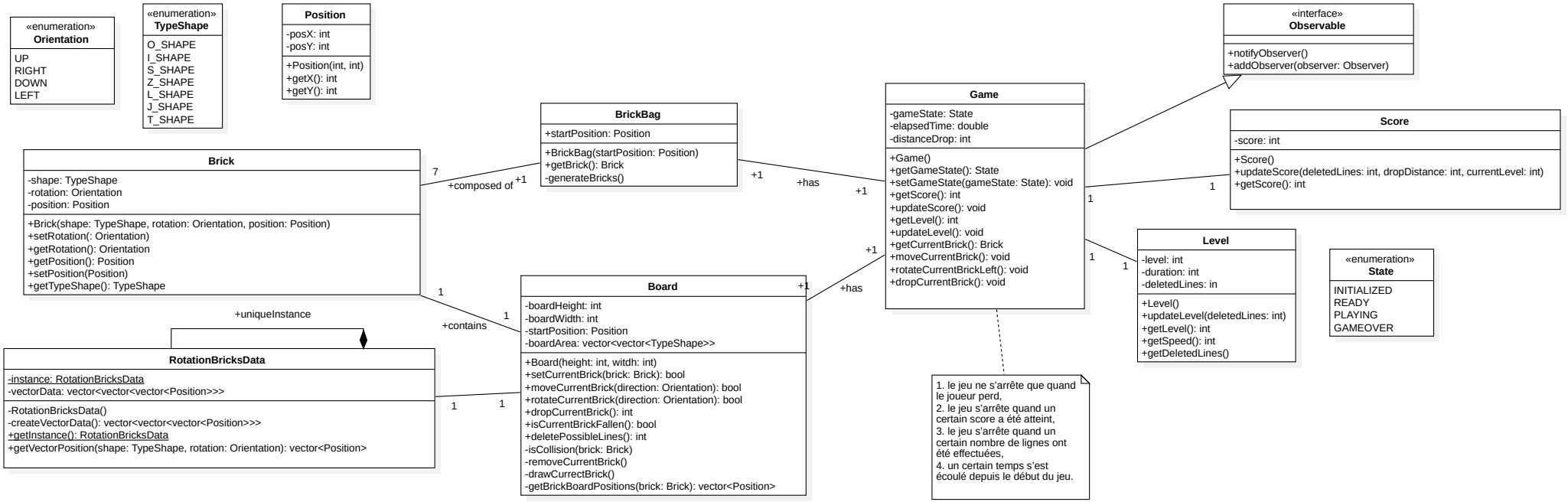
# Model::Main


«enumeration»
Orientation
UP
RIGHT
DOWN
LEFT
«enumeration»
TypeShape
O_SHAPE
I_SHAPE
S_SHAPE
Z_SHAPE
L_SHAPE
J_SHAPE
T_SHAPE
Position
-posX: int
-posY: int
+Position(int, int)
+getX(): int
+getY(): int
«interface»
Observable
+notifyObserver()
+addObserver(observer: Observer)
BrickBag
+startPosition: Position
+BrickBag(startPosition: Position)
+getBrick(): Brick
-generateBricks()
Game
-gameState: State
-elapsedTime: double
-distanceDrop: int
+Game()
+getGameState(): State
+setGameState(gameState: State): void
+getScore(): int
+updateScore(): void
+getLevel(): int
+updateLevel(): void
+getCurrentBrick(): Brick
+moveCurrentBrick(): void
+rotateCurrentBrickLeft(): void
+dropCurrentBrick(): void
Score
-score: int
+Score()
+updateScore(deletedLines: int, dropDistance: int, currentLevel: int)
+getScore(): int
Brick
-shape: TypeShape
-rotation: Orientation
-position: Position
+Brick(shape: TypeShape, rotation: Orientation, position: Position)
+setRotation(: Orientation)
+getRotation(): Orientation
+getPosition(): Position
+setPosition(Position)
+getTypeShape(): TypeShape
7
+composed of
+1
+1
+has
+1
1
1
1
Level
-level: int
-duration: int
-deletedLines: in
+Level()
+updateLevel(deletedLines: int)
+getLevel(): int
+getSpeed(): int
+getDeletedLines()
«enumeration»
State
INITIALIZED
READY
PLAYING
GAMEOVER
+1
+has
+1
1
+contains
1
+uniqueInstance
Board
-boardHeight: int
-boardWidth: int
-startPosition: Position
-boardArea: vector<vector<TypeShape>>
+Board(height: int, witdh: int)
+setCurrentBrick(brick: Brick): bool
+moveCurrentBrick(direction: Orientation): bool
+rotateCurrentBrick(direction: Orientation): bool
+dropCurrentBrick(): int
+isCurrentBrickFallen(): bool
+deletePossibleLines(): int
-isCollision(brick: Brick)
-removeCurrentBrick()
-drawCurrectBrick()
-getBrickBoardPositions(brick: Brick): vector<Position>
RotationBricksData
-instance: RotationBricksData
-vectorData: vector<vector<vector<Position>>>
-RotationBricksData()
-createVectorData(): vector<vector<vector<Position>>>
+getInstance(): RotationBricksData
+getVectorPosition(shape: TypeShape, rotation: Orientation): vector<Position>
1
1
1. le jeu ne s'arrête que quand le joueur perd,
2. le jeu s'arrête quand un certain score a été atteint,
3. le jeu s'arrête quand un certain nombre de lignes ont été effectuées,
4. un certain temps s'est écoulé depuis le début du jeu.