

ESCAPE!

Connor Guerrieri
Adam Holt
Brent Pivnik
Joey Rener

What is it?

- Puzzle game for Android
- Player navigates grid filled with traps and obstacles to reach next level!



Factory Method Pattern

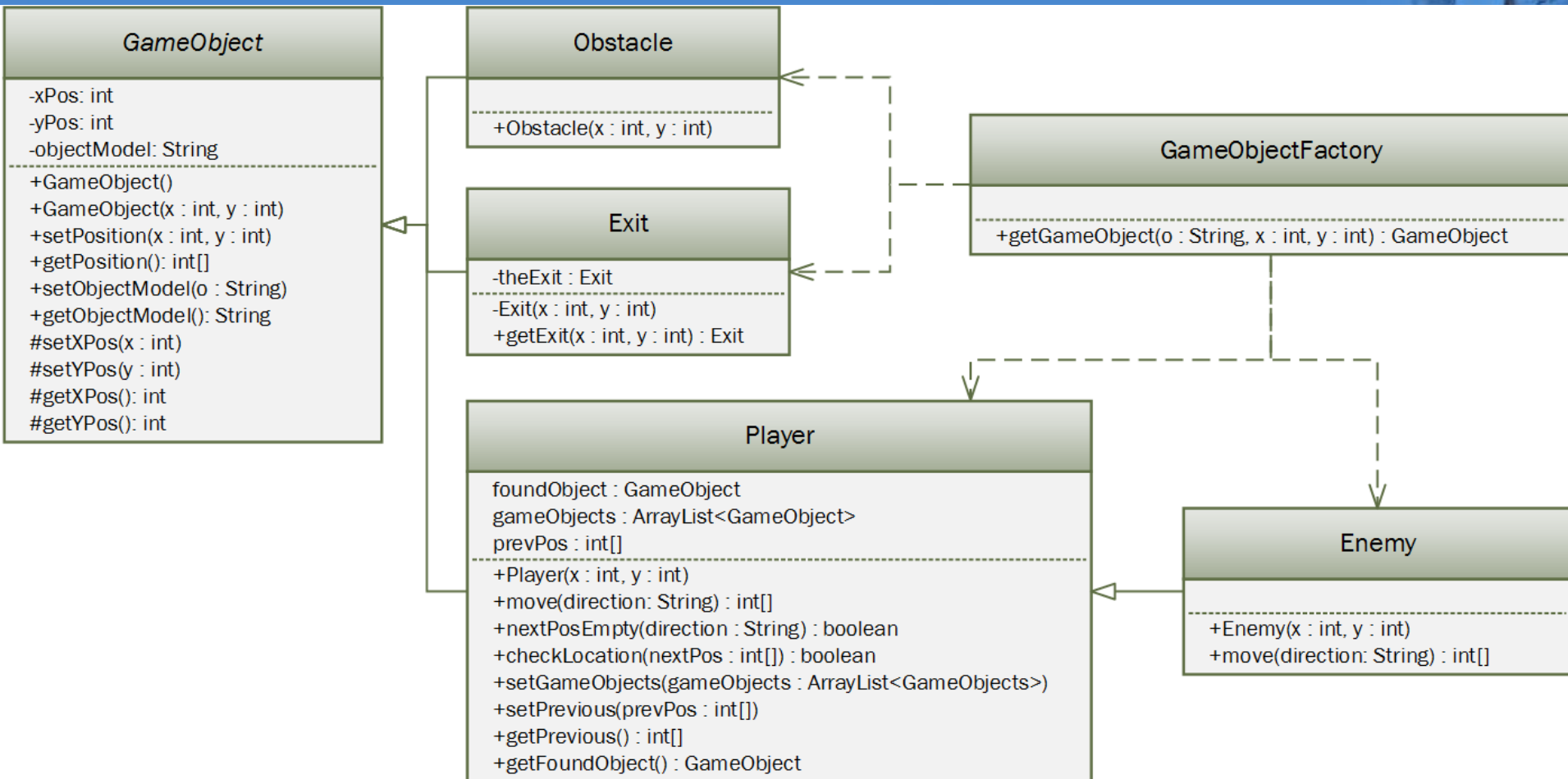
GameObjectFactory

Generates polymorphic GameObjects based on type of GameObject and x,y coordinates

LevelFactory

Creates Level object based on placement of GameObject instances

Factory Method Pattern



Singleton Pattern

- Used when we want to restrict instantiation of a class to one instance of an object
- We only wanted one Exit and one GameBoard in the game
- Allows game to execute around the methods of one object instead of multiple classes all calling/instantiating back and forth

Singleton Pattern

GameBoard

```
-board : GameBoard
-gameObjects : ArrayList<GameObject>
-player : Player
-levelNumber : int
-----
-GameBoard()
-buildNextLevel()
-buildGameBoard(objectModels : String[], xCoords : int[], yCoords : int[])
+getGameBoard() : GameBoard
+moveUp(view : View)
+moveRight(view : View)
+moveLeft(view : View)
+moveDown(view : View)
+move(newPos : int[])
+changeImage(coord : int[], color : String)
+loseGame()
+winGame()
```

Exit

```
-theExit : Exit
-----
-Exit(x : int, y : int)
+getExit(x : int, y : int) : Exit
```

What Did We Learn?

- The importance of using diagrams
- The importance of having a well constructed back-end
- The utility of design patterns

Game Demo

[Link to Demo Video](#)

