# Casino Simulator Refactor

Paige Johnson, Derek Gorthy, Michael Condon

April 6, 2017

# Original Class Diagram
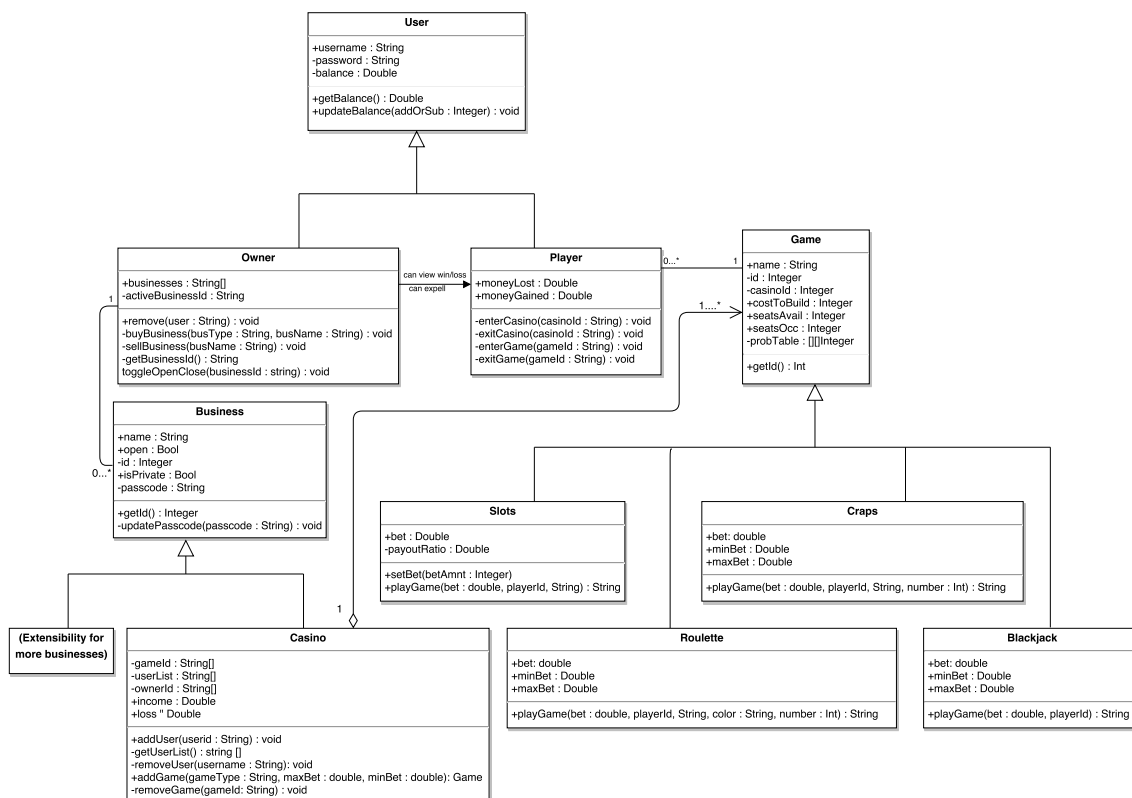
Figure 1: Class Diagram

# Refactored Class Diagram

## 2.1 Modifications

### 2.1.1 Model View Controller

The feedback that we had on our original class diagram was that we didnt have a controller. In order to fix that we created both a controller class, UI classes, and an input handler. These classes, along with User and Business as the "Model Classes" use the MVC design pattern.

### 2.1.2 Decorator Pattern

We realized that having a different subclass for every single game would get extremely cluttered if we wanted to add more games. We decided to fix this by using the Decorator Design Pattern. This way we can dynamically assign behaviors to each game depending on what we want to load.
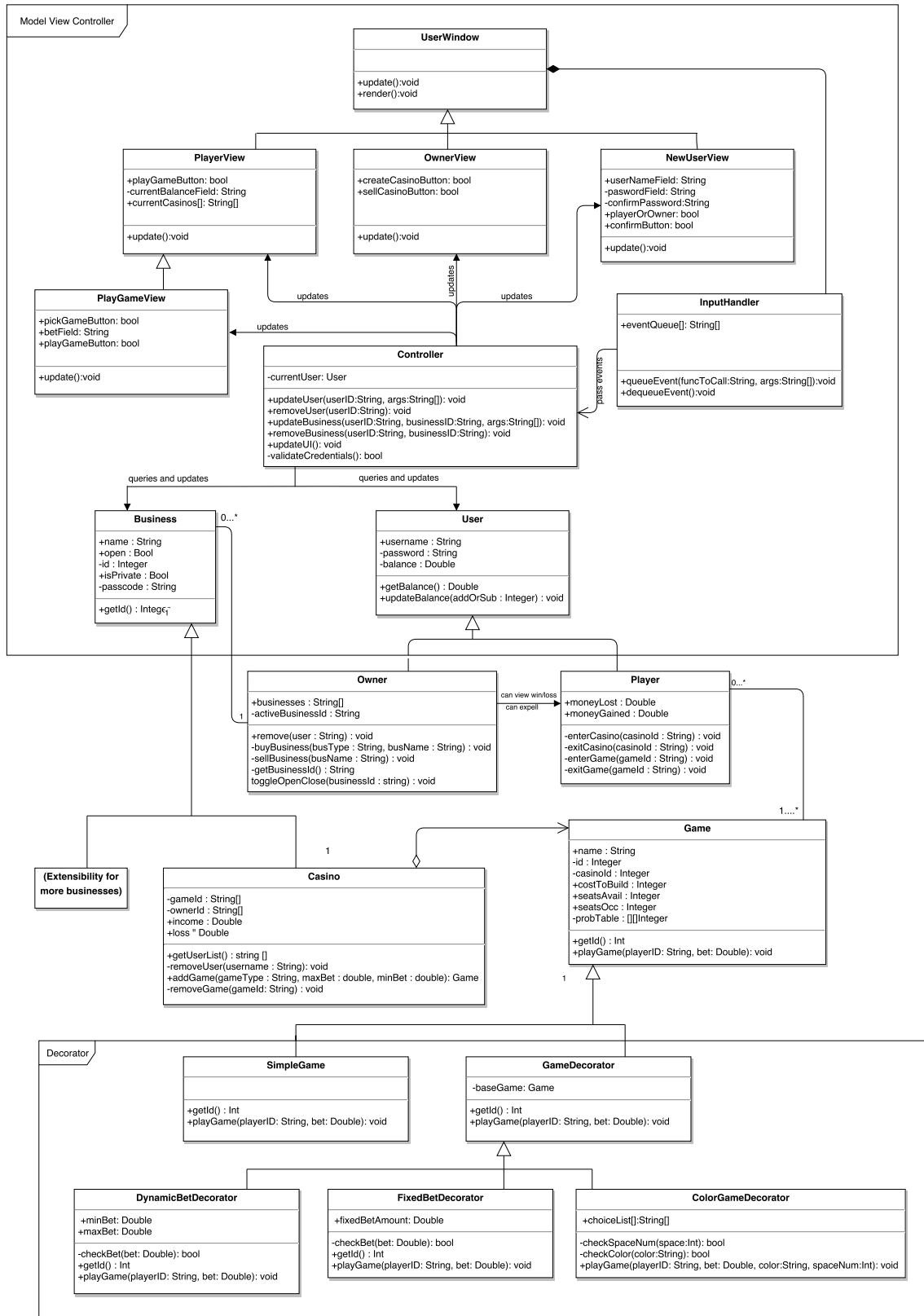
## 2.2 Diagram

**Model View Controller**

**UserWindow**
+update():void
+render():void

**PlayerView**
+playGameButton: bool
-currentBalanceField: String
+currentCasinos[]: String[]
+update():void

**OwnerView**
+createCasinoButton: bool
+sellCasinoButton: bool
+update():void

**NewUserView**
+userNameField: String
-paswordField: String
-confirmPassword:String
+playerOrOwner: bool
+confirmButton: bool
+update():void

**PlayGameView**
+pickGameButton: bool
+betField: String
+playGameButton: bool
+update():void

**InputHandler**
+eventQueue[]: String[]
+queueEvent(funcToCall:String, args:String[]):void
+dequeueEvent():void

updates    updates    updates    updates    updates

pass events

**Controller**
-currentUser: User
+updateUser(userID:String, args:String[]): void
+removeUser(userID:String): void
+updateBusiness(userID:String, businessID:String, args:String[]): void
+removeBusiness(userID:String, businessID:String): void
+updateUI(): void
-validateCredentials(): bool

queries and updates            queries and updates

**Business**
+name : String
+open : Bool
-id : Integer
+isPrivate : Bool
-passcode : String
+getId() : Integer

0...*

**User**
+username : String
+password : String
-balance : Double
+getBalance() : Double
+updateBalance(addOrSub : Integer) : void

**Owner**
+businesses : String[]
-activeBusinessId : String
+remove(user : String) : void
-buyBusiness(busType : String, busName : String) : void
-sellBusiness(busName : String) : void
-getBusinessId() : String
toggleOpenClose(businessId : string) : void

can view win/loss
can expell

**Player**
+moneyLost : Double
+moneyGained : Double
-enterCasino(casinoId : String) : void
-exitCasino(casinoId : String) : void
-enterGame(gameId : String) : void
-exitGame(gameId : String) : void

0...*

1

**(Extensibility for more businesses)**

**Casino**
-gameId : String[]
-ownerId : String[]
+income : Double
+loss " Double
+getUserList() : string []
-removeUser(username : String): void
+addGame(gameType : String, maxBet : double, minBet : double): Game
-removeGame(gameId: String) : void

1

**Game**
+name : String
-id : Integer
-casinoId : Integer
+costToBuild : Integer
+seatsAvail : Integer
+seatsOcc : Integer
-probTable : [][]Integer
+getId() : Int
+playGame(playerID: String, bet: Double): void

1....*

1

**Decorator**

**SimpleGame**
+getId() : Int
+playGame(playerID: String, bet: Double): void

**GameDecorator**
-baseGame: Game
+getId() : Int
+playGame(playerID: String, bet: Double): void

**DynamicBetDecorator**
+minBet: Double
+maxBet: Double
-checkBet(bet: Double): bool
+getId() : Int
+playGame(playerID: String, bet: Double): void

**FixedBetDecorator**
+fixedBetAmount: Double
-checkBet(bet: Double): bool
+getId() : Int
+playGame(playerID: String, bet: Double): void

**ColorGameDecorator**
+choiceList[]:String[]
-checkSpaceNum(space:Int): bool
-checkColor(color:String): bool
+playGame(playerID: String, bet: Double, color:String, spaceNum:Int): void

Figure 2: Class Diagram