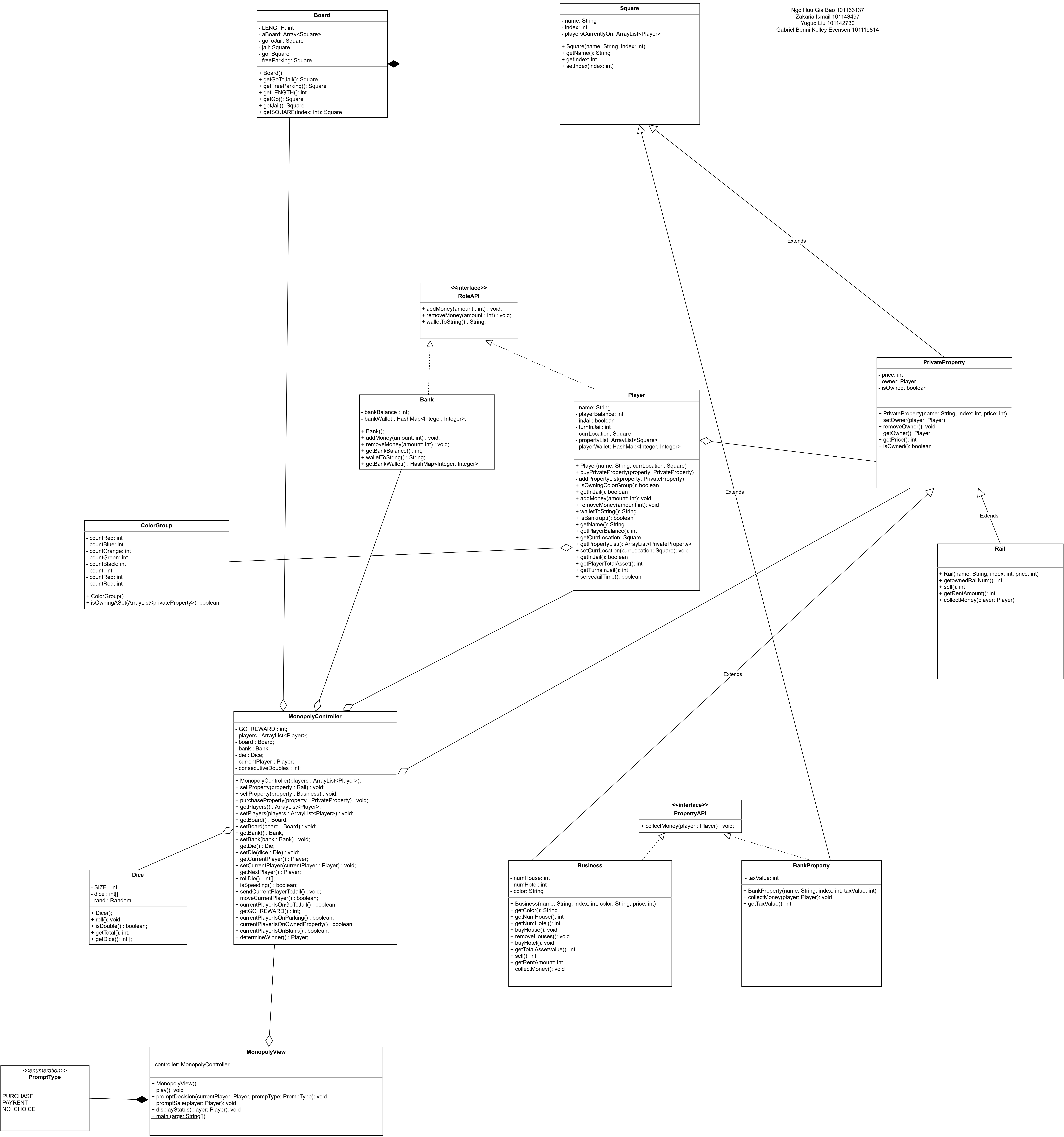
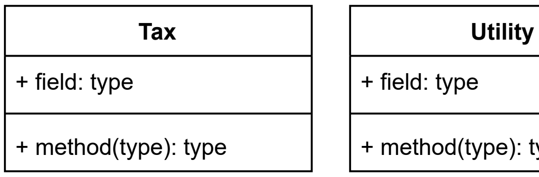
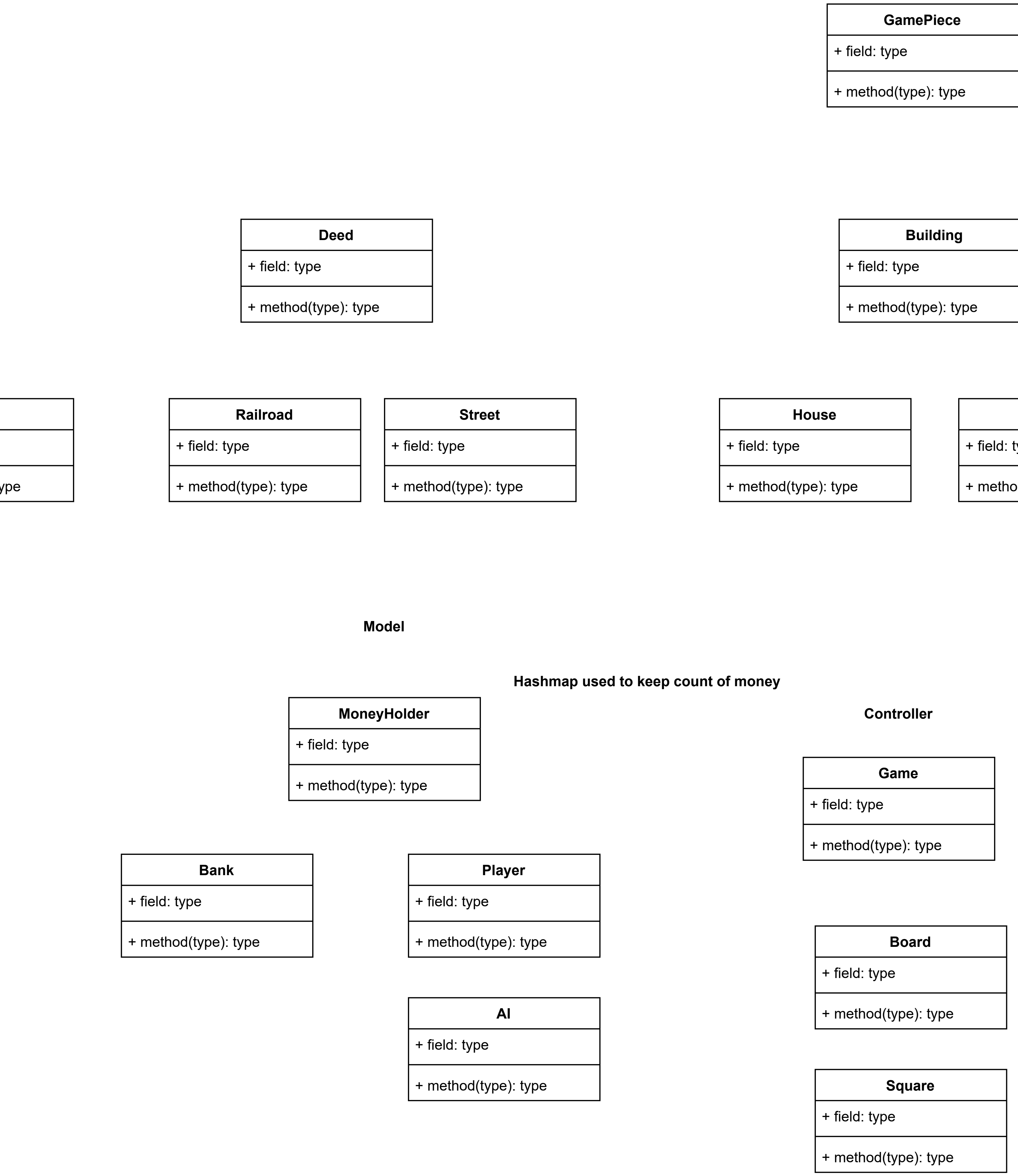
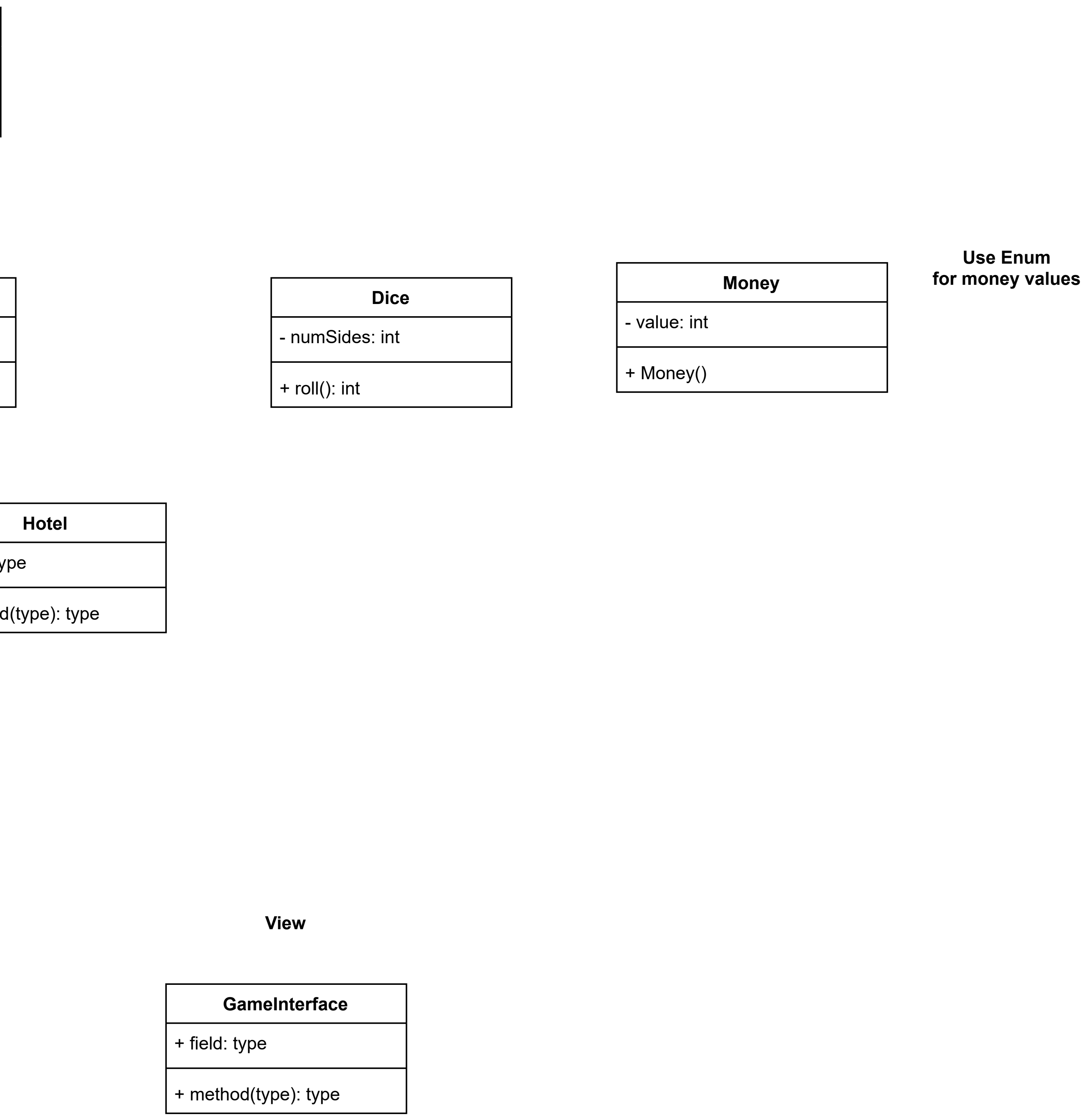
Ngo Huu Gia Bao 101163137
Zakaria Ismail 101143497
Yuguo Liu 101142730
Gabriel Benni Kelley Evensen 101119814


Board
- LENGTH: int
- aBoard: Array<Square>
- goToJail: Square
- jail: Square
- go: Square
- freeParking: Square
+ Board()
+ getGoToJail(): Square
+ getFreeParking(): Square
+ getLENGTH(): int
+ getGo(): Square
+ getJail(): Square
+ getSQUARE(index: int): Square
Square
- name: String
- index: int
- playersCurrentlyOn: ArrayList<Player>
+ Square(name: String, index: int)
+ getName(): String
+ getIndex: int
+ setIndex(index: int)
Extends
<<interface>>
RoleAPI
+ addMoney(amount : int) : void;
+ removeMoney(amount : int) : void;
+ walletToString() : String;
PrivateProperty
- price: int
- owner: Player
- isOwned: boolean
+ PrivateProperty(name: String, index: int, price: int)
+ setOwner(player: Player)
+ removeOwner(): void
+ getOwner(): Player
+ getPrice(): int
+ isOwned(): boolean
Bank
- bankBalance : int;
- bankWallet : HashMap<Integer, Integer>;
+ Bank();
+ addMoney(amount: int) : void;
+ removeMoney(amount: int) : void;
+ getBankBalance() : int;
+ walletToString() : String;
+ getBankWallet() : HashMap<Integer, Integer>;
Player
- name: String
- playerBalance: int
- inJail: boolean
- turnInJail: int
- currLocation: Square
- propertyList: ArrayList<Square>
- playerWallet: HashMap<Integer, Integer>
+ Player(name: String, currLocation: Square)
+ buyPrivateProperty(property: PrivateProperty)
- addPropertyList(property: PrivateProperty)
+ isOwningColorGroup(): boolean
+ getInJail(): boolean
+ addMoney(amount: int): void
+ removeMoney(amount int): void
+ walletToString(): String
+ isBankrupt(): boolean
+ getName(): String
+ getPlayerBalance(): int
+ getCurrLocation: Square
+ getPropertyList(): ArrayList<PrivateProperty>
+ setCurrLocation(currLocation: Square): void
+ getInJail(): boolean
+ getPlayerTotalAsset(): int
+ getTurnsInJail(): int
+ serveJailTime(): boolean
Extends
Extends
ColorGroup
- countRed: int
- countBlue: int
- countOrange: int
- countGreen: int
- countBlack: int
- count: int
- countRed: int
- countRed: int
+ ColorGroup()
+ isOwningASet(ArrayList<privateProperty>): boolean
Rail
+ Rail(name: String, index: int, price: int)
+ getownedRailNum(): int
+ sell(): int
+ getRentAmount(): int
+ collectMoney(player: Player)
Extends
MonopolyController
- GO_REWARD : int;
- players : ArrayList<Player>;
- board : Board;
- bank : Bank;
- die : Dice;
- currentPlayer : Player;
- consecutiveDoubles : int;
+ MonopolyController(players : ArrayList<Player>);
+ sellProperty(property : Rail) : void;
+ sellProperty(property : Business) : void;
+ purchaseProperty(property : PrivateProperty) : void;
+ getPlayers() : ArrayList<Player>;
+ setPlayers(players : ArrayList<Player>) : void;
+ getBoard() : Board;
+ setBoard(board : Board) : void;
+ getBank() : Bank;
+ setBank(bank : Bank) : void;
+ getDie() : Die;
+ setDie(dice : Die) : void;
+ getCurrentPlayer() : Player;
+ setCurrentPlayer(currentPlayer : Player) : void;
+ getNextPlayer() : Player;
+ rollDie() : int[];
+ isSpeeding() : boolean;
+ sendCurrentPlayerToJail() : void;
+ moveCurrentPlayer() : boolean;
+ currentPlayerIsOnGoToJail() : boolean;
+ getGO_REWARD() : int;
+ currentPlayerIsOnParking() : boolean;
+ currentPlayerIsOnOwnedProperty() : boolean;
+ currentPlayerIsOnBlank() : boolean;
+ determineWinner() : Player;
<<interface>>
PropertyAPI
+ collectMoney(player : Player) : void;
Dice
- SIZE : int;
- dice : int[];
- rand : Random;
+ Dice();
+ roll(): void
+ isDouble() : boolean;
+ getTotal(): int;
+ getDice(): int[];
Business
- numHouse: int
- numHotel: int
- color: String
+ Business(name: String, index: int, color: String, price: int)
+ getColor(): String
+ getNumHouse(): int
+ getNumHotel(): int
+ buyHouse(): void
+ removeHouses(): void
+ buyHotel(): void
+ getTotalAssetValue(): int
+ sell(): int
+ getRentAmount: int
+ collectMoney(): void
BankProperty
- taxValue: int
+ BankProperty(name: String, index: int, taxValue: int)
+ collectMoney(player: Player): void
+ getTaxValue(): int
MonopolyView
- controller: MonopolyController
+ MonopolyView()
+ play(): void
+ promptDecision(currentPlayer: Player, prompType: prompType): void
+ promptSale(player: Player): void
+ displayStatus(player: Player): void
+ main (args: String[])
<<enumeration>>
PromptType
PURCHASE
PAYRENT
NO_CHOICE

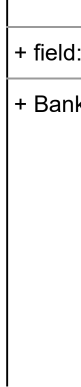

Tax
+ field: type
+ method(type): type
Utility
+ field: type
+ method(type): ty


GamePiece
+ field: type
+ method(type): type
Deed
+ field: type
+ method(type): type
Building
+ field: type
+ method(type): type
ype
Railroad
+ field: type
+ method(type): type
Street
+ field: type
+ method(type): type
House
+ field: type
+ method(type): type
+ field: ty
+ metho
Model
Hashmap used to keep count of money
MoneyHolder
+ field: type
+ method(type): type
Controller
Game
+ field: type
+ method(type): type
Bank
+ field: type
+ method(type): type
Player
+ field: type
+ method(type): type
Board
+ field: type
+ method(type): type
AI
+ field: type
+ method(type): type
Square
+ field: type
+ method(type): type

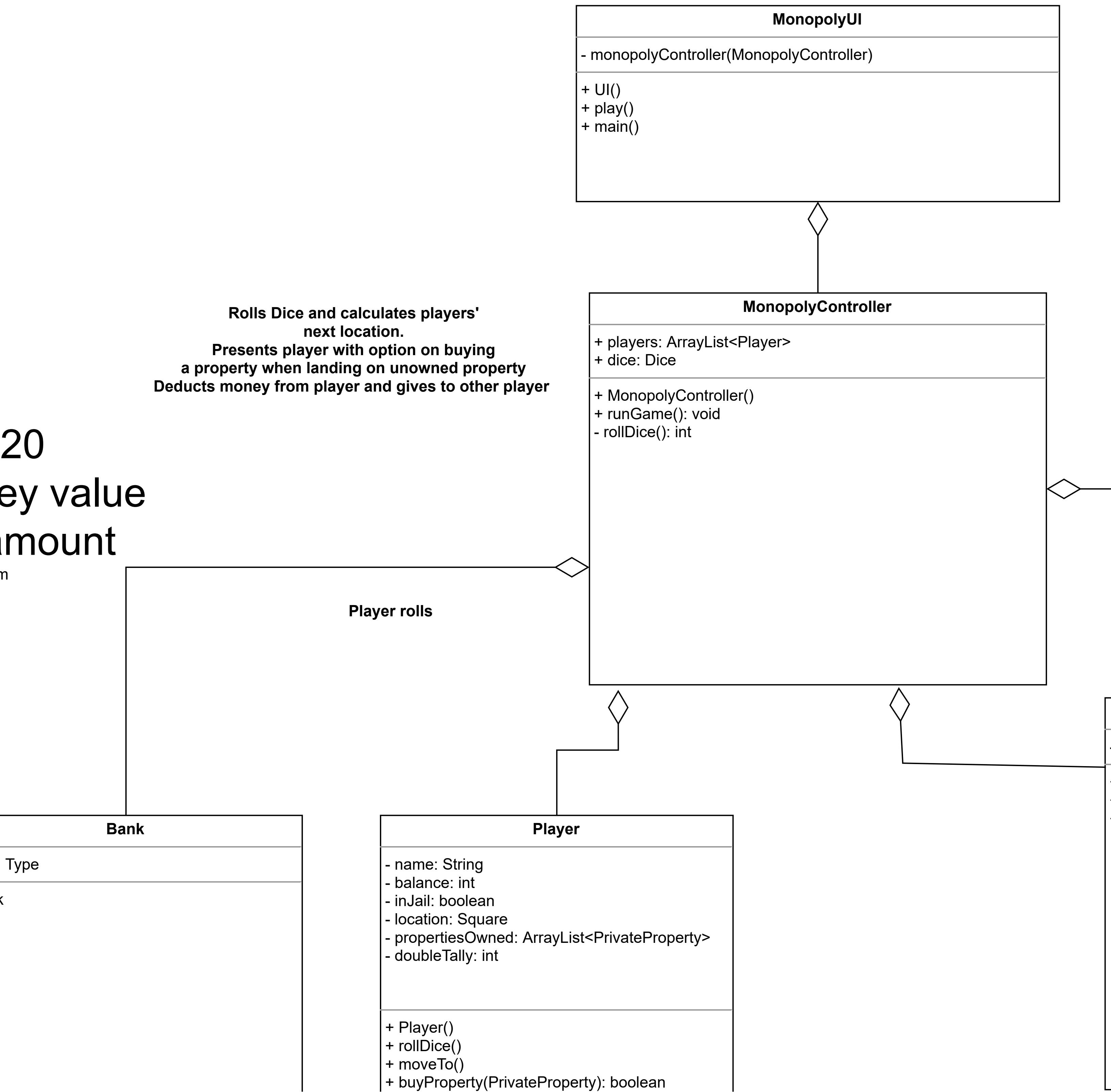

Dice
- numSides: int
+ roll(): int
Money
- value: int
+ Money()
Use Enum
for money values
Hotel
ype
d(type): type
View
GameInterface
+ field: type
+ method(type): type

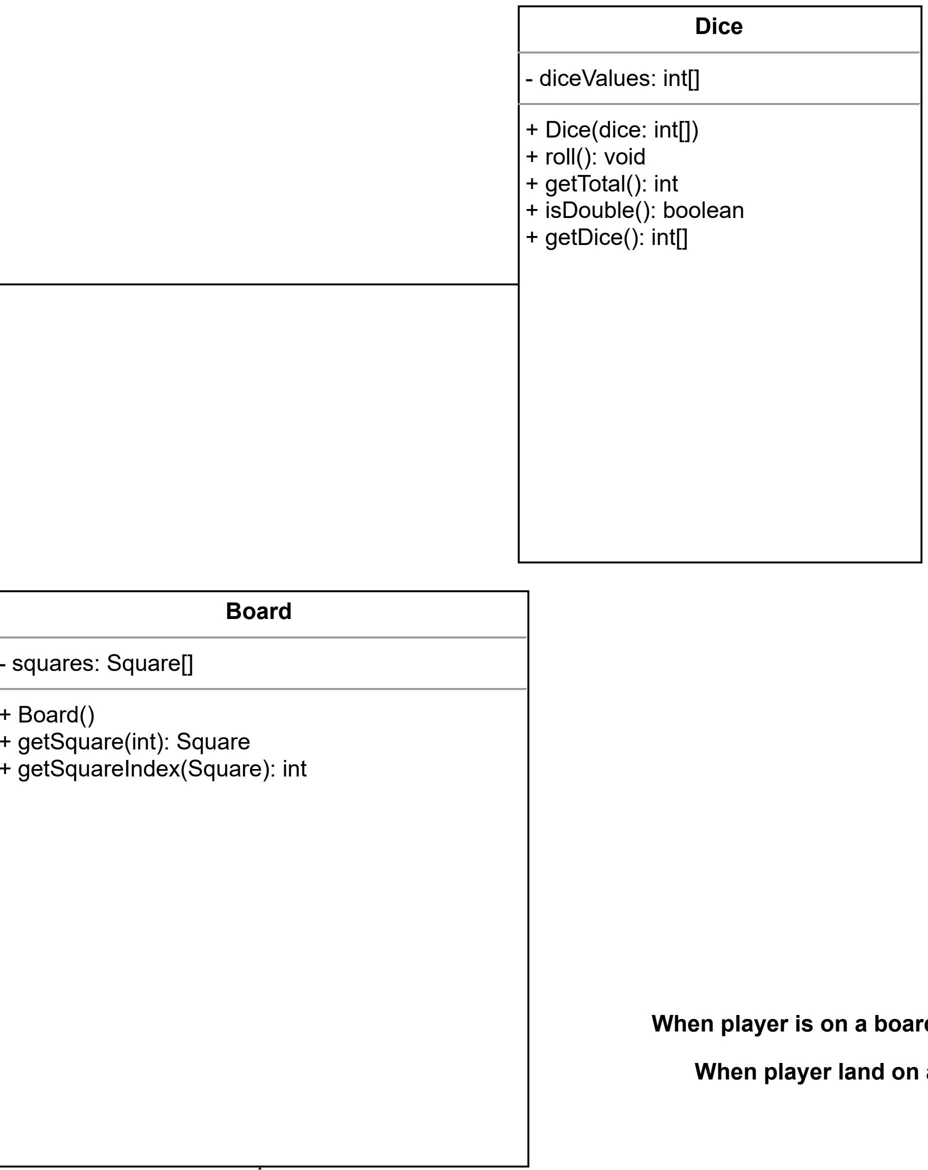
$1:
key: mone
value: a
enur

List of design patterns:
- strategy
- observer
- mvc
- java event model



What if money was just integers, and not these bills of specific values?


MonopolyUI
- monopolyController(MonopolyController)
+ UI()
+ play()
+ main()
MonopolyController
+ players: ArrayList<Player>
+ dice: Dice
+ MonopolyController()
+ runGame(): void
- rollDice(): int
Rolls Dice and calculates players' next location.
Presents player with option on buying a property when landing on unowned property
Deducts money from player and gives to other player
20
ey value
mount
m
Player rolls
Bank
Type
k
Player
- name: String
- balance: int
- inJail: boolean
- location: Square
- propertiesOwned: ArrayList<PrivateProperty>
- doubleTally: int
+ Player()
+ rollDice()
+ moveTo()
+ buyProperty(PrivateProperty): boolean



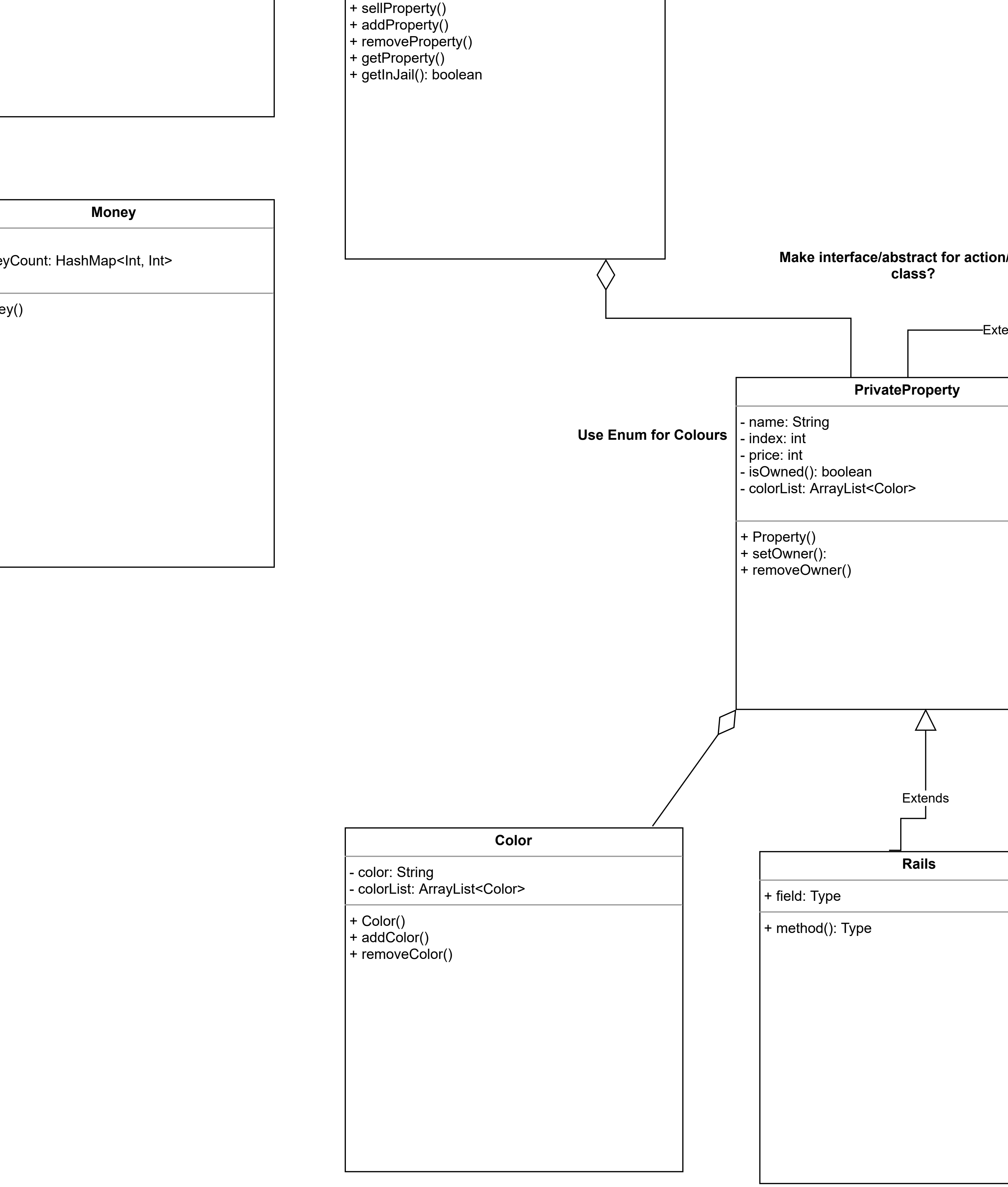
When player is on a board, something should happen...

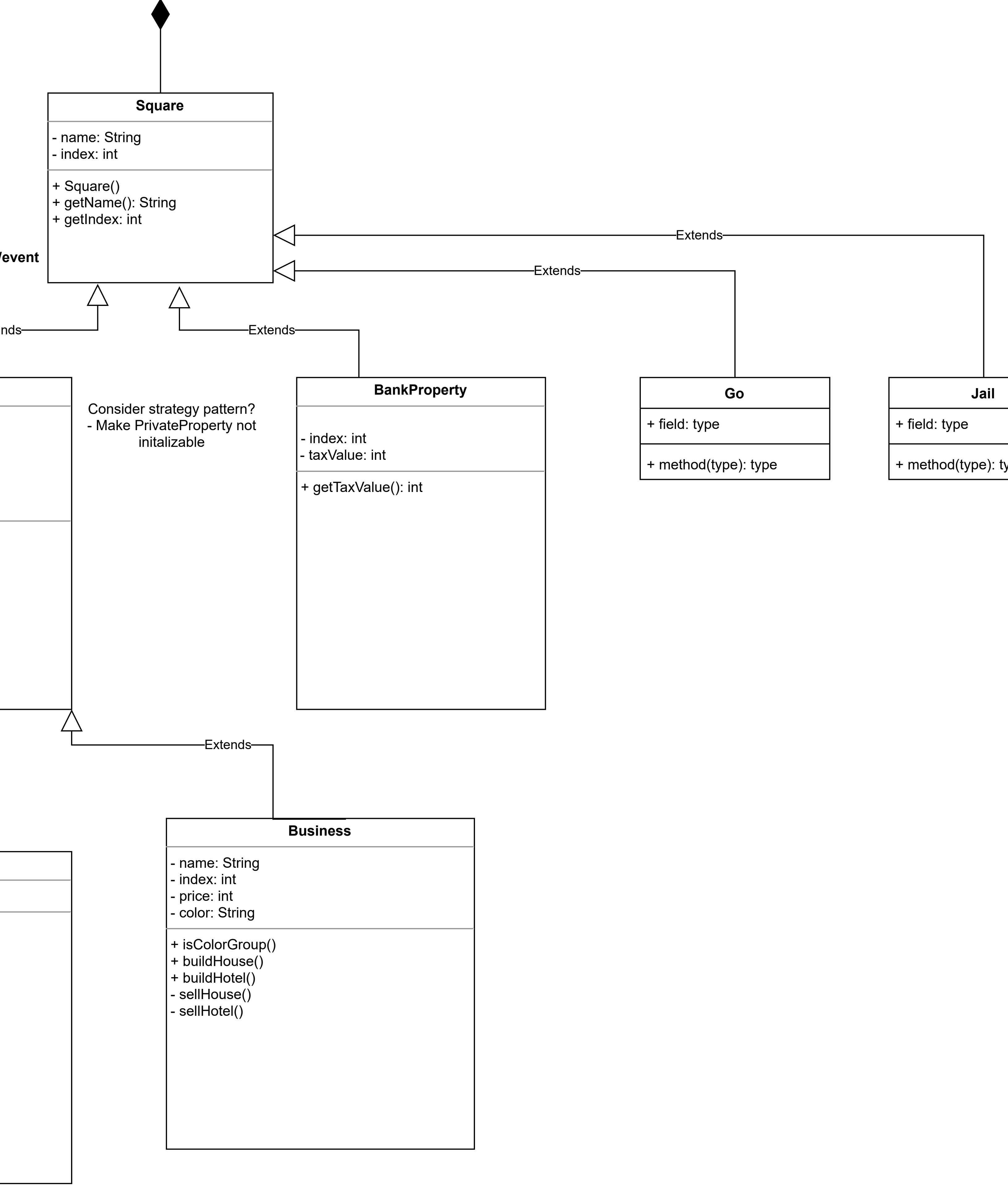
When player land on a square, an event happens?

Note: We will need to "serialize" the

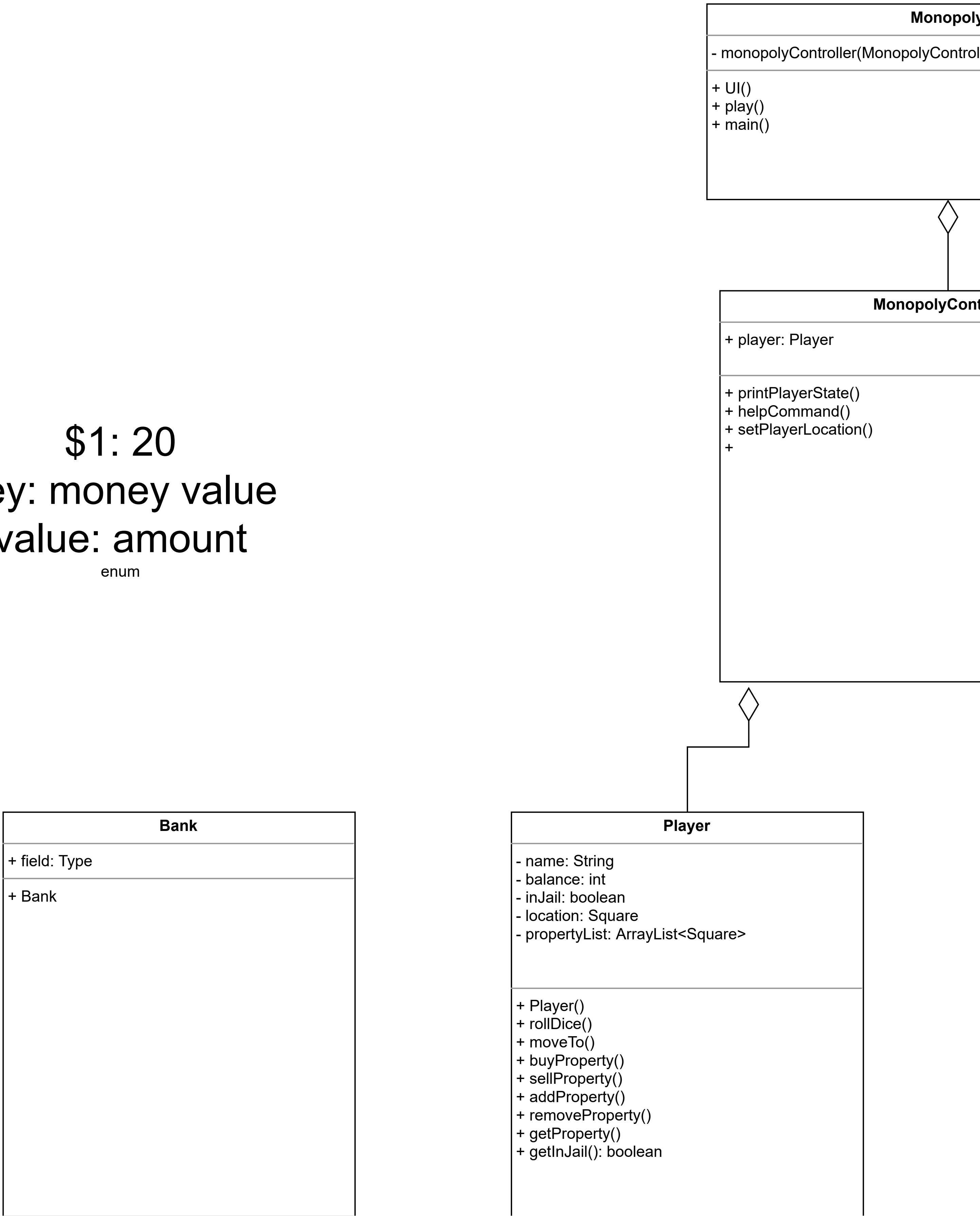
- mone

+ Mon
+ get

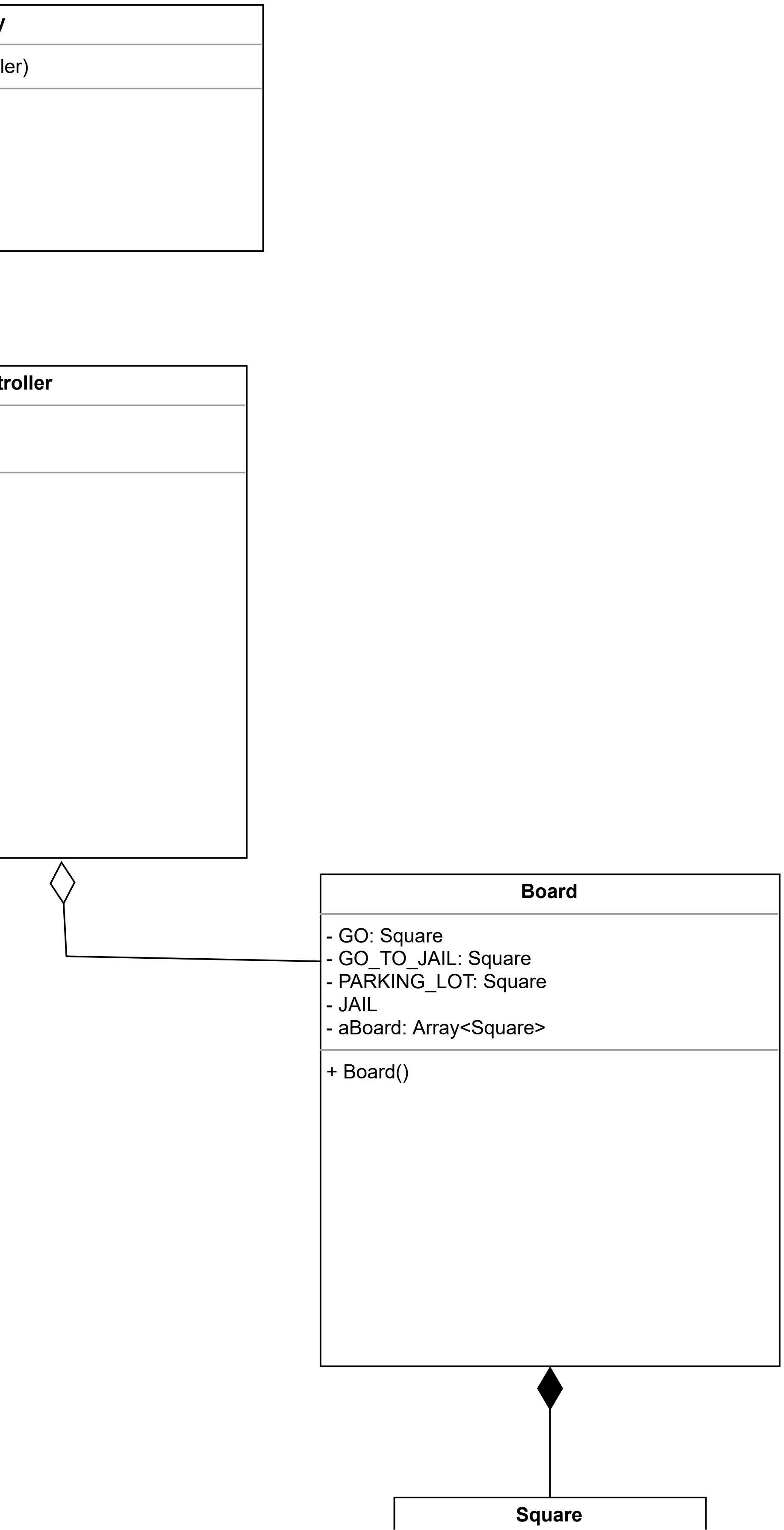

+ sellProperty()
+ addProperty()
+ removeProperty()
+ getProperty()
+ getInJail(): boolean
Money
eyCount: HashMap<Int, Int>
ey()
Make interface/abstract for action/
class?
Exte
PrivateProperty
- name: String
- index: int
- price: int
- isOwned(): boolean
- colorList: ArrayList<Color>
+ Property()
+ setOwner():
+ removeOwner()
Use Enum for Colours
Extends
Color
- color: String
- colorList: ArrayList<Color>
+ Color()
+ addColor()
+ removeColor()
Rails
+ field: Type
+ method(): Type


Square
- name: String
- index: int
+ Square()
+ getName(): String
+ getIndex: int
event
Extends
Extends
nds
Extends
Consider strategy pattern?
- Make PrivateProperty not initalizable
BankProperty
- index: int
- taxValue: int
+ getTaxValue(): int
Go
+ field: type
+ method(type): type
Jail
+ field: type
+ method(type): ty
Extends
Business
- name: String
- index: int
- price: int
- color: String
+ isColorGroup()
+ buildHouse()
+ buildHotel()
- sellHouse()
- sellHotel()

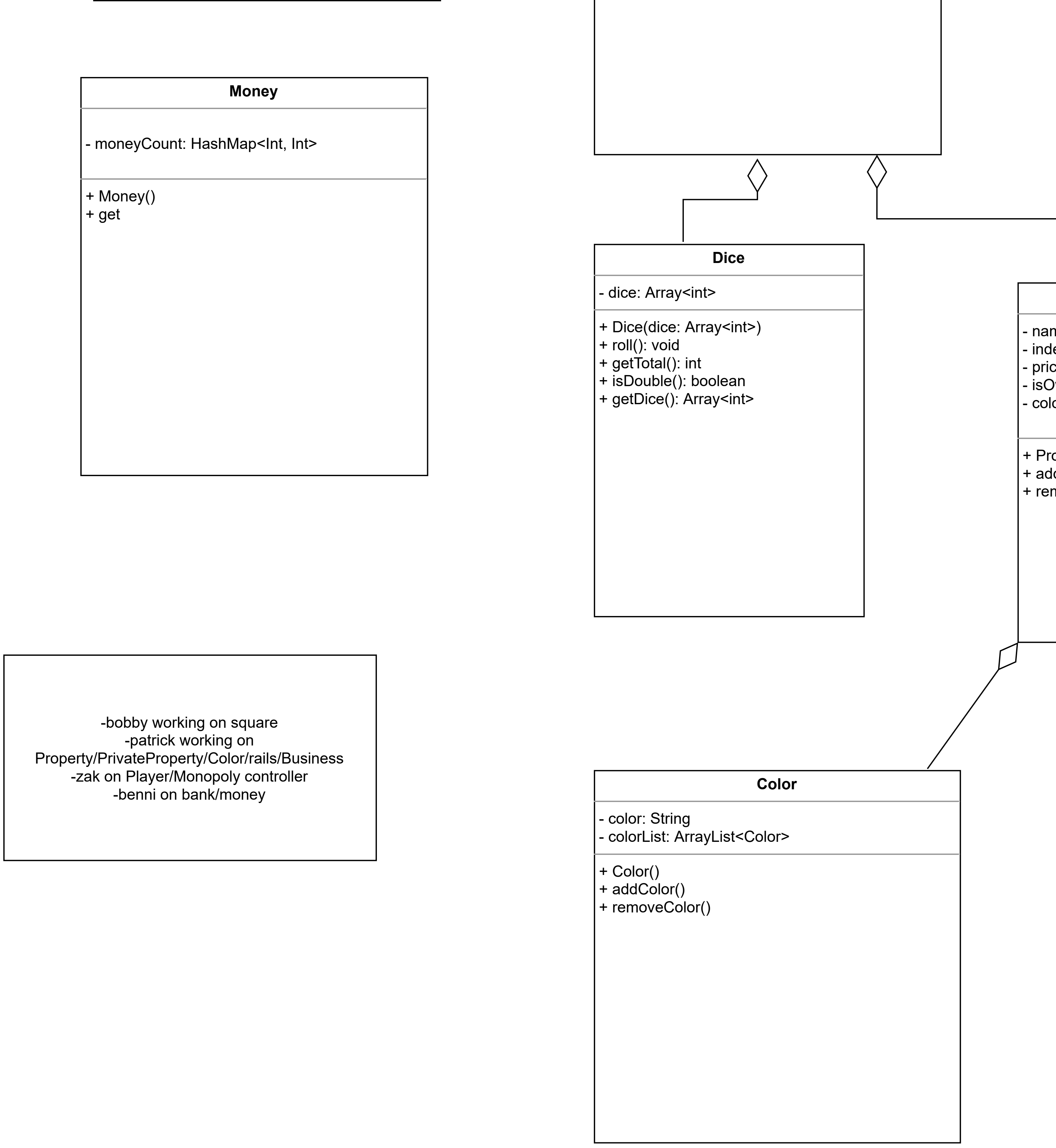
ype

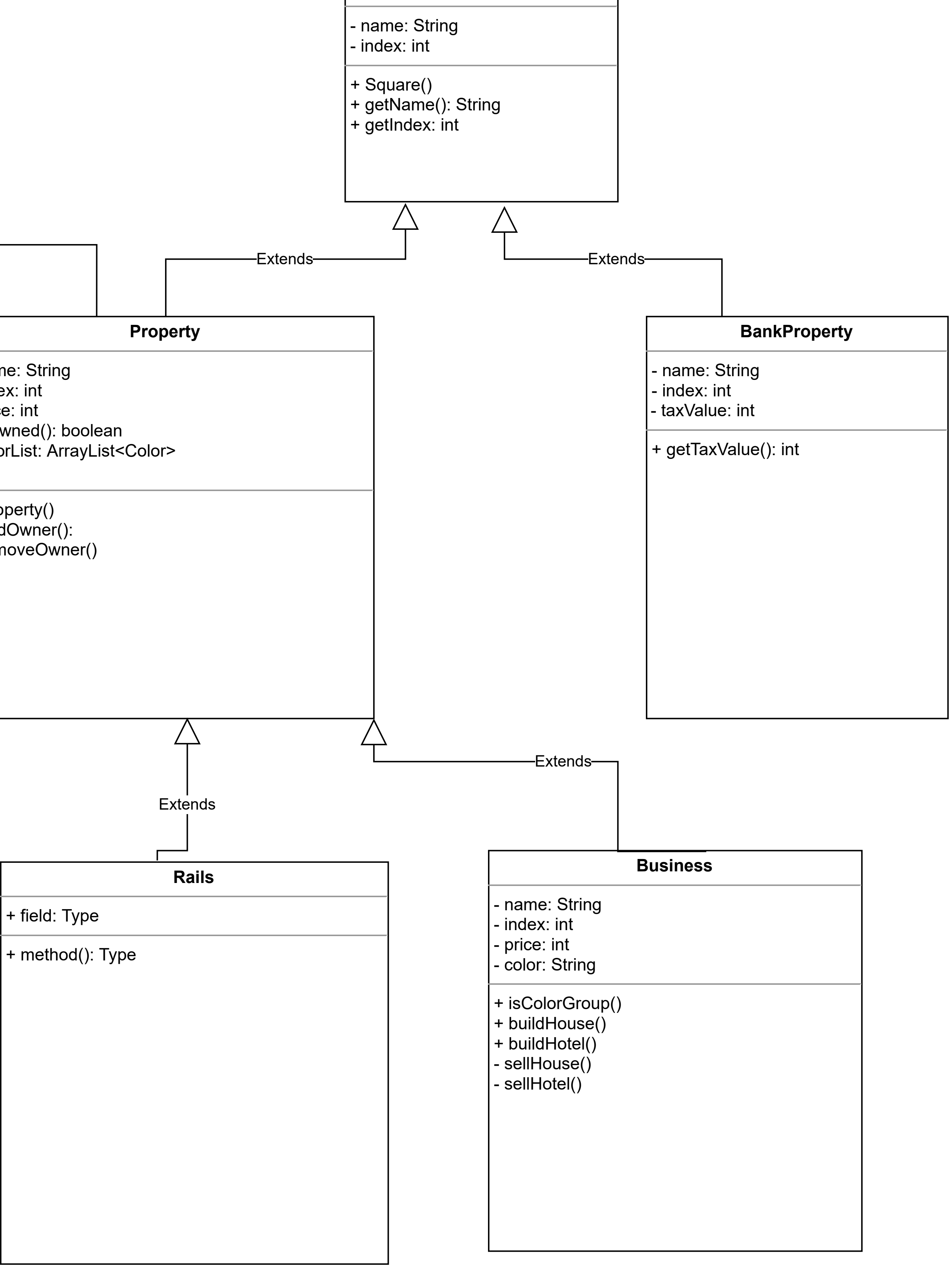


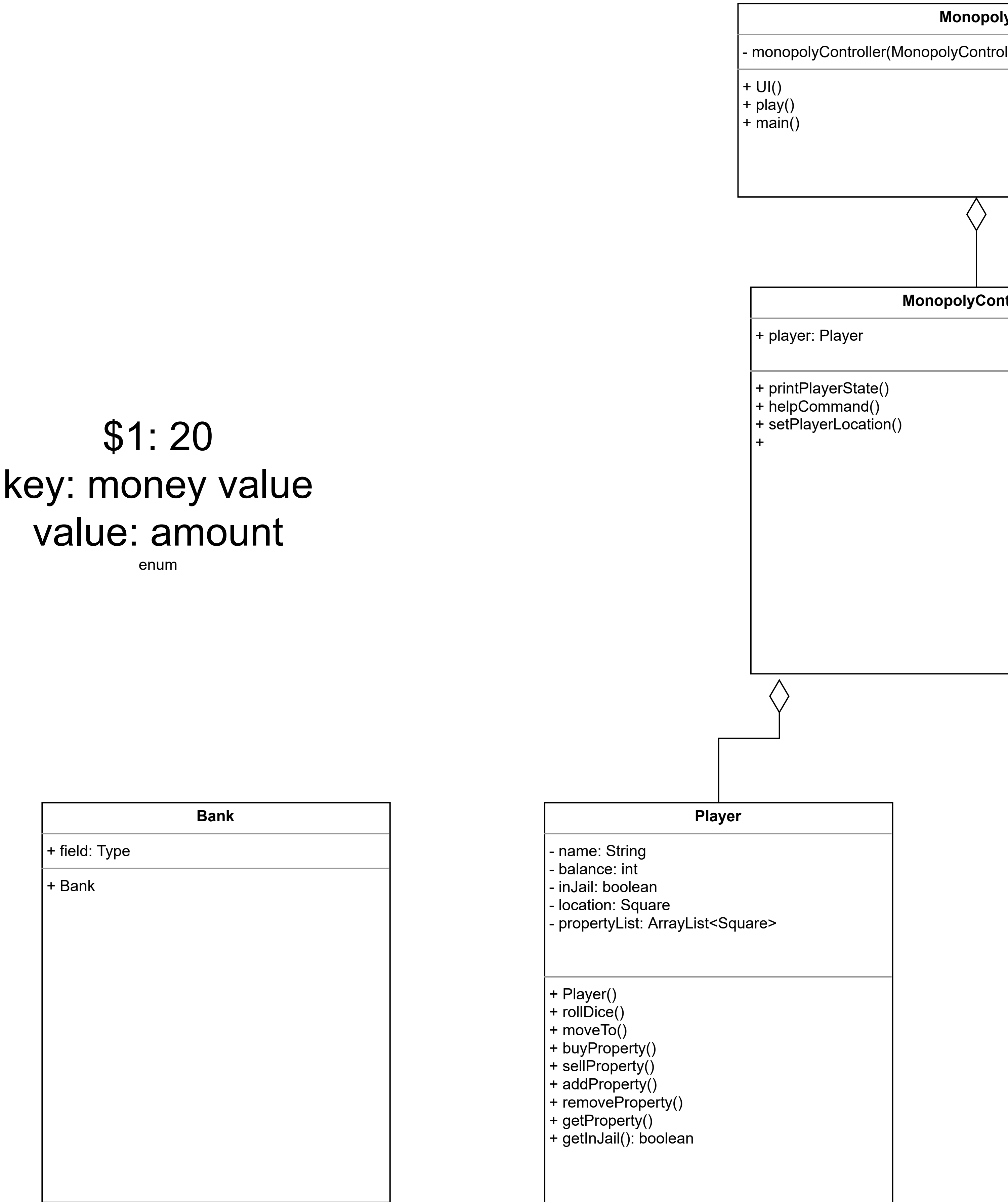
$1: 20
key: money value
value: amount
enum

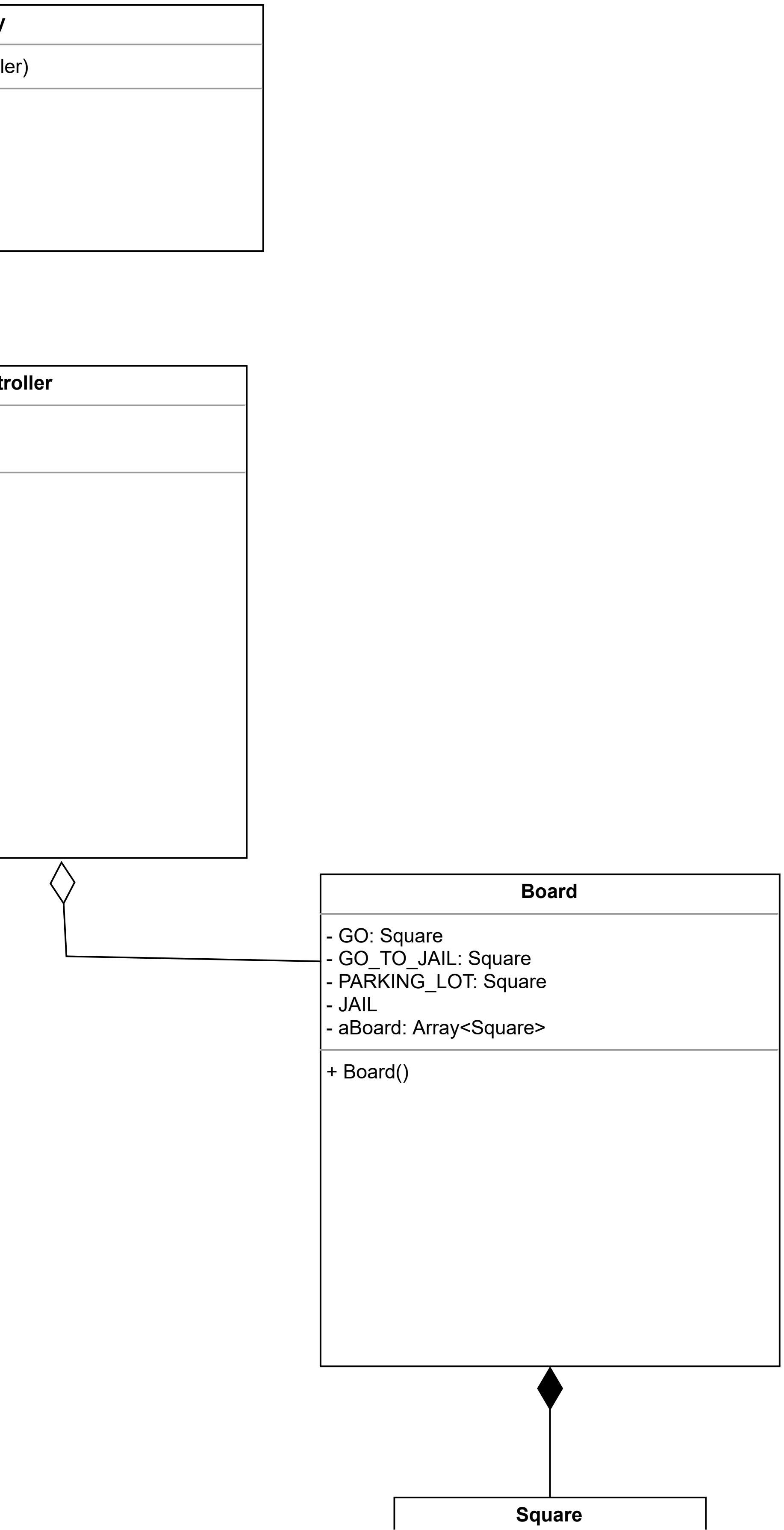


V
ler)
troller
Board
- GO: Square
- GO_TO_JAIL: Square
- PARKING_LOT: Square
- JAIL
- aBoard: Array<Square>
+ Board()
Square

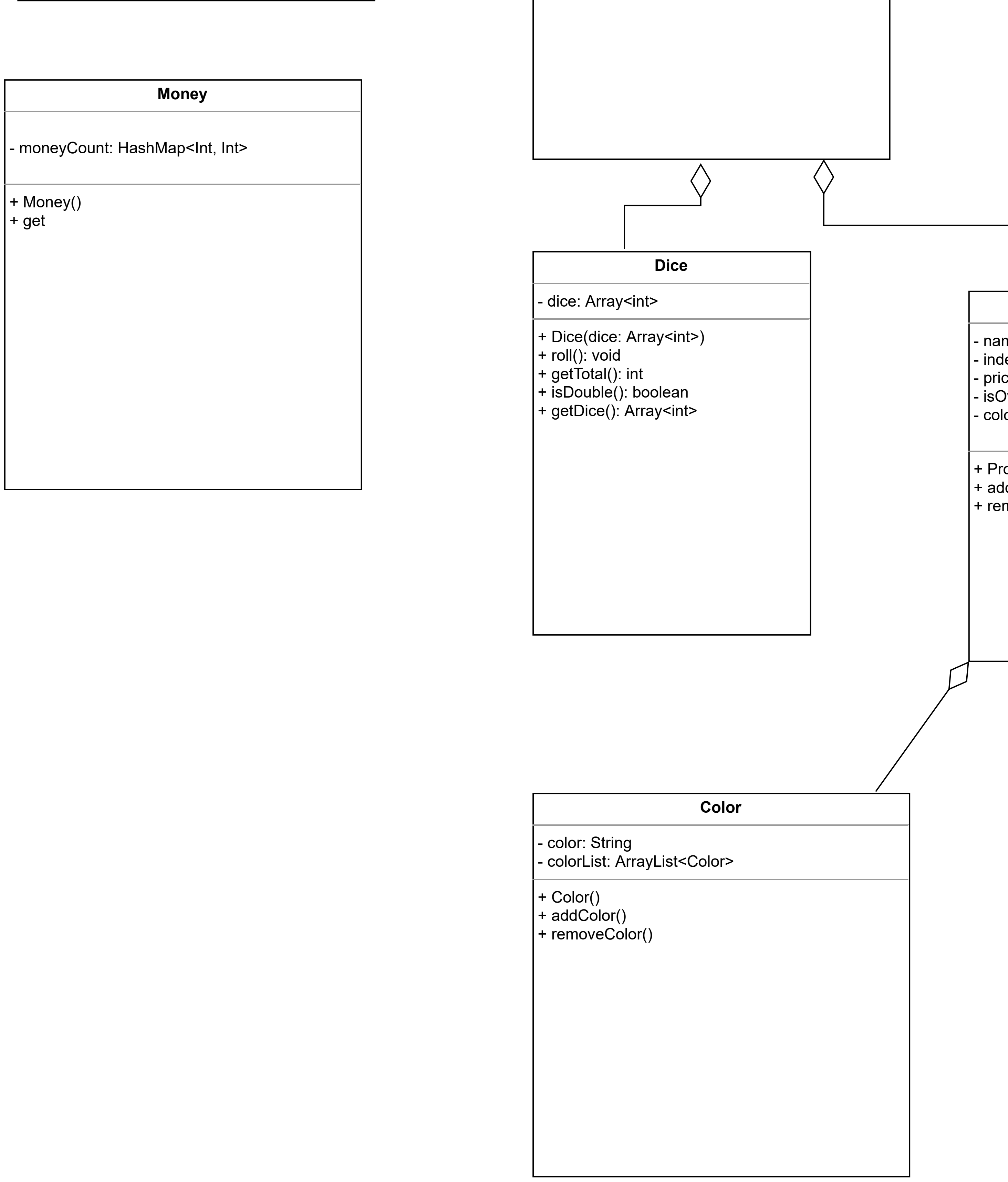

Money
- moneyCount: HashMap<Int, Int>
+ Money()
+ get
Dice
- dice: Array<int>
+ Dice(dice: Array<int>)
+ roll(): void
+ getTotal(): int
+ isDouble(): boolean
+ getDice(): Array<int>
- nam
- inde
- pric
- isO
- colo
+ Pro
+ add
+ rem
Color
- color: String
- colorList: ArrayList<Color>
+ Color()
+ addColor()
+ removeColor()
-bobby working on square
-patrick working on
Property/PrivateProperty/Color/rails/Business
-zak on Player/Monopoly controller
-benni on bank/money


- name: String
- index: int
+ Square()
+ getName(): String
+ getIndex: int
Extends
Extends
Property
me: String
ex: int
e: int
wned(): boolean
orList: ArrayList<Color>
operty()
dOwner():
noveOwner()
BankProperty
- name: String
- index: int
- taxValue: int
+ getTaxValue(): int
Extends
Extends
Rails
+ field: Type
+ method(): Type
Business
- name: String
- index: int
- price: int
- color: String
+ isColorGroup()
+ buildHouse()
+ buildHotel()
- sellHouse()
- sellHotel()


Monopoly
- monopolyController(MonopolyControl
+ UI()
+ play()
+ main()
MonopolyCont
+ player: Player
+ printPlayerState()
+ helpCommand()
+ setPlayerLocation()
+
$1: 20
key: money value
value: amount
enum
Bank
+ field: Type
+ Bank
Player
- name: String
- balance: int
- inJail: boolean
- location: Square
- propertyList: ArrayList<Square>
+ Player()
+ rollDice()
+ moveTo()
+ buyProperty()
+ sellProperty()
+ addProperty()
+ removeProperty()
+ getProperty()
+ getInJail(): boolean



y
ler)
troller
Board
- GO: Square
- GO_TO_JAIL: Square
- PARKING_LOT: Square
- JAIL
- aBoard: Array<Square>
+ Board()
Square



Money

- moneyCount: HashMap<Int, Int>

+ Money()
+ get

Dice

- dice: Array<int>

+ Dice(dice: Array<int>)
+ roll(): void
+ getTotal(): int
+ isDouble(): boolean
+ getDice(): Array<int>

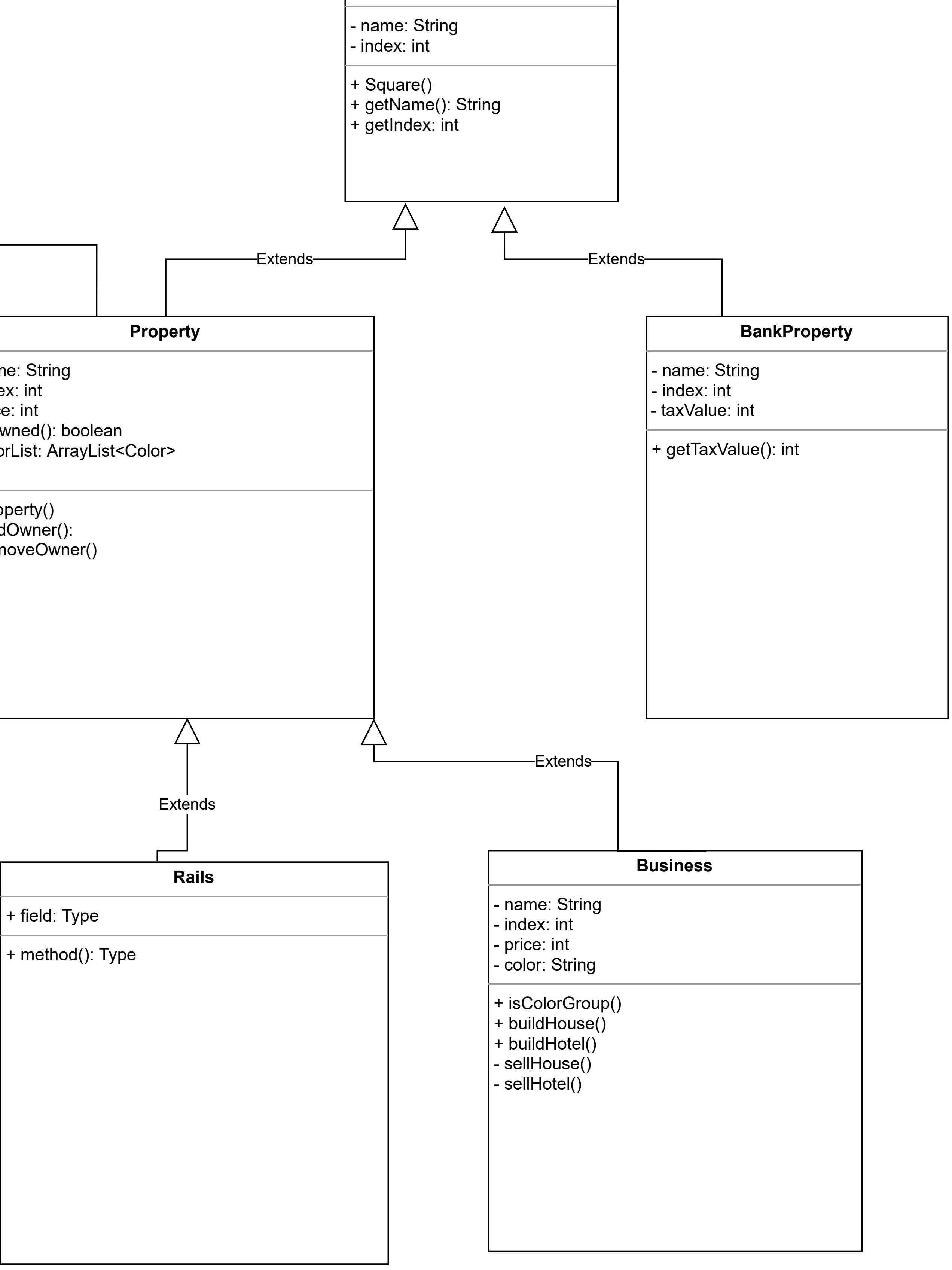
- nam
- inde
- pric
- isO
- colo

+ Pro
+ ado
+ rem

Color

- color: String
- colorList: ArrayList<Color>

+ Color()
+ addColor()
+ removeColor()


- name: String
- index: int
+ Square()
+ getName(): String
+ getIndex: int
Extends
Extends
Property
ne: String
ex: int
e: int
wned(): boolean
orList: ArrayList<Color>
operty()
dOwner():
noveOwner()
BankProperty
- name: String
- index: int
- taxValue: int
+ getTaxValue(): int
Extends
Extends
Rails
+ field: Type
+ method(): Type
Business
- name: String
- index: int
- price: int
- color: String
+ isColorGroup()
+ buildHouse()
+ buildHotel()
- sellHouse()
- sellHotel()

Tax Prompt (bankrupt) example:

```
Player X is currently at Square Y.
Player X rolls the dice. A 3 and 4 are rolled.
Player X has landed at Tax Square.

Player X, you have 5$ cash, your net worth is 20$, and the tax to
the bank is 50$.
Player X cannot afford to pay the tax.

--GAME OVER--
Winner: Player K with a net worth of 1000$
```

Tax Prom

```
Player X
Player X
Player X

Player X,
1. Pay th
2. Sell pr
3. Displa

Enter cho
```

Prompts for rent should be similar

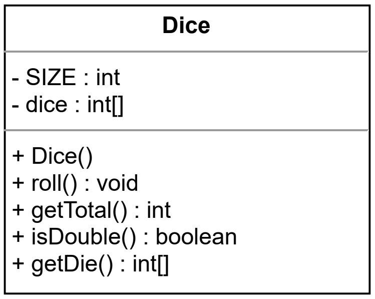
NOTE: there is also th
Simply tell the us

Unowned property (can afford) prompt example:

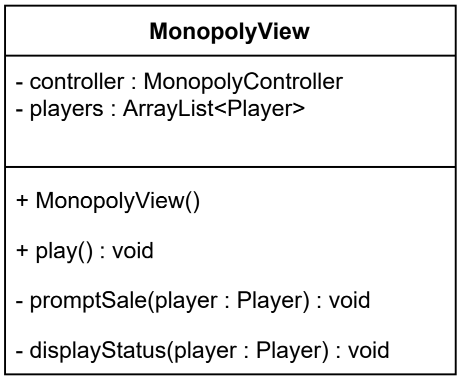
```
Player X is currently at Square Y.
Player X rolls the dice. A 3 and 4 are rolled.
Player X, you have landed at Unowned Private Property Square.

Player X, you have 100$ cash, your net worth is 500$, and the price of UPPS is 50$.
1. Purchase UPPS and end turn
2. End turn without purchasing UPPS
3. Sell properties
4. Display player status

Enter choice:
```

Unowned property (car

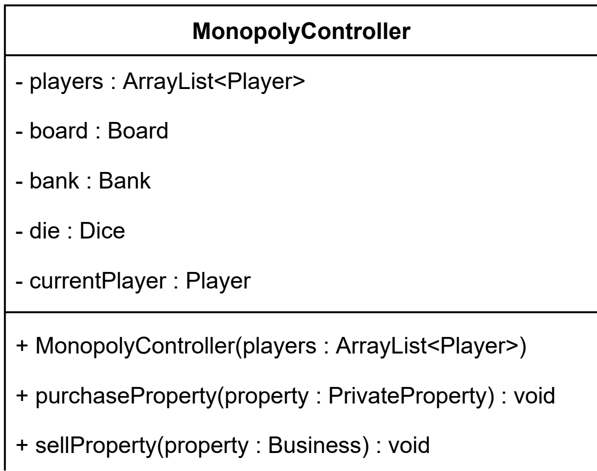
```
Player X is currently at
Player X rolls the dice.
Player X, you have lanc

Player X, you have 100
Player X cannot afford
1. End turn
2. Sell properties
3. Display player status

Enter choice:
```



```
Dice
- SIZE : int
- dice : int[]
+ Dice()
+ roll() : void
+ getTotal() : int
+ isDouble() : boolean
+ getDie() : int[]
```



```
MonopolyView
- controller : MonopolyController
- players : ArrayList<Player>
+ MonopolyView()
+ play() : void
- promptSale(player : Player) : void
- displayStatus(player : Player) : void
```



```
MonopolyController
- players : ArrayList<Player>
- board : Board
- bank : Bank
- die : Dice
- currentPlayer : Player
+ MonopolyController(players : ArrayList<Player>)
+ purchaseProperty(property : PrivateProperty) : void
+ sellProperty(property : Business) : void
```

npt (can afford) example:

is currently at Square Y.
rolls the dice. A 3 and 4 are rolled.
has landed at Tax Square.

, you have 100$ cash, your net worth is 500$, and the tax to the bank is 50$.
e tax and end turn
roperties
y player status

oice:

he case of having enough netWorth but not enough cash.
ser to sell some properties before ending their turn.

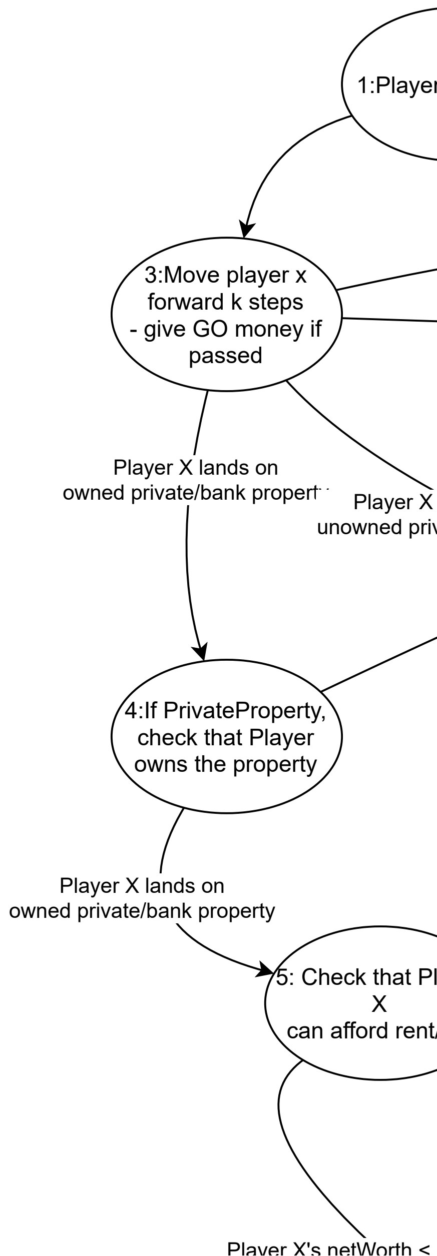
nnot afford) prompt example:

Square Y.
A 3 and 4 are rolled.
ded at Unowned Private Property Square.

0$ cash, your net worth is 500$, and the price of UPPS is 800$.
UPPS.

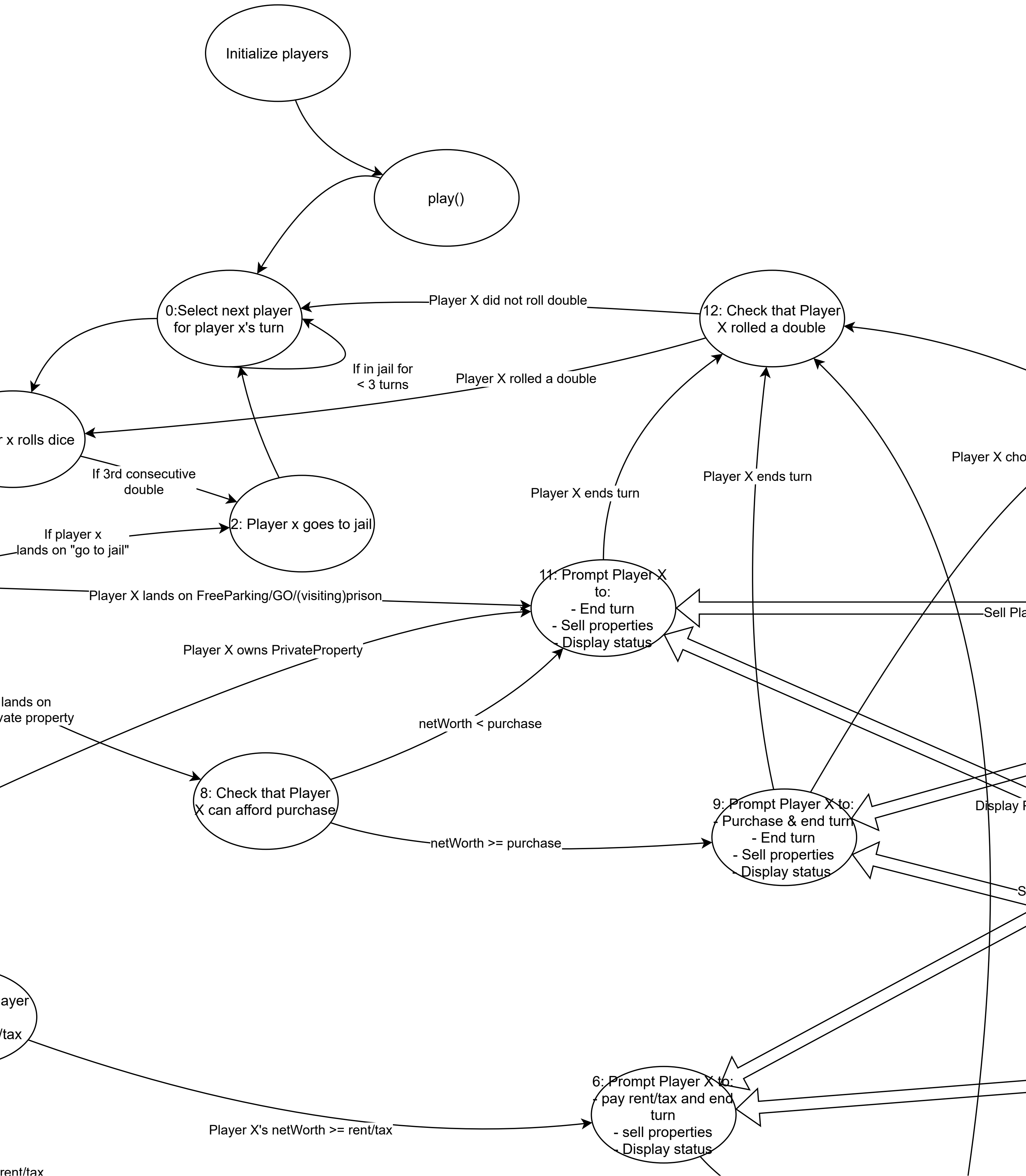
s

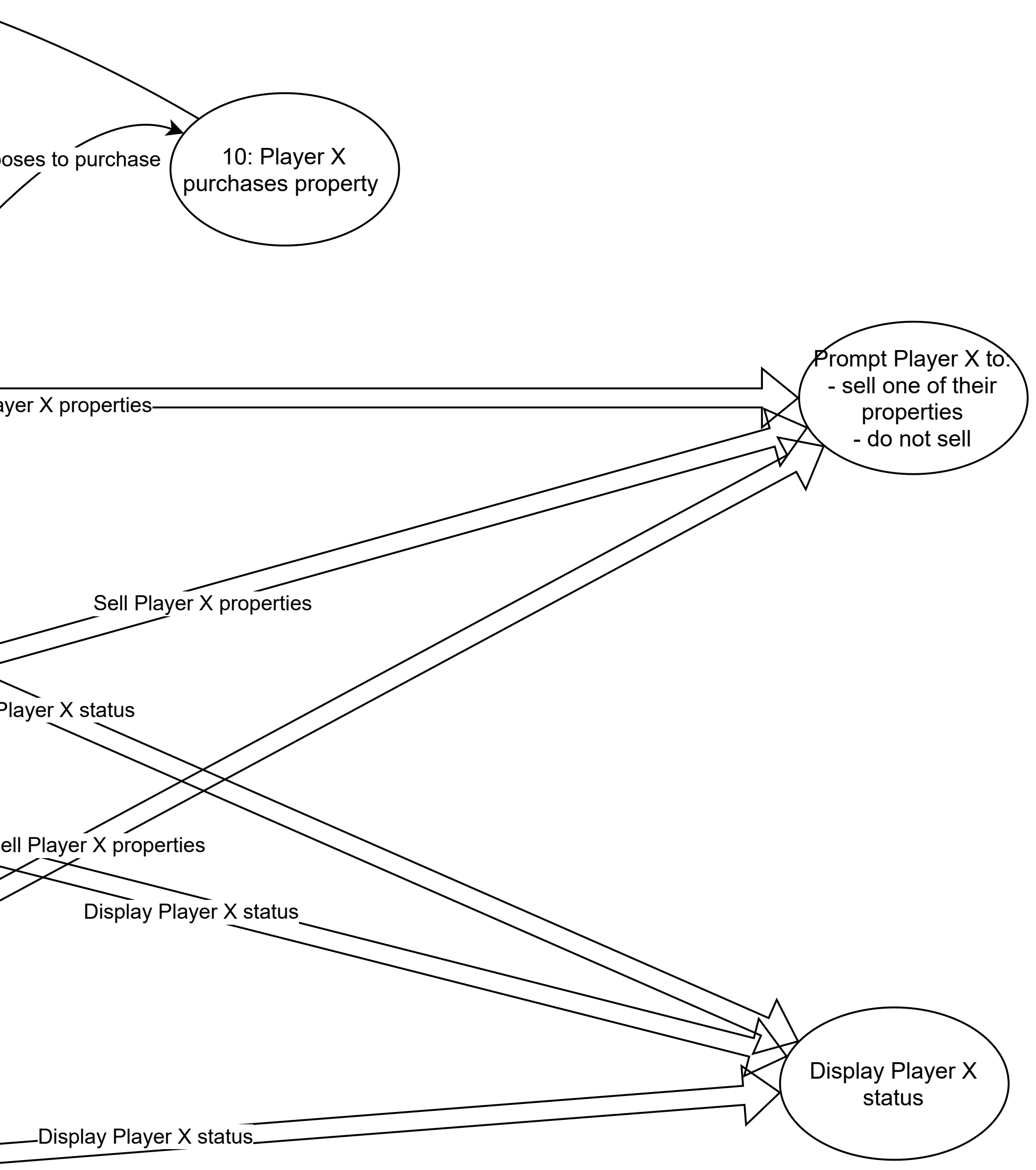
Note: I need a "visiting jail"



- list of Players will be defined and declared in MonopolyView
via user input (customizable # players) and passed to Controller
- note: return value of functions is subject to discretion.
if i cant think of one now i put void but if you think it should return
then make it return
- sellProperty() is used by promptSale()
- i want a Die class so that I MonopolyView can access and present
its data EDIT: nvm bobby wrote one poggers

FIXME: i forgot to include case where player landsd on their own pprperty


Initialize players
play()
0:Select next player for player x's turn
12: Check that Player X rolled a double
Player X did not roll double
If in jail for < 3 turns
Player X rolled a double
r x rolls dice
If 3rd consecutive double
2: Player x goes to jail
If player x lands on "go to jail"
Player X lands on FreeParking/GO/(visiting)prison
Player X ends turn
Player X ends turn
Player X cho
11: Prompt Player X to:
- End turn
- Sell properties
- Display status
Sell Pla
Player X owns PrivateProperty
lands on
vate property
netWorth < purchase
8: Check that Player X can afford purchase
netWorth >= purchase
9: Prompt Player X to:
- Purchase & end turn
- End turn
- Sell properties
- Display status
Display P
S
ayer
/tax
Player X's netWorth >= rent/tax
6: Prompt Player X to:
- pay rent/tax and end turn
- sell properties
- Display status
rent/tax



-Use a "returnState" var to return to current st
after entering state with multiple entryways?
- make a function that gives a prompt variant
depending on parameters passed?
- > void promptPlayer(Player, Square
PromptType.TYPE)
note: PromptType is an Enum {AffordPurchas
CannotPurchase, AffordRent, AffordTax}

Note: the act of paying taxes vs rent does not
seem all that different to me.... Is there a way to m
this one single function?

ate

e,

nake

+ sellProperty(property : Rail) : void

Notes:-
purchaseProperty does not check
if the currentPlayer does already
own the property.

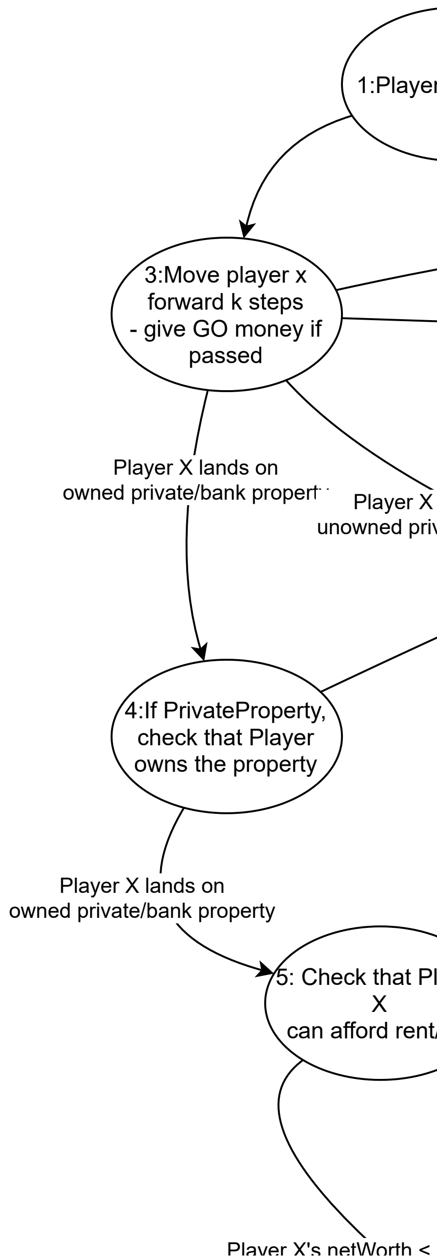
purchaseProperty automatically
gets a house for the square if
a full colour set is owned by
the currentPlayer

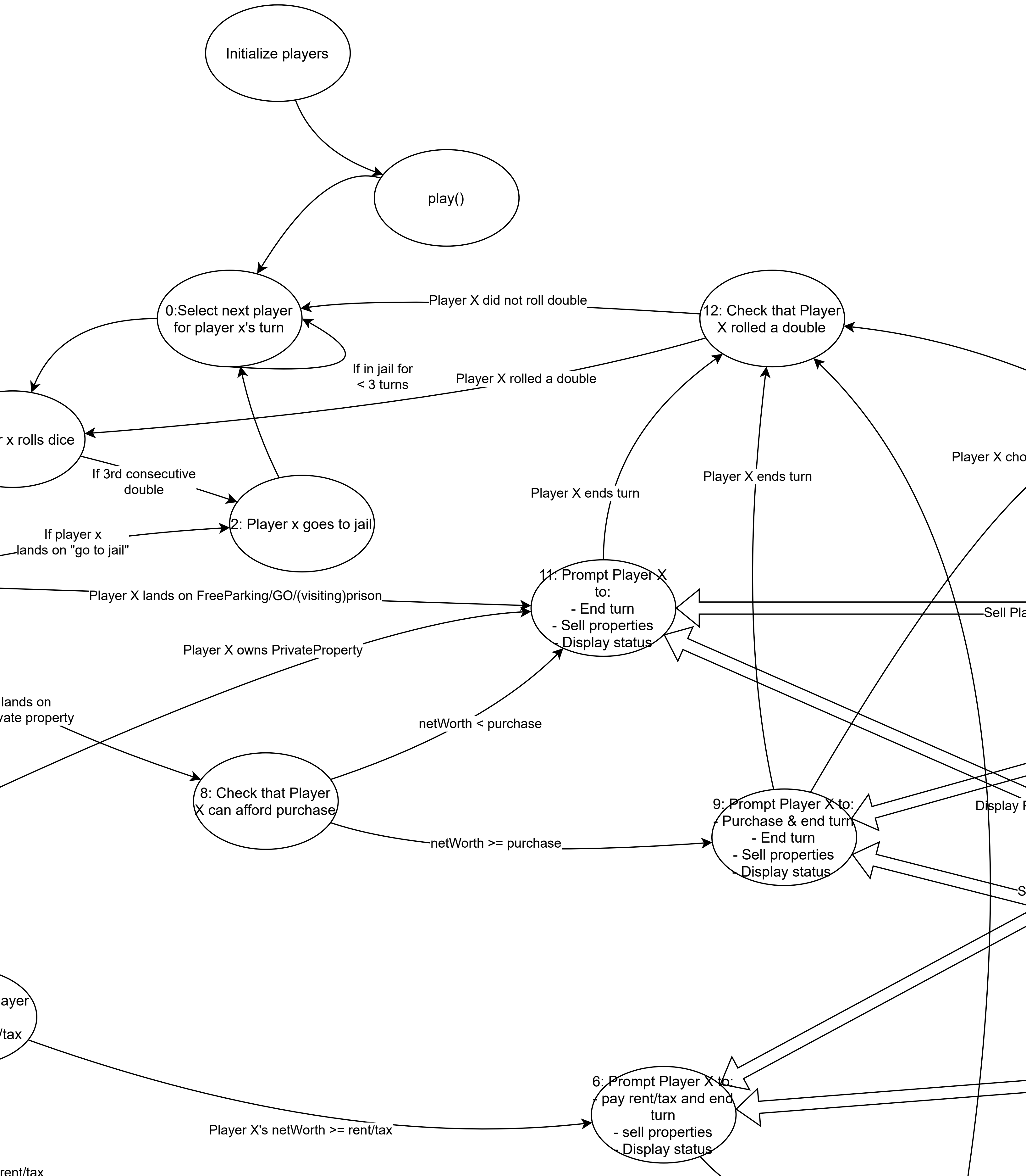
currentPlayer set to which ever Player
is at the index 0 of the ArrayList<Player>
(passed as a parameter in
MonopolyController())

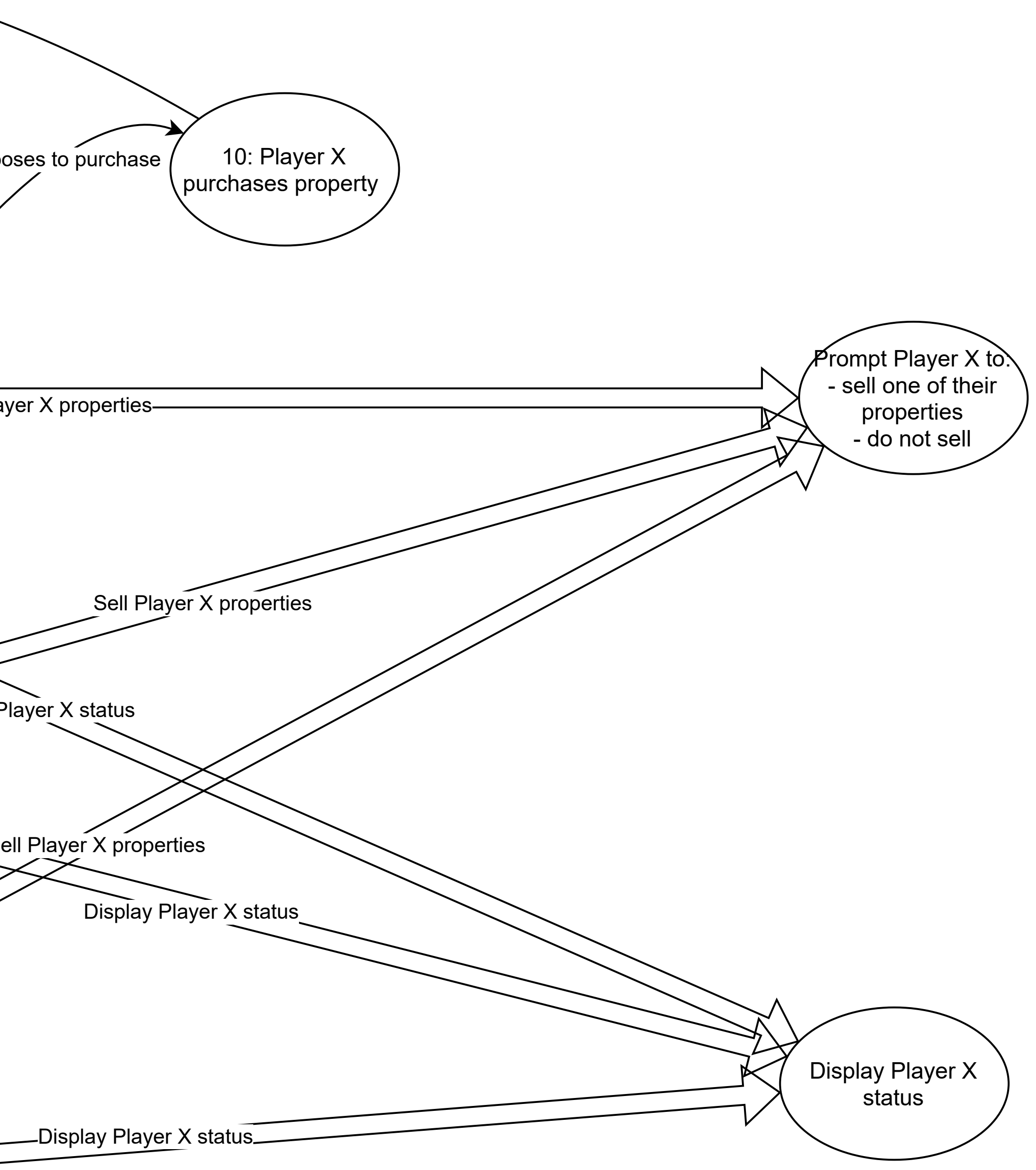
...after each turn you could put a new
player at index 0

Player X's networth

rent/tax
Player X pays rent/tax
7: Player X pays rent/tax to Player Y/bank
END GAME Declare Winner


1:Player
3:Move player x
forward k steps
- give GO money if
passed
Player X lands on
owned private/bank propert
Player X
unowned priv
4:If PrivateProperty,
check that Player
owns the property
Player X lands on
owned private/bank property
5: Check that Pl
X
can afford rent
Player X's netWorth <


Initialize players
play()
0:Select next player for player x's turn
12: Check that Player X rolled a double
Player X did not roll double
If in jail for < 3 turns
Player X rolled a double
r x rolls dice
If 3rd consecutive double
2: Player x goes to jail
If player x lands on "go to jail"
Player X cho
Player X ends turn
Player X ends turn
Player X lands on FreeParking/GO/(visiting)prison
11: Prompt Player X to: - End turn - Sell properties - Display status
Sell Pla
Player X owns PrivateProperty
lands on vate property
netWorth < purchase
8: Check that Player X can afford purchase
netWorth >= purchase
9: Prompt Player X to: - Purchase & end turn - End turn - Sell properties - Display status
Display P
S
ayer
/tax
Player X's netWorth >= rent/tax
6: Prompt Player X to: - pay rent/tax and end turn - sell properties - Display status
rent/tax



oses to purchase
10: Player X purchases property
ayer X properties
Prompt Player X to:
- sell one of their properties
- do not sell
Sell Player X properties
Player X status
ell Player X properties
Display Player X status
Display Player X status
Display Player X status

Player X's networth

rent/tax
Player X pays rent/tax
7: Player X pays rent/tax to Player Y/bank
END GAME Declare Winner