Sprint 2

1) **Summary**	
Sprint leader(s)	Vivien
Sprint start date	13/03/2020
Sprint end date	27/03/2020

2) **Individual key contributions**	
Team member	Key Contribution(s)
Neumann, Vivien	Task Cards, Requirement Analysis + Planning, Use Case Diagrams
Jiao, Haotian (Hallton)	Buy properties, houses and hotels
Wang, Mingfeng (Foret)	Jail
Banes, Hayden J	Connecting Classes from S1
Tang, Zhenyu (tang)	Potluck / opportunity cards

3) User stories / task card

Task Card 1: Connecting the classes from Sprint 1
Priority: 1
Value: 10

- Goal: have a running prototype
- Basic GUI

Task Card 2: Buy properties
Priority: 2
Value: 8

Players make progress in the game by buying property as they move around the board. Therefore, players have the opportunity - after they have completed one complete circuit of the board by passing the Go space - to purchase the property they have landed on after they have made their move. Given the property is still available which has to be checked before the user can make a purchase.

Sprint 2

All properties are initially the property of the bank. When a player purchases a property, the card is transferred from the bank to that player and the amount shown on the card is paid to the bank. It has to be ensured that the player paid the right amount of money for the property they buy. Additionally, the stored data of the property (field 'owner') has to be changed.

- Classes that are related to this task card: property and bank (have to update on game launch from Excel)

Task Card 3: Potluck / opportunity cards
Priority: 2
Value: 6

The Board also consists of the "potluck" space and an "opportunity knocks" space. If a player lands on these spaces, they take a card for the top of the corresponding pile and carry out the instructions on the card. Some cards can require that they pay a fine which is placed in the centre of the board (Free Parking), or pay money to another player, keep the card ("Get out of jail") or move to a specific space. These actions have to be kept in mind and the player has to be able to execute them accordingly although they have already moved their token.
When this is completed, the card is replaced at the bottom of the corresponding pile.

- Related Class: board

Task Card 4: Buying houses and hotels
Priority: 3
Value: 7

When a player has finished moving their token, and has completed any property purchase activity, they have the option to buy houses and hotels to improve their properties. Players are not permitted to improve their properties at any other time. Therefore, the player should not only see the value of the property but also the costs of purchasing houses which are shown on the game card. If a player wishes to buy a hotel, that is the equivalent of 5 houses in cost. However, since houses and hotels may only be purchased for properties where a player owns all of the properties in a particular colour coded group, this feature has to check which properties are owned by this player and whether they are able to develop the property.
Where a coloured set of properties is owned and developed by a player, there may never be a difference of more than 1 house between the properties in that set. This has to be checked before the purchase and show a notification if necessary. A player may have 4 houses on one set and a hotel on another in that set. The maximum development permitted on any one property is one hotel.
Nevertheless, the player has to have an option (e.g. as a pop-up window) where they can select the amount of houses or hotels they want to buy with its total costs. When the player has confirmed their purchase, it has to be checked if the player has enough money. If so, the player then gives the money to the bank. Otherwise the transaction is declined and the player will be notified.
After a player has purchased houses or hotels, the GUI has to change and show the development on this space. In addition, the rent of this property has to be adjusted and the changes have to be stored in the property card.

- Related Classes: player, board, bank, property

Task Card 5: Jail
Priority: 2

Sprint 2

Value: 8

A player is sent to jail through several possible scenarios:

1. A player throws a third double
2. A player lands on a 'go to jail' space
3. A player takes the card 'go to jail'

For each scenario, the player's token is immediately moved to the jail space (without passing Go).

To get out of jail, there are several possibilities:

- If a player is in jail, they may pay £50 in the next round to be released from jail. These £50 are added to the free parking fines. The player token is then moved to "just visiting" and the player's turn ends. The player takes a normal turn in the next round.
- If a player has a "get out of jail free" card, then they place the card at the bottom of the "pot luck" or "opportunity knocks" pile as appropriate, the player token is moved to "just visiting" and the player's turn ends. The player takes a normal turn in the next round. When using the card and placing it at the bottom of the appropriate pile, the card is now 'available' again and the card's status in the database is changed.
- If a player opts to stay in jail, they give up their turn for the next 2 rounds. Which means, at the end of the next 2 rounds, the player token is moved to "just visiting" and the player's turn ends. The player takes a normal turn in the next round.

In general: Whilst a player is in jail, a player may not collect any rents from other players.

4) Requirement analysis

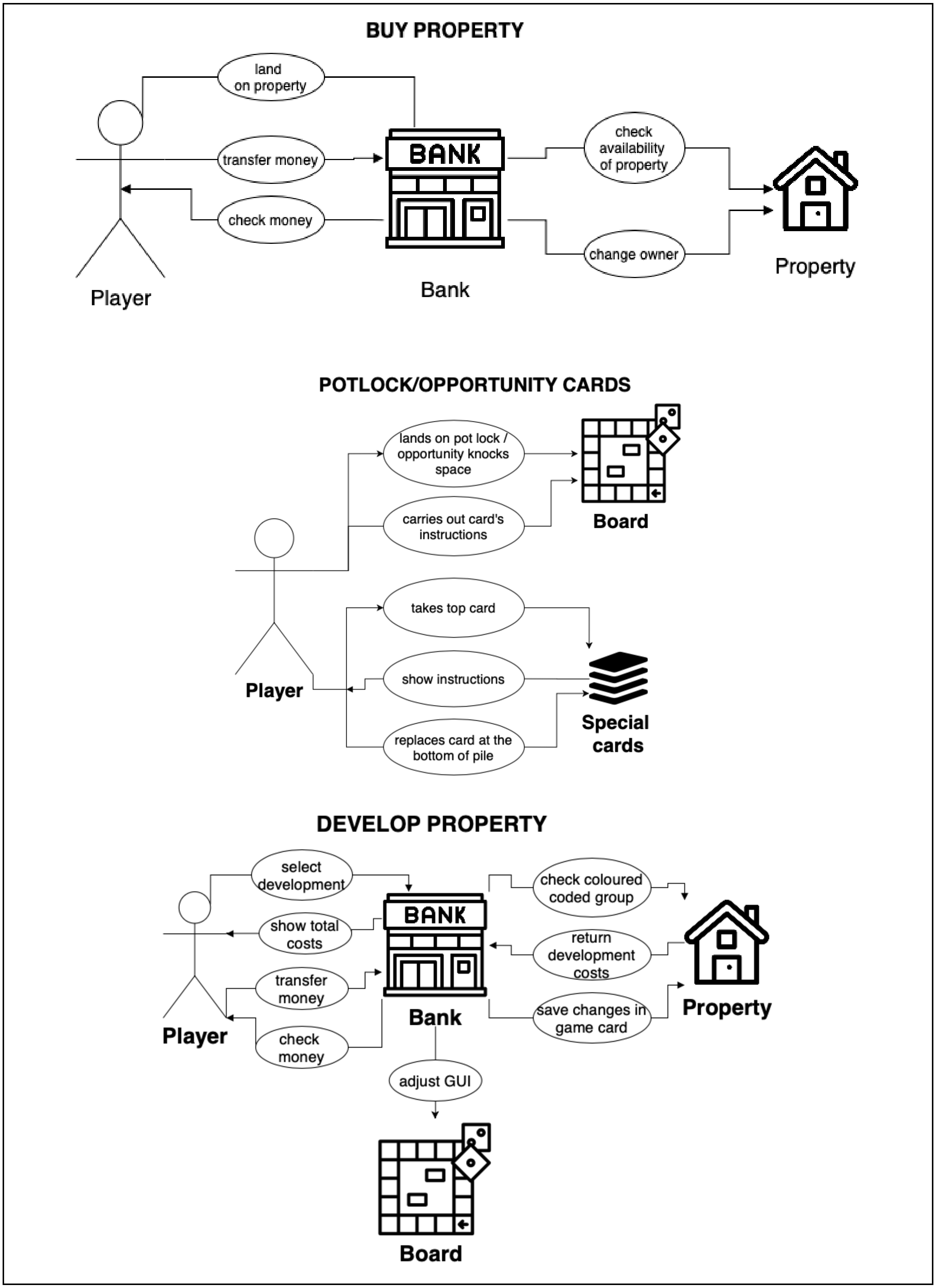
	Functional	Non-Functional
TC2: Buy Properties	TC2-F1: Player shall be able to buy a property TC2-F2: Show details of Property based on provided xls-spreadsheet TC2-F3: Before a player buys a property, the amount of money the player shall be compared with property price in order to check whether the player has enough money TC2-F4: When a player has enough money to buy property, the system shall change the owner of the property. TC2-F5: The amount of money for the property shall be transferred from the player's bank account to	

	the bank.	
TC3: Potluck & Opportunity Cards	TC3-F1: When player lands on Potluck or Opportunity Knocks space, one of the cards shall be shown TC3-F2: Carry out the instructions on the card (incl. moving the player) TC3-F3: After the player carried out the instructions, the card shall be replaced on the bottom of corresponding pile	TC3-NF1: Ensure that a card is shown only once at the time ("Get out of jail" card can be kept but is then not available for other players)
TC5: Jail	TC5-F1: If player takes the potlock/opportunity card "Go to jail", the player shall be moved to jail space TC5-F2: If a player takes the card "get out of jail", the player shall keep this card to use it later. TC5-F3: If player uses its "get out of jail" card, the card shall be placed on the bottom of the corresponding pile and the player moves to "just visiting" TC5-F4: If a player throws a double at the third turn, its turn ends immediately and the token shall be moved to jail TC5-F5: Release player from jail after the player has paid £50. TC5-F6: Release player from jail after they gave up its turns for 2 rounds	TC3-NF1: Ensure that "get out of jail" card is kept and only after using it, placed on the bottom of the corresponding pile TC3-NF2: Ensure that user is not released from jail before they have payed or gave up two turns

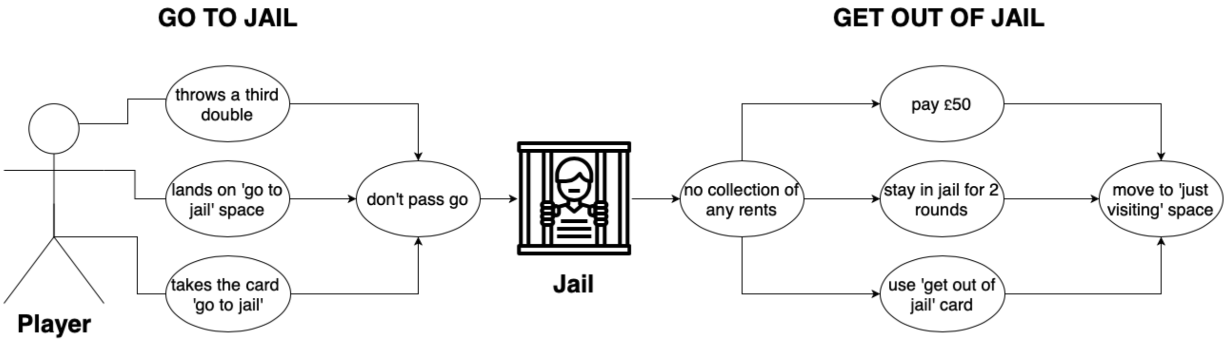
5) Design

Use Case Diagrams:

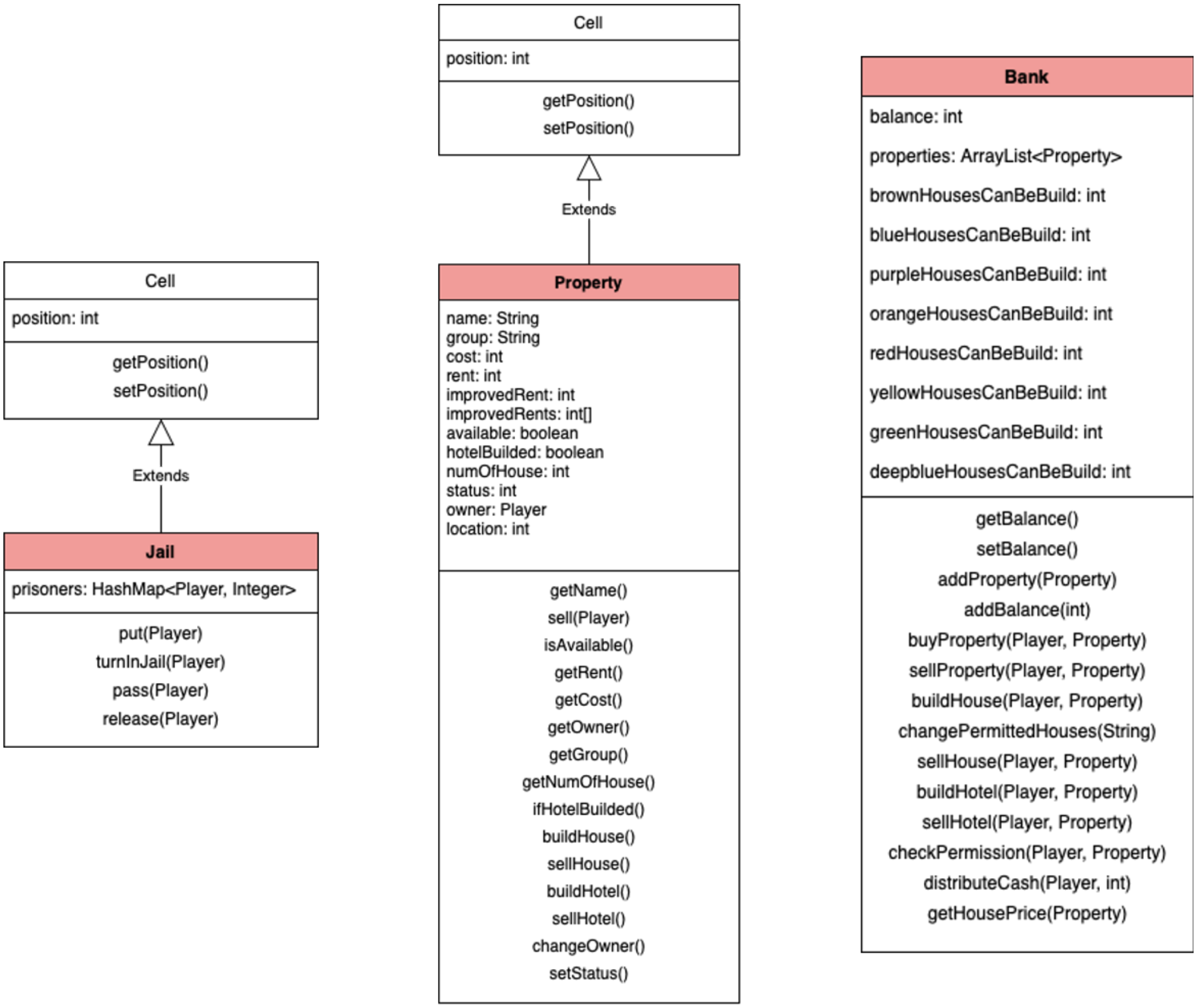
Sprint 2


BUY PROPERTY
land on property
transfer money
check money
BANK
check availability of property
change owner
Player
Bank
Property
POTLOCK/OPPORTUNITY CARDS
lands on pot lock / opportunity knocks space
carries out card's instructions
Board
takes top card
show instructions
replaces card at the bottom of pile
Player
Special cards
DEVELOP PROPERTY
select development
show total costs
transfer money
check money
BANK
Bank
check coloured coded group
return development costs
save changes in game card
Property
Player
adjust GUI
Board

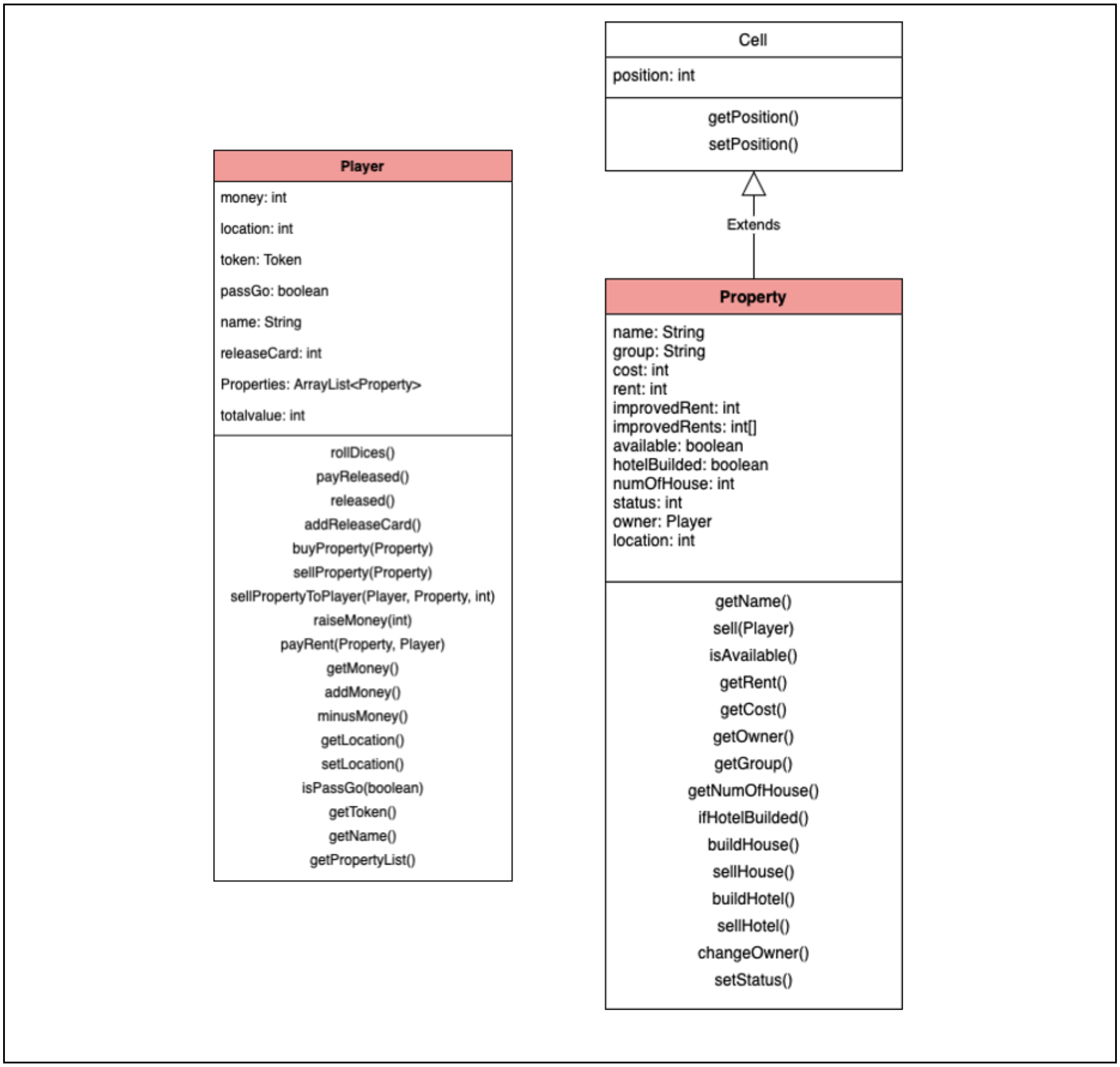
Sprint 2


GO TO JAIL
GET OUT OF JAIL
Player
throws a third double
lands on 'go to jail' space
takes the card 'go to jail'
don't pass go
Jail
no collection of any rents
pay £50
stay in jail for 2 rounds
use 'get out of jail' card
move to 'just visiting' space

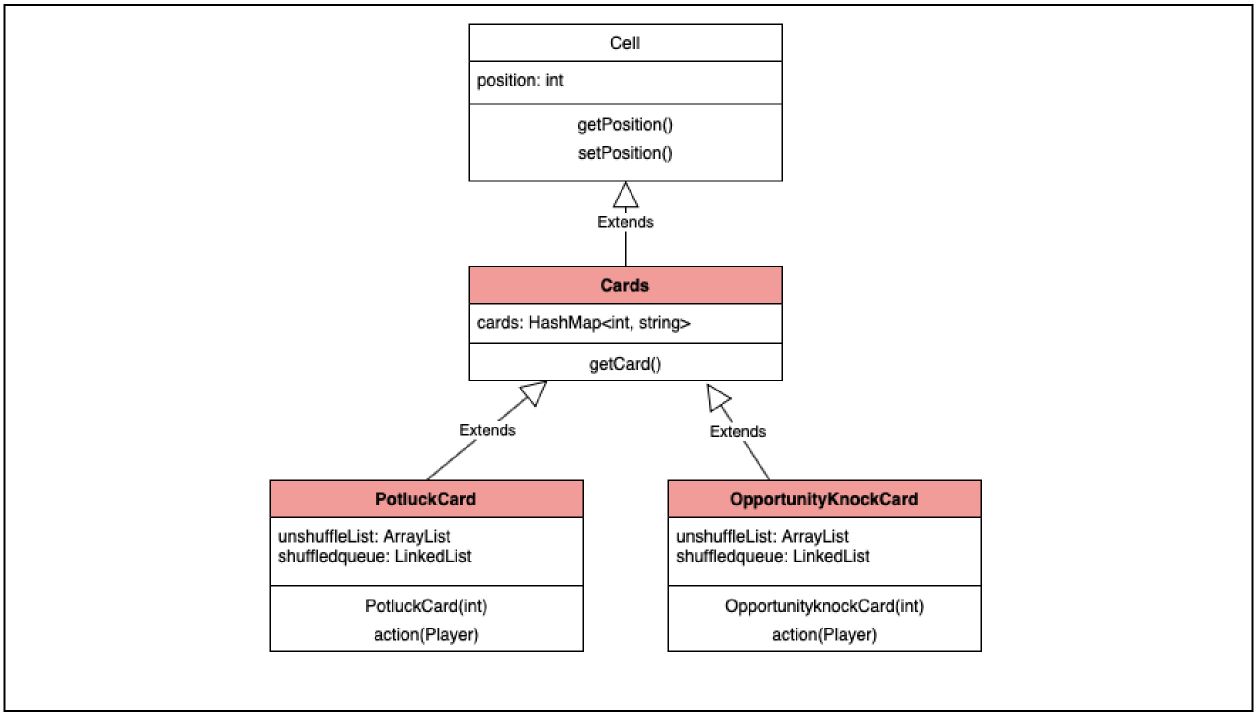
Class diagrams:



Sprint 2


Cell
position: int
getPosition()
setPosition()
Extends
Player
money: int
location: int
token: Token
passGo: boolean
name: String
releaseCard: int
Properties: ArrayList<Property>
totalvalue: int
rollDices()
payReleased()
released()
addReleaseCard()
buyProperty(Property)
sellProperty(Property)
sellPropertyToPlayer(Player, Property, int)
raiseMoney(int)
payRent(Property, Player)
getMoney()
addMoney()
minusMoney()
getLocation()
setLocation()
isPassGo(boolean)
getToken()
getName()
getPropertyList()
Property
name: String
group: String
cost: int
rent: int
improvedRent: int
improvedRents: int[]
available: boolean
hotelBuilded: boolean
numOfHouse: int
status: int
owner: Player
location: int
getName()
sell(Player)
isAvailable()
getRent()
getCost()
getOwner()
getGroup()
getNumOfHouse()
ifHotelBuilded()
buildHouse()
sellHouse()
buildHotel()
sellHotel()
changeOwner()
setStatus()

Sprint 2



6) Test plan and evidence of testing

TC2: Buy Properties

TC2-F1: Player shall be able to buy a property

System test:

Build Bank

Build a player

Add a property to the bank

Call buyProperty()

Call getOwner() in the Property Class to check if the owner has changed to the player

Expected output: the property owned by the player

Result: Passed

TC2-F3: Before a player buys a property, the amount of money the player shall be compared with property price in order to check whether the player has enough money

System test 1:

Build Bank

Build a player

Add a property to the bank

Call minusMoney(1500) to subtract all the initial balance from the player

Call buyProperty()

Expected output: the PropertyException

Result: Passed

Sprint 2

System test 2:

Build Bank
Build a player
Add a property to the bank
Call buyProperty()
Expected output: no PropertyException
Result: Passed

TC2-F4: When a player has enough money to buy property, the system shall change the owner of the property.

Result: Tested in TC2-F1.

TC2-F5: The amount of money for the property shall be transferred from the player's bank account to the bank.

System test:

Build Bank
Build a player
Add a property to the bank
Call buyProperty()
Call getBalance()
Call getMoney() in the Player Class to check if the price has subtracted from the balance
Expected output: the bank's original balance plus the price and the player's original balance minus the price
Result: Passed

***TC5-F2&&F3*: If a player takes the card "get out of jail", the player shall keep this card to use it later. If player uses its "get out of jail" card, the card shall be placed on the bottom of the corresponding pile and the player moves to "just visiting"**

System test1:

Initialize game
Create a player 'player1' with token boot
Add a released card from potluck cards for player1
Put player1 into jail
Player1 uses card to get himself out of jail
Expected output: Ensure player1 does not have more released cards.
Ensure "get out of jail" card is placed on the bottom of the corresponding potluck

System test2:

Initialize game
Create a player 'player1' with the token boot
Add a released card from opportunity cards for the player1
Put player1 into jail
Player1 uses the card to get himself out of the jail
Expected output: Ensure player1 does not have more released cards.

Ensure “get out of jail” card is placed on the bottom of the corresponding opportunity
Ensure player 1 is no longer at Jail.

Player is not in jail but tries to use “get out of jail” card
System test3:
Initialize game
Create a player ‘player1’ with the token boot
Add a released card from opportunity cards for the player1
Player1 uses the card to get himself out of the jail
Expected: NotInJailException

***TC5-F4*: If a player throws a double at the third turn, its turn ends immediately and the token shall be moved to jail**
System test1:
Set times of double as 3
Roll dice
Ensure player1 is put into jail and turn ends up immediately.

***TC5-F5*: Release player from jail after the player has paid £50. && *Sprint 3 TC3-F1*: The paid fine should be compared with the stated number on the special card / when getting out of jail**
System test1:
Put player1 into jail
Pay 50$ to release himself
Ensure to add 50 to the collected fine in the park.
Player`s money is minus by 50.
Player`s turn ends up.
Player is no longer in jail.

When player does not have enough money but wants to pay 50$
System test2:
Put player1 into jail
Pay 50$ to release himself
Expected: LackMoneyException.

***TC5-F6*: Release player from jail after they gave up its turns for 2 rounds**
System test1:
Put player1 into jail
Roll dice(pass one turn immediately)
Roll dice(pass one turn immediately)
Ensure when the player is in jail and it is player1`s turn, he only can use the released card. or pay money function. Otherwise, he rolls dice but passes 1 turn.

Board test1:
Initialize game
Build board
Build a Property p1

Sprint 2

Put Property in board with position 1
New a Property p2
Call get method with position 1
Make Property 2 equal it
Check if they have same attribute(location,name,owner)
Expected output:p1 and p2 get same result
Result:passed

Board test2:

Initialize game
Read csvfile to auto create board.
Create new Property p2
Set p2 to be any Property from board
Check if it has same attribute(location,name,owner)in csv file
Expected output:p2 has same attribute(location,name,owner)in csv file
Result :passed

Board test3:

Initialize game
Read csvfile to auto create board.
Add new player ,player 1
Create new Jail jail
jail equal board.getcell(11)
Put player1 in jail
Expected output:player1 in jail
Result :passed

CentralControlTest1:

Initialize game
Read csvfile to auto create board.
Create Potluck pl
Make pl equal getcell(3) in board
pl.action()
Run 15times make sure each case is tried
Result:passed

CentralControlTest2:

Initialize game
Read csvfile to auto create board.
Create opportunityknocksCard OP
Make OP equal getcell(8) in board
pl.action()

Run 15times make sure each case is tried
result:passed

7) Summary of sprint

Due to the corona virus situation and the abrupt changes in teaching (incl. travelling back home), we have extended the (originally planned one week) Sprint to two weeks.

Learnings from Sprint 2:

- **Communication** should be improved:
  - More daily reports to share our current work status (Summary Channel in Discord)
  - One call per week is enough to do the Sprint reflection
- More **deadlines**:
  - We set deadlines for our tasks (during the Sprint reflection call)
  - Team Leader is responsible for checking whether the deadlines are achieved and asks for status updates
- Establish common pattern when **changing code** so that the others know what has changed, who has changed it and why
- **Standard for writing comments** within code and **documentation** including how exactly methods work, especially when changing other's code/class

Status of Task Cards:

Pot luck / opportunity cards → Complete
Buying properties → Complete (Property and Bank Class)
Jail → Complete
Connecting Classes → Incomplete, must be pushed to the next sprint.

Additional open questions:

- **Jail class** to store people who are in jail and release people
- Should we have a **bank class**? → already implemented (at least parts of it)
- "I think it would be better if i'm responsible to class design and you guys tests, GUI and so on? maybe i need a person to help me" → We should design the classes as a group, so we don't miss any features etc
- Change the **Board class** so it loads the properties from the excel document rather than hard-coded