Sprint 1

1) **Summary**	
Sprint leader(s)	Vivien
Sprint start date	27/02/2020
Sprint end date	04/03/2020 (Overran to 12/03/2020)

2) **Individual key contributions**	
Team member	Key Contribution(s)
Neumann, Vivien	Task Cards, Requirement Analysis + Planning, Use Case Diagrams
Jiao, Haotian (Hallton)	Central Control
Wang, Mingfeng (Foret)	Player
Banes, Hayden J	Dice
Tang, Zhenyu (tang)	Board

3) User stories / task card

User Story:
The game Property Tycoon is for 2-6 players. Each player is assigned one of the game tokens. The tokens are: boot, smartphone, goblet, hatstand, cat and spoon. Each player takes a turn by rolling two dice to determine how they move around the board. At the outset, all players start on the board space labelled Go and move clockwise around the board. The board spaces may consist of properties, a “pot luck” space, an “opportunity knocks” space, “free parking”, the jail/just visiting space or a space with specific instructions that must be followed by the player. All of the space content and its detailed information (such as rent, housing development costs, etc.) is stored in a xls-file and in the beginning of the game, the board model needs to fetch the data automatically.

Task Card 1: Player / Token
Priority: 2
Value: 10

In the beginning of the game, each player chooses a token and enters their name. This information shall be stored together with its current position on the board which has to be available at every time of the game.

Task Card 2: Board
Priority: 1
Value: 10
The board is the basis of the game Property Tycoon. The square board shall be generated (Hash Map) and its model shall fetch data on the content of each board square from a configuration file (xls-file is on canvas). This data is stored in the corresponding space. By loading the data from a separate file before each game, the content of the squares can be changed easily.

Task Card 3: Dice Roller
Priority: 3
Value: 10
In order to determine how many spaces a player is allowed to move, the player has to roll two dice. Therefore, the implementation of a dice roller is essential. It rolls two dice randomly and adds the two numbers. Additionally, it checks whether a player has thrown a double. If so, the user takes another turn. However, when a player throws a third double in a row, they go to jail. The jail part is not necessary at this development stage, but the dice roller shall already include the "double check".

Task Card 4: Central Control
Priority: 1
Value: 10
The class central control should control the main features and keep the game running. For now, the main features of central control is to be able to load and save the players and to determine who's turn it is.

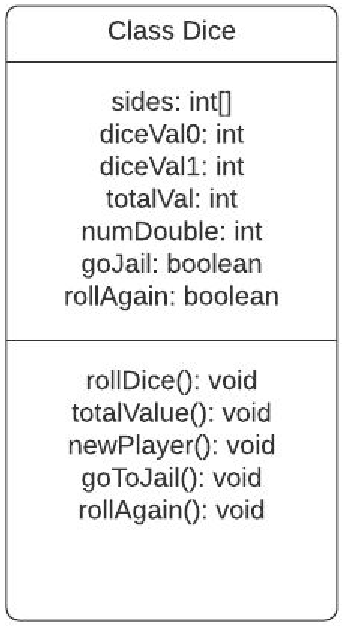
For all task cards, the priority and value is assessed. The priority is measured based on a scale of 1 to 5 where 1 denotes the highest priority and 5 the lowest. In each sprint planning meeting, we decide which tasks currently have the highest priority for us in order to have a relevance ranking of the planned tasks.
The measurement value is evaluated based on a scale of 10 to 1. 10 indicates an essential feature which is crucial for the game simulation and which has a lot of dependencies and 1 denotes a nice-to-have feature which has no additional benefit for the game in general.

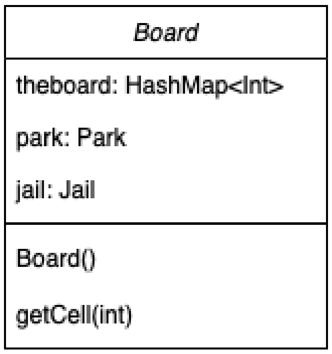
4) Requirement analysis

Sprint 1

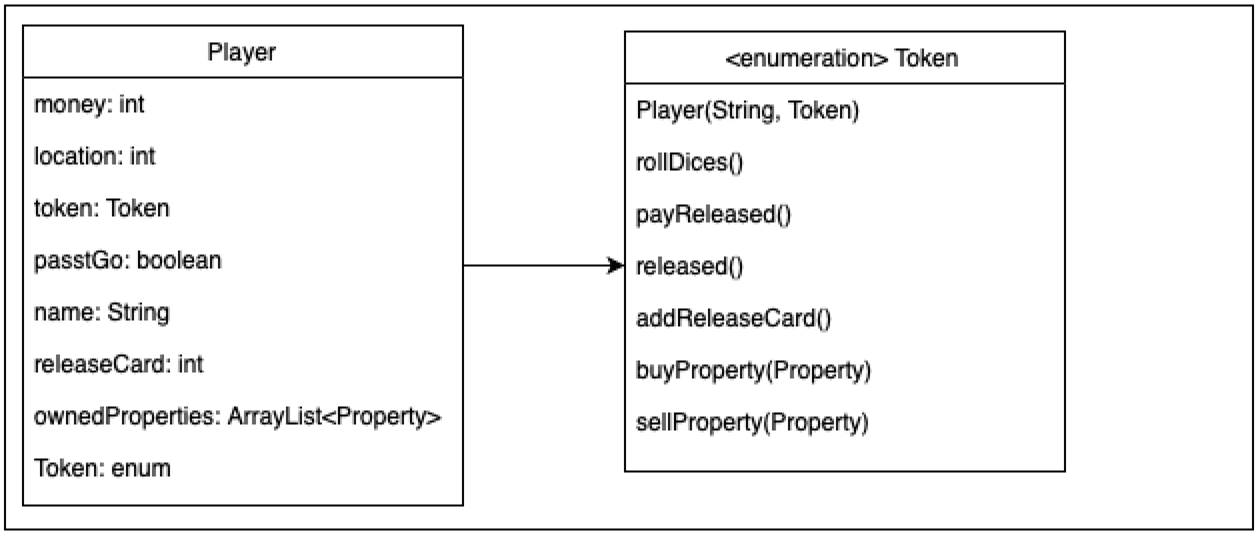
	Requirements
TC1: Player	TC1-F1: Minimum of 2 players TC1-F2: Maximum of 6 players TC1-F3: Each player has a name and a token TC1-F4: Give back the current position of a player
TC2: Board	TC2-F1: A Go point is needed for the players to start the game TC2-F2: The board shall automatically fetch the board data from the configuration file TC2-F3: The board spaces shall contain the content of the configuration file
TC3: Dice Roller	TC3-F1: Rolling two dices each turn TC3-F2: Add the two thrown numbers TC3-F3: The thrown number has to be valid and an integer TC3-F4: Give a notification when a player has thrown three doubles in a row
TC4: Central Control	TC4-F1: Determine who's turn it is

5) Design





Sprint 1



6) **Test plan and evidence of testing**
@Before SetUp() Initialize game Create a player 'player1' with token boot **TC1-F3: Each player has a name and a token:** System test1: Create another player 'player1' with token phone Expected output: CreatePlayerException: Duplicate name: player1 System test2: Create another player 'player2' with token boot Expected output: CreatePlayerException: Duplicate token: boot **TC4-F1: Determine who's turn it is** System test: Build CentralControl Build some players Add all players Call nexPlayer() Call getCurrentPlayer() Expected output: the second player that added to the system **Result: Passed** **TC5 - F1 Roll Dice**

Create new Dice
Roll the dice
Assertion: the value of dice 1 and dice 2 is within the range 1 - 6
Assertion: the value of dice 1 and dice 2 is equal to the total value
Repeat 10 times

TC5 - F2 If the player rolls a double they can roll again

Create new dice
Roll the dice with values manually set to be the same
Assertion: roll again is true (the player can roll again)
Roll the dice with values manually set to be different
Assertion: roll again is false (the player cannot roll again)
Create new dice
Roll the dice with values manually set to be different
Assertion: roll again if false (the player cannot roll again)
Create new dice
Roll the dice with the values manually set to be the same
Assertion: roll again is true (the player can roll again)
Roll the dice with the values manually set to be the same
Assertion: roll again is false (the player can roll again)

TC5 - F3 If the player rolls too many doubles, they should go to jail

Create new dice
Manually set the number of doubles to two
Roll the dice with each manually set to be the same
Assertion: goJail is true, instructing the game that the player should go to jail

TC5 - F4 When there is a new player, the dice should be reset

Create new dice
Roll the dice
Call new player method
Assertion: all dice variables should be reset to their default values

CentralControlTest1:

Initialize game
Build CentralControl normal
Add new player ,player 1
Add new player,player 2
Add new player ,player 3
Print players list by call getPlayers
Try to call method firstrol
Print players list by call getPlayers
Expected output:the order of player in player list may change if test is ran again.

CentralControlTest12:

Initialize game
Build CentralControl normal
Add new player ,player 1
Add new player,player 2
Print player 1 and player 2 location
Player 1 rollDice
Call nextplayer
Player 2 rollDice
Print player 1 and player 2 location
Expected output:players moved in their round
Result :passed

7) Summary of sprint

Learnings Sprint 1:

- Delay caused by some confusions → more detailed planning and assigning the tasks
- Failed to implement import data from excel file to java document function.
- No working prototype delivered due to inadequate planning
- Task cards were helpful
- Some are inexperienced with GitHub
- Unforeseen issues with Java/Unity functionality, caused delays in sprint