Gameplay

- board: Board
- players: List<Player>
- RoundNum: int
- maxRounds: int

+ runGame(): void
+ playRound(): void
+ isGameOver():
+ printPoints(): void

TurnController

- board: Board
- players: List<Player>
- CurrentPlayerIndex: int

+ startGame()
+ nextTurn()
+ handleBuildRoad(player, n1, n2)
+ handleBuildSett(player, node)
+ handleUpgradeCity(player, node)

*

p: player
n1, n2 → node

Board

- tiles: List<Tile>
- nodes: List<Node>
- roads: List<Road>

+ boardSetup(): void

~~+ validPlacement (node : Node) : boolean~~

+ addRoad (road : Road) : boolean → placeRoad(player, n1, n2)

+ add Settlement (node:Node, player:Player):boolean → placeSettlement (player, node):b

+ City Upgrade (node: Node): boolean → Upgrade to City (player, node):b

b → boolean

* using cards to upgrade building instead of adding a new one (needs to replace node with settlement with a city)

Player

~~- id : int~~

- resources : Map<Resource, Integer>

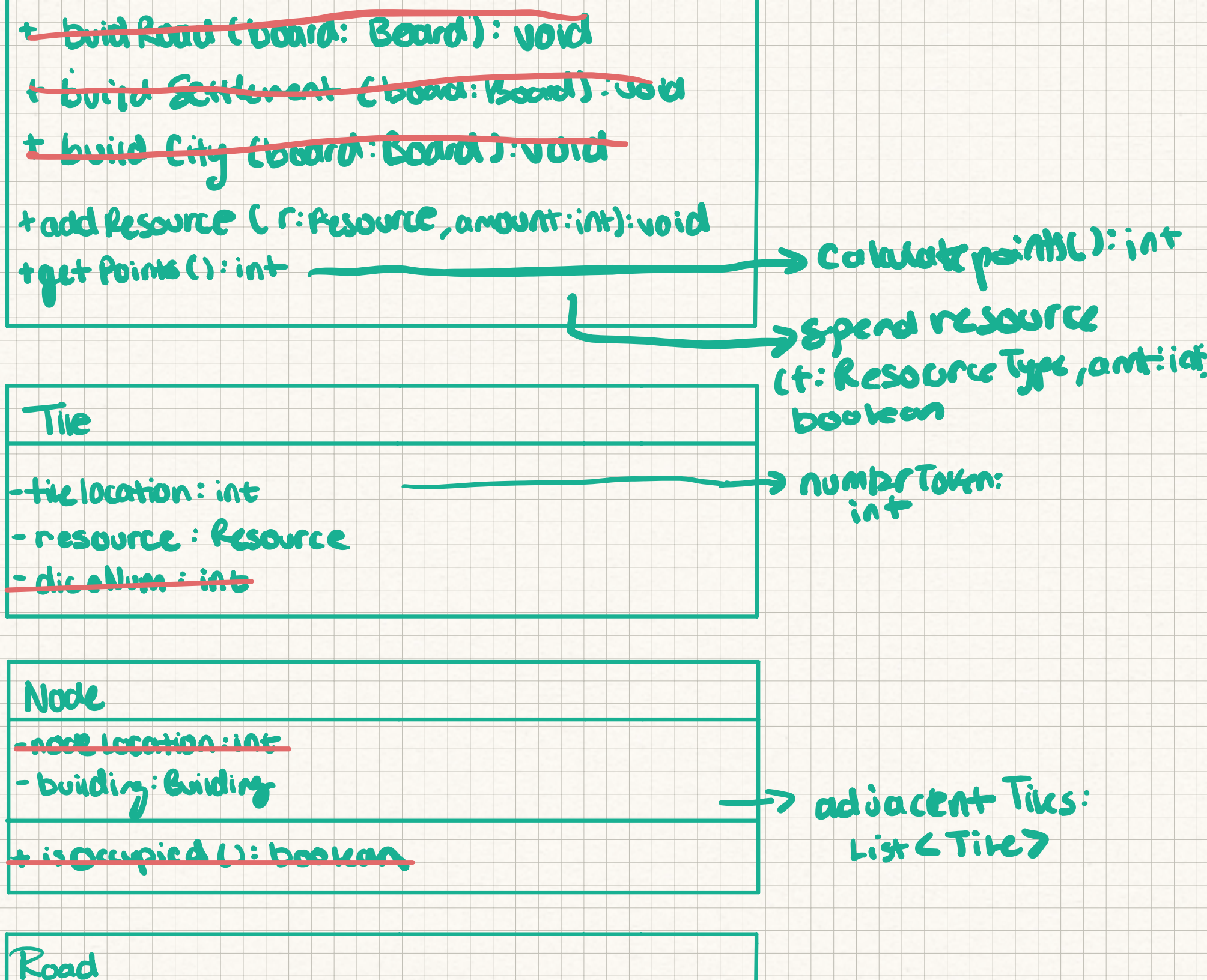
~~- points : int~~

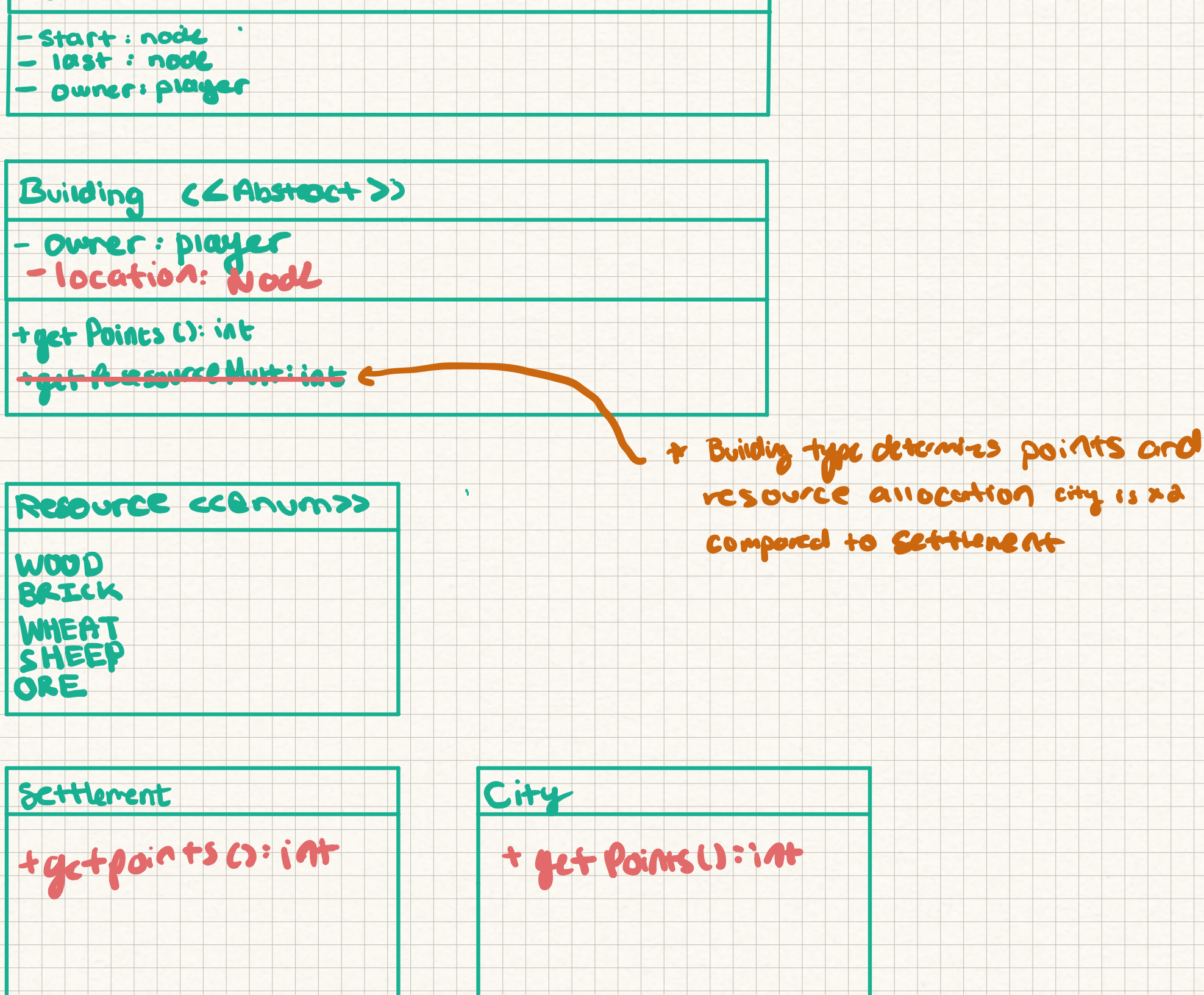
- roads : List<Road>

- buildings : List<Building>

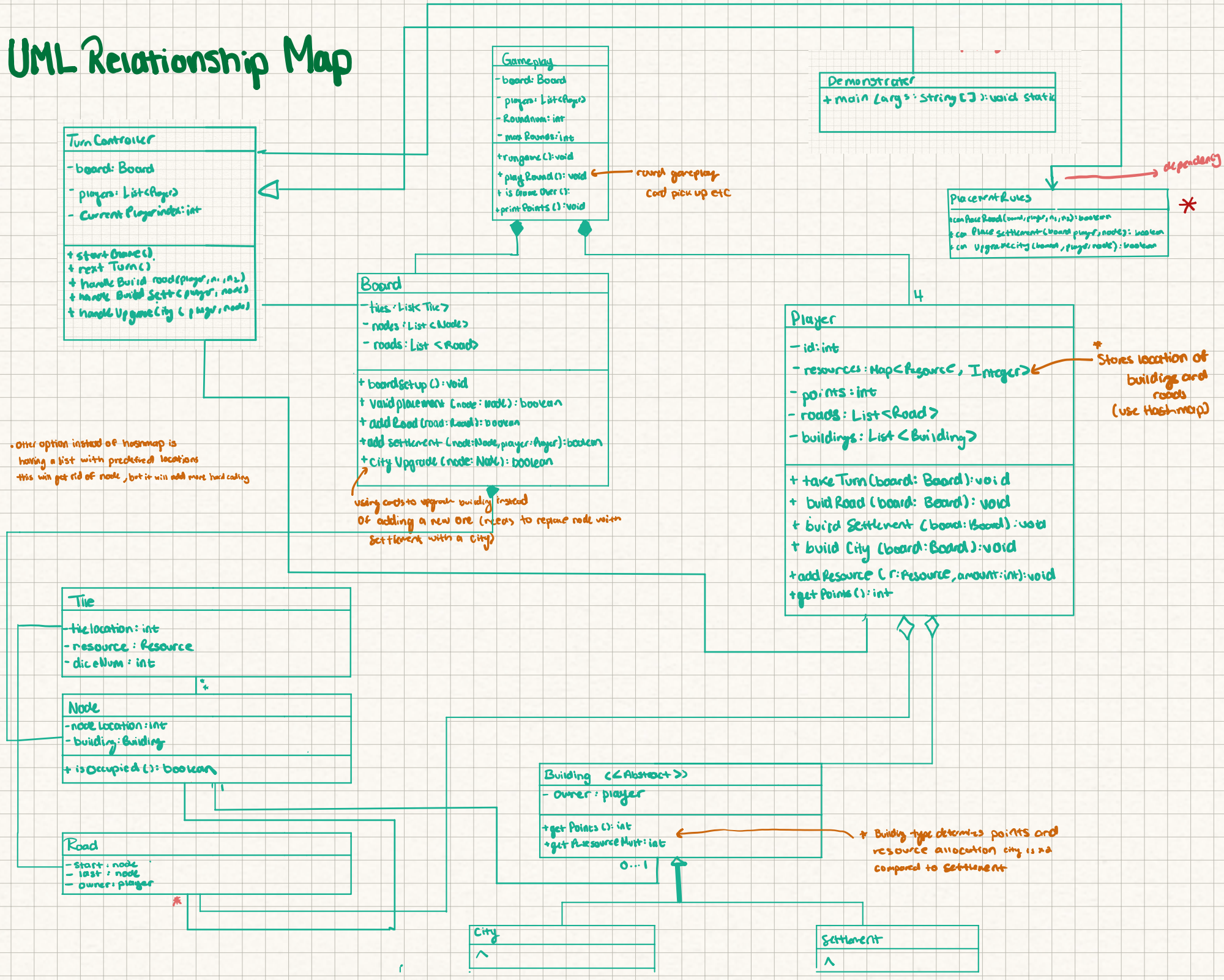
* Stores location of buildings and roads (use HashMap)

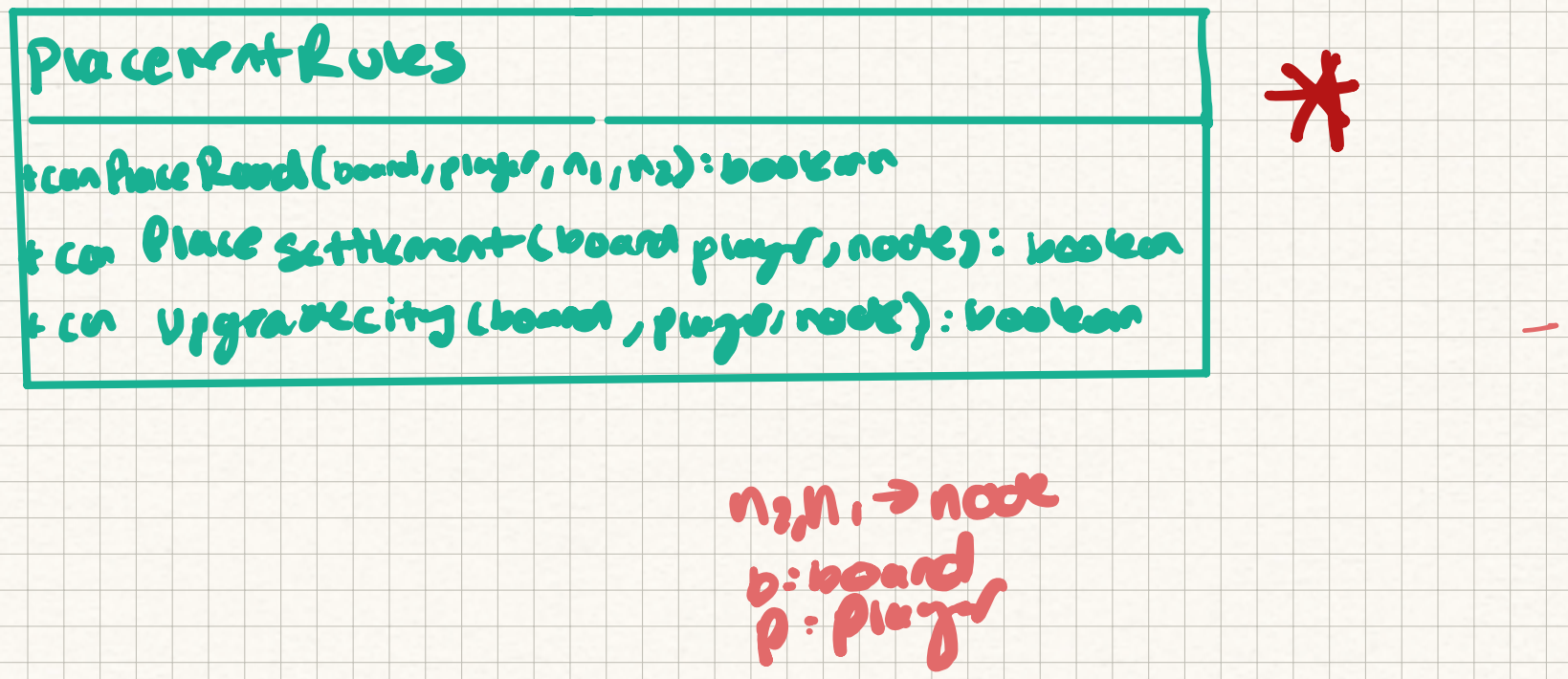
~~+ takeTurn (board: Board) : void~~


+ buildRoad(board: Board): void
+ buildSettlement(board: Board): void
+ buildCity(board: Board): void
+addResource(r: Resource, amount: int): void
+getPoints(): int
calculatePoints(): int
spendResource(t: ResourceType, amt: int): boolean
Tile
-tileLocation: int
- resource: Resource
- diceNum: int
numberTown: int
Node
- nodeLocation: int
- building: Building
+ isOccupied(): boolean
adjacentTiles: List<Tile>
Road


- start : node
- last : node
- owner: player
Building <<Abstract>>
- owner : player
- location: Node
+get Points (): int
+get ResourceMult: int
* Building type determines points and resource allocation city is x2 compared to settlement
Resource <<enum>>
WOOD
BRICK
WHEAT
SHEEP
ORE
Settlement
+getpoints (): int
City
+ get Points(): int

UML Relationship Map




PlacementRules
+canPlaceRoad(board, player, n1, n2): boolean
+canPlaceSettlement(board, player, node): boolean
+canUpgradeCity(board, player, node): boolean
n2,n1 → node
b: board
p: player
Demonstrator
+main(args: String[]): void static