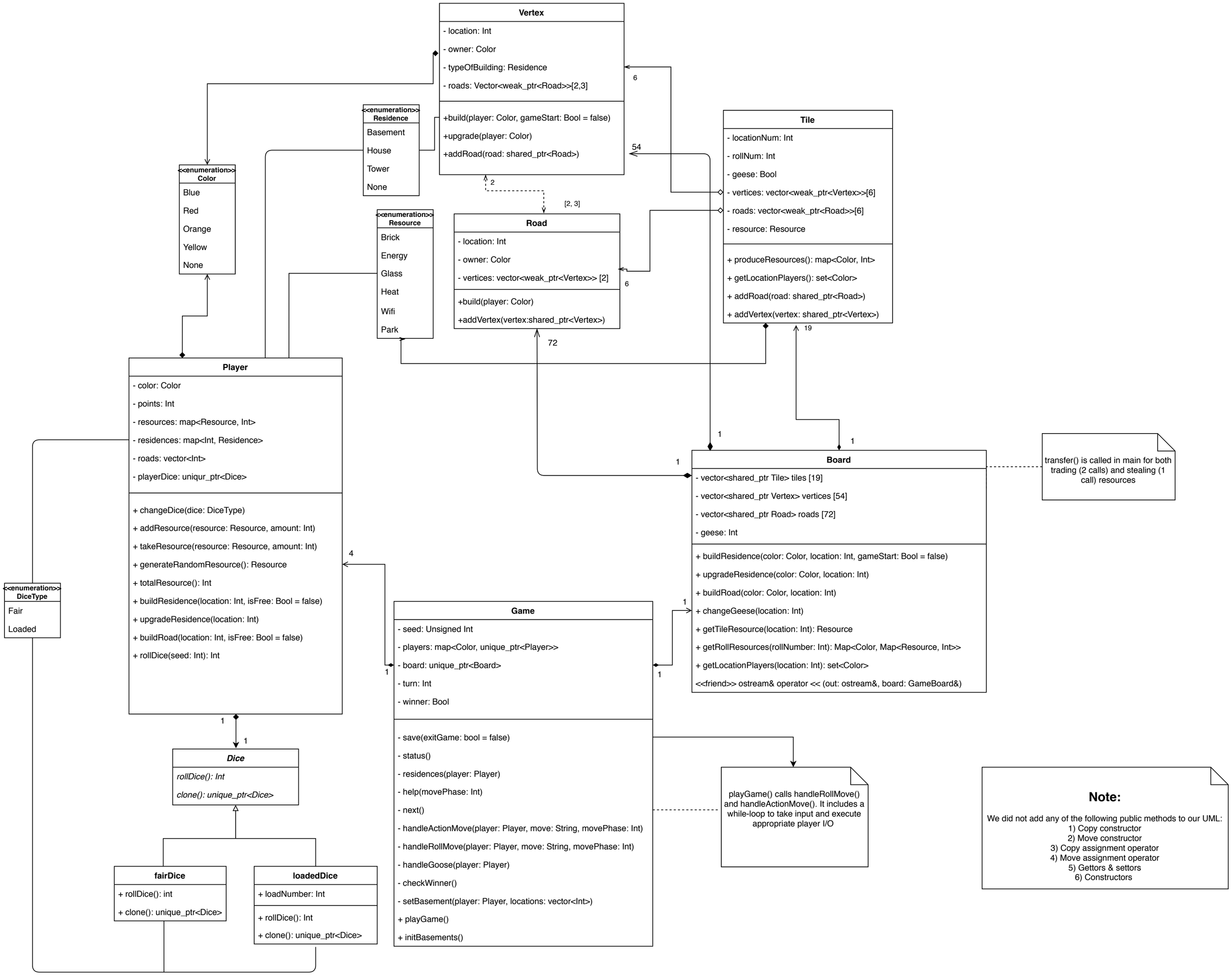
Vertex
- location: Int
- owner: Color
- typeOfBuilding: Residence
- roads: Vector<weak_ptr<Road>>[2,3]
+build(player: Color, gameStart: Bool = false)
+upgrade(player: Color)
+addRoad(road: shared_ptr<Road>)
<<enumeration>>
Residence
Basement
House
Tower
None
<<enumeration>>
Color
Blue
Red
Orange
Yellow
None
<<enumeration>>
Resource
Brick
Energy
Glass
Heat
Wifi
Park
Road
- location: Int
- owner: Color
- vertices: vector<weak_ptr<Vertex>> [2]
+build(player: Color)
+addVertex(vertex:shared_ptr<Vertex>)
Tile
- locationNum: Int
- rollNum: Int
- geese: Bool
- vertices: vector<weak_ptr<Vertex>>[6]
- roads: vector<weak_ptr<Road>>[6]
- resource: Resource
+ produceResources(): map<Color, Int>
+ getLocationPlayers(): set<Color>
+ addRoad(road: shared_ptr<Road>)
+ addVertex(vertex: shared_ptr<Vertex>)
Player
- color: Color
- points: Int
- resources: map<Resource, Int>
- residences: map<Int, Residence>
- roads: vector<Int>
- playerDice: uniqur_ptr<Dice>
+ changeDice(dice: DiceType)
+ addResource(resource: Resource, amount: Int)
+ takeResource(resource: Resource, amount: Int)
+ generateRandomResource(): Resource
+ totalResource(): Int
+ buildResidence(location: Int, isFree: Bool = false)
+ upgradeResidence(location: Int)
+ buildRoad(location: Int, isFree: Bool = false)
+ rollDice(seed: Int): Int
Board
- vector<shared_ptr Tile> tiles [19]
- vector<shared_ptr Vertex> vertices [54]
- vector<shared_ptr Road> roads [72]
- geese: Int
+ buildResidence(color: Color, location: Int, gameStart: Bool = false)
+ upgradeResidence(color: Color, location: Int)
+ buildRoad(color: Color, location: Int)
+ changeGeese(location: Int)
+ getTileResource(location: Int): Resource
+ getRollResources(rollNumber: Int): Map<Color, Map<Resource, Int>>
+ getLocationPlayers(location: Int): set<Color>
<<friend>> ostream& operator << (out: ostream&, board: GameBoard&)
transfer() is called in main for both trading (2 calls) and stealing (1 call) resources
<<enumeration>>
DiceType
Fair
Loaded
Game
- seed: Unsigned Int
- players: map<Color, unique_ptr<Player>>
- board: unique_ptr<Board>
- turn: Int
- winner: Bool
- save(exitGame: bool = false)
- status()
- residences(player: Player)
- help(movePhase: Int)
- next()
- handleActionMove(player: Player, move: String, movePhase: Int)
- handleRollMove(player: Player, move: String, movePhase: Int)
- handleGoose(player: Player)
- checkWinner()
- setBasement(player: Player, locations: vector<Int>)
+ playGame()
+ initBasements()
playGame() calls handleRollMove() and handleActionMove(). It includes a while-loop to take input and execute appropriate player I/O
Dice
rollDice(): Int
clone(): unique_ptr<Dice>
fairDice
+ rollDice(): int
+ clone(): unique_ptr<Dice>
loadedDice
+ loadNumber: Int
+ rollDice(): Int
+ clone(): unique_ptr<Dice>
2
[2, 3]
6
54
6
72
19
1
1
1
1
4
1
1
1
1
1
Note:
We did not add any of the following public methods to our UML:
1) Copy constructor
2) Move constructor
3) Copy assignment operator
4) Move assignment operator
5) Gettors & settors
6) Constructors