Ankara Yıldırım Beyazıt University
Department of Computer Engineering

CENG 201 – Object Oriented Programming
Course Project

G18: Catan Game

Design Report

Eren Yılmaz 22050151024,
Muhammed Enes Uğraş 23050151030,
Rasin Taha Yılmaz 22050111056,
Sühan Sarı 18050111015

Instructor: Muhammed Abdullah Bülbül
Teaching Assistant: Elif Şanlıalp, Yusuf Şevki Günaydın
Date: 12/12/2024

Table of Contents

1. Introduction

This report presents the class design and Class-Responsibility-Collaborator (CRC) cards for developing a digital version of Catan using C++. It explores the system architecture, detailing essential classes and their interactions. Emphasizing modularity and scalability, the design ensures that Catan's strategic gameplay and complex mechanics are effectively and reliably reproduced in a software environment.

2. Class-responsibility-collaboration (CRC) cards

1. Catan	
Responsibilities • Initialize and manage gameplay elements • Manage players' turns • Determine game end conditions	**Collaborations** • GameState • Board • Player • Bank • Deck • Dice • TradeManager

2. GameState

Responsibilities

- Store each game state
- Enable undo/redo functionality

Collaborations

- Catan
- Board
- Player
- Bank
- Deck
- Dice
- TradeManager

3. Board

Responsibilities

- Represent board structure
- Provide robber event management
- Manage hex, edge, and vertex for building

Collaborations

- Hex
- Edge
- Vertex
- Harbor

4. Player

Responsibilities

- Manage player's resources and development cards
- Provide building actions and card actions
- Track victory points, longest road, and largest army

Collaborations

- DevelopmentCard
- Settlement
- City
- Road
- Bank
- Harbor
- TradeManager
- ResourceType

5. Bank

Responsibilities

- Manage resources
- Handle player-to-bank trades

Collaborations

- Player
- ResourceType

6. Deck

Responsibilities

- Manage development cards
- Allow players to buy development cards

Collaborations

- DevelopmentCard
- Player

7. Dice

Responsibilities

- Simulate dice rolls
- Return a number between 2-12

Collaborations

- Catan

8. TradeManager

Responsibilities

- Manage trade between players

Collaborations

- Player
- ResourceType

9. Hex

Responsibilities

- Store resource type, terrain type, and dice number
- Represent terrain tiles
- Store adjacent edges and vertices

Collaborations

- Board
- Player
- Edge
- Vertex
- ResourceType
- TerrainType

10. Edge

Responsibilities

- Store information about building roads
- Represent connection between vertices
- Store adjacent vertices

Collaborations

- Board
- Vertex
- Road

11. Vertex

Responsibilities

- Store information about building settlements or cities
- Represent points where edges meet

Collaborations

- Board
- Building
- Settlement
- City

12. Harbor

Responsibilities

- Provide trading opportunities for players
- Enforce special trade ratios

Collaborations

- Vertex
- Player
- ResourceType

13. Building

Responsibilities

- Serve as base class for Settlement and City

Collaborations

- Player
- Settlement
- City

14. Settlement

Responsibilities

- Represent a player's settlement on the board

Collaborations

- Player
- Vertex

15. City

Responsibilities

- Represent a player's city on the board
- Upgraded version of Settlement

Collaborations

- Player
- Vertex

16. Road

Responsibilities

- Represent a player's road on the board

Collaborations

- Player
- Edge

17. DevelopmentCard

Responsibilities

- Serve as base class for development cards
- Allow players to use or purchase development cards

Collaborations

- Player
- KnightCard
- VictoryPointCard
- RoadBuildingCard
- MonopolyCard
- YearOfPlentyCard

18. KnightCard

Responsibilities

- Allow player to move robber
- Included in largest army calculation

Collaborations

- Player

19. VictoryPointCard

Responsibilities

- Provide 1 victory point to player

Collaborations

- Player

20. RoadBuildingCard

Responsibilities

- Allow player to build two roads for free

Collaborations

- Player
- Road

21. MonopolyCard

Responsibilities

- Allow player to get all of one type of resource from other players

Collaborations

- Player
- ResourceType

22. YearOfPlentyCard

Responsibilities

- Allow player to take any two resource cards from the bank

Collaborations

- Player
- Bank
- ResourceType

23. ResourceType

Responsibilities

- Define the type of resources available in game

Collaborations

- Player
- Hex
- Bank
- TradeManager
- Harbor
- MonopolyCard
- YearOfPlentyCard

24. TerrainType

Responsibilities

- Define the types of terrains for hex tiles

Collaborations

- Hex

3. Class Diagram

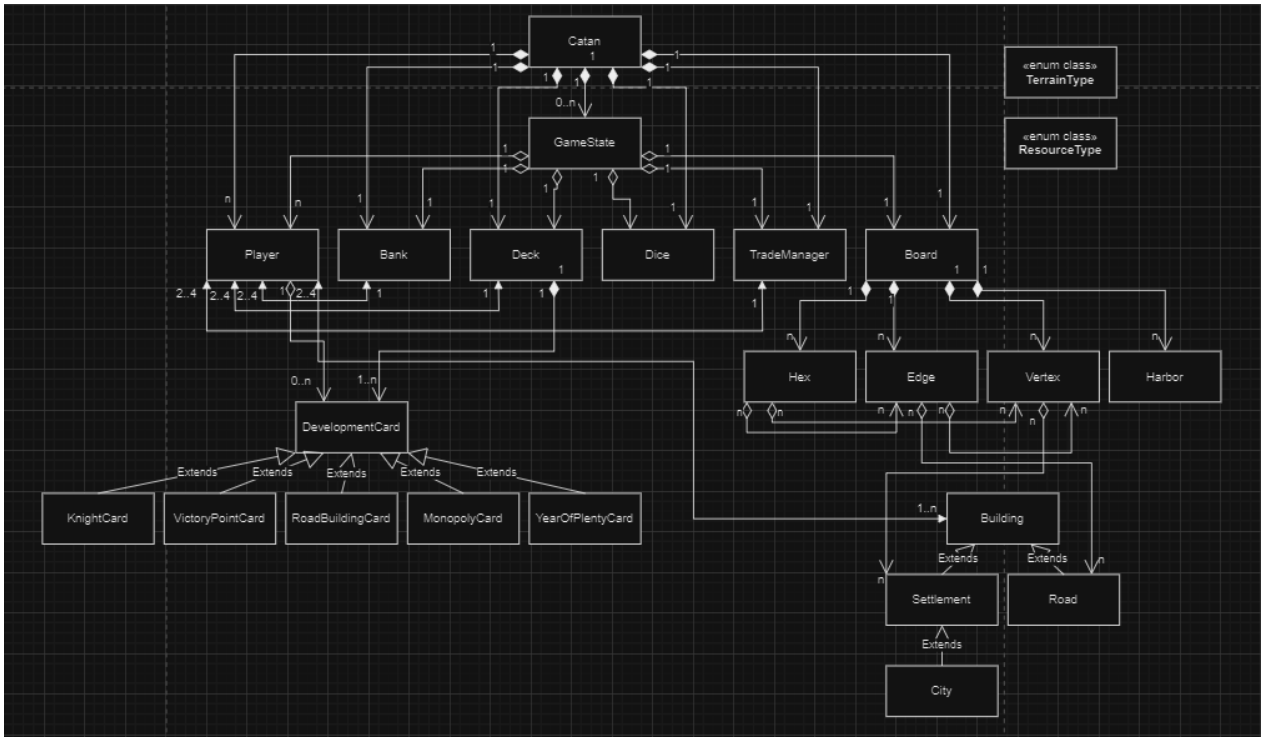


Figure-1 Class Design

Catan

- players : std::vector<Player>
- currentPlayer : Player
- board : Board
- deck : Deck
- dice : Dice
- bank : Bank
- tradeManager : TradeManager
- gameStates : std::vector<GameState>
- gameEnded : bool
- turn : int
- gui attribues(window, texture etc...)

+ Catan()
+ startGame() : void
+ saveState() : void
+ undoState() : void
+ redoState() : void
+ executeTurn() : void
+ endGame() : void
+ playCard() : void
+ rollDice() : void
+ startTrade() : void

GameState

- playersState : std::vector<Player>
- currentPlayerState : Player
- boardState : Board
- deckState : Deck
- diceState : Dice
- bankState : Bank
- turnState : int

+ GameState(playersState, currentPlayerState, boardState, deckState, diceState, bankState, turnState)
+ getPlayersState() : std::vector<Player>
+ getCurrentPlayerState() : Player
+ getBoardState() : Board
+ getDeckState() : Deck
+ getDİceState() : Dice
+ getBankState() : Bank
+ getTurnState() : int

Board

- hexes : std::vector<Hex>
- edges : std::vector<Edge>
- vertices : std::vector<Vertex>
- harbors : std::vector<Harbor>
- robberPlace : Hex
- gui attribues(window, texture etc...)

+ Board()
+ initBoard() : void
+ executeRobberAction() : void
+ placeBuilding(Building) : void
+ placeRoad(Road) : void
+ produceResource(int) : void
+ setHarborOwner() : void
+ getHarborAvailibity() : bool

Player

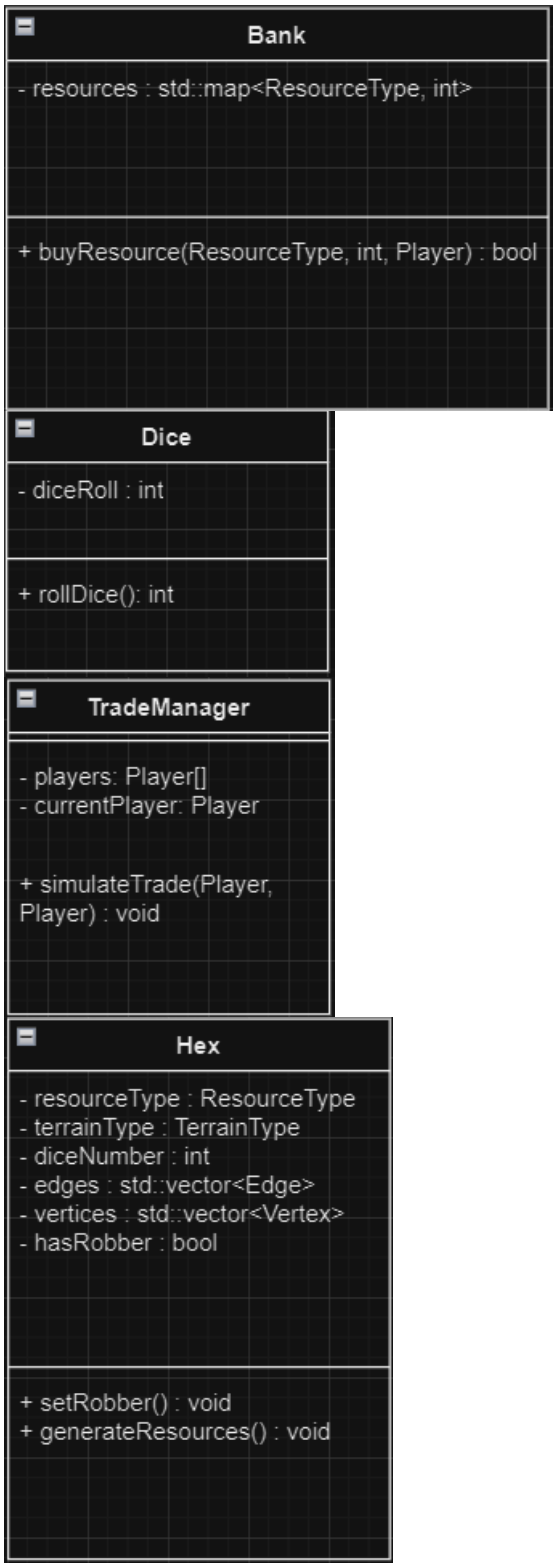
- name : std::string
- resources : std::map<ResourceType, int>
- developmentCards : std::vector<DevelopmentCard>
- settlements : std::vector<Settlement>
- cities : std::vector<City>
- roads : std::vector<Road>
- harbors : std::vector<Harbor>
- victoryPoints : int
- hasLongestsRoad : bool
- hasLargestArmy : bool
- knightCardsPlayed : int
- longestRoad : int
- color : int
- gui attribues(window, texture etc...)

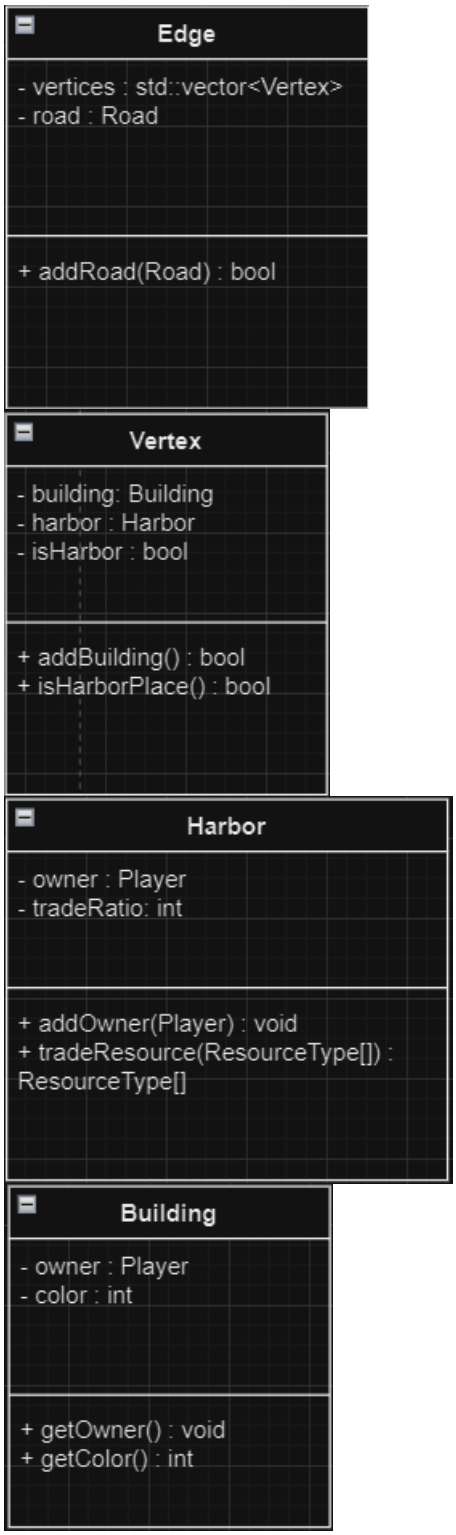
+ Player(name)
+ initPlayer() : void
+ addCard(DevelopmentCard) : void
+ addSettlement(Settlement) : void
+ addResource(ResourceType, int) : void
+ addRoad(Road) : void
+ getDevelopmentCards() : std::vector<DevelopmentCard>
+ playCard(DevelopmentCard) : void
+ updatePlayerState() : void
+ getLongestRoad() : int
+ getLargerstArmy() : int

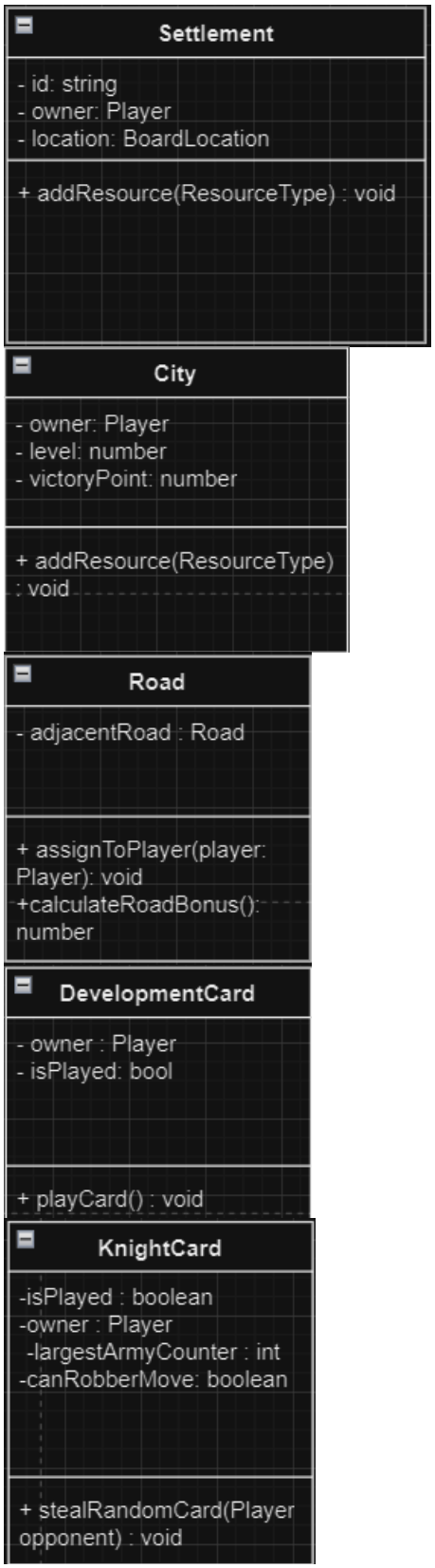
Deck

- developmentCards : std::vector<DevelopmentCards>

+ buyCard(DevelopmentCard, Player) : bool
+ shuffleDeck() : void


Bank
- resources : std::map<ResourceType, int>
+ buyResource(ResourceType, int, Player) : bool
Dice
- diceRoll : int
+ rollDice(): int
TradeManager
- players: Player[]
- currentPlayer: Player
+ simulateTrade(Player, Player) : void
Hex
- resourceType : ResourceType
- terrainType : TerrainType
- diceNumber : int
- edges : std::vector<Edge>
- vertices : std::vector<Vertex>
- hasRobber : bool
+ setRobber() : void
+ generateResources() : void


Edge
- vertices : std::vector<Vertex>
- road : Road
+ addRoad(Road) : bool
Vertex
- building: Building
- harbor : Harbor
- isHarbor : bool
+ addBuilding() : bool
+ isHarborPlace() : bool
Harbor
- owner : Player
- tradeRatio: int
+ addOwner(Player) : void
+ tradeResource(ResourceType[]) :
ResourceType[]
Building
- owner : Player
- color : int
+ getOwner() : void
+ getColor() : int


Settlement
- id: string
- owner: Player
- location: BoardLocation
+ addResource(ResourceType) : void
City
- owner: Player
- level: number
- victoryPoint: number
+ addResource(ResourceType) : void
Road
- adjacentRoad : Road
+ assignToPlayer(player: Player): void
+calculateRoadBonus(): number
DevelopmentCard
- owner : Player
- isPlayed: bool
+ playCard() : void
KnightCard
-isPlayed : boolean
-owner : Player
-largestArmyCounter : int
-canRobberMove: boolean
+ stealRandomCard(Player opponent) : void


VictoryPointCard
- victoryPoint : int
- getVictoryPoint() : int
RoadBuildingCard
- roadAmount : int
- getRoadAmount() : void
MonopolyCard
- owner : Player
+ executeCard(std::vector<Player>, Player, ResourceType) : void
YearOfPlentyCard
- resourceAmount : int
- gainResource(ResourceType) : int
«enum class»
ResourceType
+ LUMBER
+ BRICK
+ ORE
+ GRAIN
+ WOOL
«enum class»
TerrainType
+ FOREST
+ HILLS
+ MOUNTAINS
+ FIELDS
+ PASTURE
+ DESERT

4. Conclusion

This report details the development plan for the Catan console game, focusing on class design and Class-Responsibility-Collaborator (CRC) cards. By creating comprehensive CRC cards and detailed class diagrams, we established a clear architectural framework to guide the coding phase. We are a team of four people and we divide contents by four to contribute equally.