**Documentation**

**The Node Class**

Attributes:

1. private int id - special identifier represented by a number;
2. private List<Node> nodeNeighbours – a list that contains the neighborhood nodes;
3. private List<Node> HasRoadWith –a list that contains the nodes which the node has roads with;
4. private List<Hex> hexNeighbours – a list with the neighborhood hexes;
5. private bool hasPlayer – a bool variable which specifies if a node has a settlement from a player;
6. private bool playerSettled – the player which has the settlement on the node;
7. private bool hasSettlement - a bool variable which tells us if it has a settlement or not;
8. private Settlements settlementType – an object of type settlement which memorizes the type of settlement;
9. private bool hasPort - a bool variable which tells us if the settlement has a port or not;
10. private Ports portType – memorizes the type of the port;

Methods:

1. public Node (int id) – the constructor which initialize the id;
2. public override string ToString – a function that outputs all the data about the node;
3. Getters and Setters;

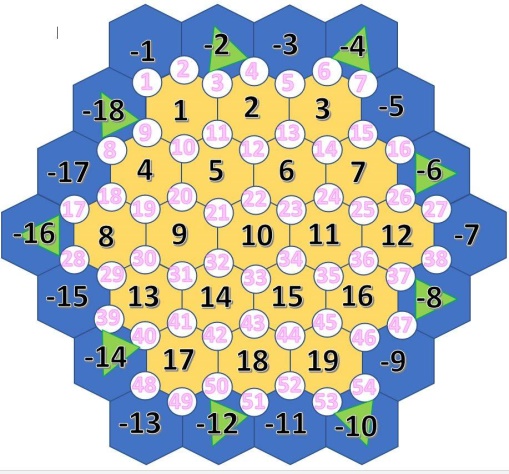
**Hex Class**

Attributes:

1. private int id – special identifier for each hex element;
2. private int number – the number on the top of each hexagon;
3. private bool hasRobber – bool variable that specifies if a robber is placed on the hex;
4. private List<Node> nodeNeighbors – list that contains all the node neighbors of the hex;
5. private List<Hex> hexNeigbords – list that contains all the hex neighbors;
6. private Resources resource – contains the resource specifics;

Methods:

1. public Hex (int id, Resources resource, int number) – initializes the id, resource, number, and hasRobber with value false;
2. public Hex (int id) – initializes the id of the hex;
3. public void setDetails – sets the resource and number to the given parameters;
4. public override string toString - a function that outputs all the data about the hex;
5. Getters and Setters;

**Map Class**

Attributes:

1. private List<Node> nodes - the list of nodes that belong to the map;
2. private List<Hex> hexes - the list of hexes that belongs to the map;
3. private static List<Resources> resourceTile – a list that contains the

resources found on the map;

1. private static List <Ports> typeOfPort - a list that contains the types of ports.

Methods:

1. public map – constructor that initializes the map and outputs the map;
2. private void initNodes – initializes 54 nodes in the nodes list;
3. private void initHexes – initializes 37 hexes on the in the hexes list(19 of them with positive values and the rest of them with negative values, for the ones on the edge of the map so we can build settlements on the edge of the map)
4. private void generateHexDetails – assigns resources for every land tile except the desert and randomly puts the tiles on the map; the ocean tiles are assigned ocean resources
5. private void addPortsToNodes – generates ports on the map; a normal port will follow a specific resource port and vice versa; the specific resource ports will be randomized;
6. private void addHexNeighborsToHexes, addNodeNeighborsToHexes, addHexNeighborsToNodes, addNodeNeighborsToNodes - hardcoded initialization of the map
7. Getters and Setters.

**Player Class**

Attributes:

1. private string name - name of the player
2. private int id - Id of the player
3. private int points - the number of points that the player accumulated
4. private List<Road> allRoads - the list of roads that the player built
5. private List<Road> settledNodes - the list of settlements that the player built
6. private int wheatQty, sheepQty, clayQty, stoneQty, woodQty - the number of resources cards of each type that the player has
7. private int villagesLeft, citiesLeft, roadsLeft - the number of settlements and roads the player can still build
8. private int knightCardsLeft , victoryPointCardsLeft, roadBuildingCardsLeft, yearOfPlentyCardsLeft, monopolyCardsLeft - the number of development cards the player has
9. private bool soldierCardsUsed - number of solider cards used by the player
10. private bool hasLongestRoad - bool variable that specifies if the player has the achievement for the longest road;

Methods:

1. public Player() – default constructor
2. public Player(string name) - constructor that initializes all the attributes that belong to the player
3. public overrideToString() - outputs the player name
4. public void useKnightCard(), useRoadBuildingCard(), useYearOfPlentyCard(), useMonopolyCard() – functions that implement the usage of the development cards;
5. Getters and Setters

**Gamestate Class**

Attributes:

1. private Map map - an object of type map
2. private List<Player> players - a list that contains the players which currently play the game
3. private int knightCardsLeft, victoryPointCardsLeft, roadBuildingCardsLeft, yearOfPlentyCardsLeft, monopolyCardsLeft - variables which contain the number of development cards of each type left;

Methods:

1. public GameState() – default constructor
2. public GameState(List<Player> players) - constructor which initializes the player list on the map and the number of development cards left;
3. Getters and Setters

**Road Class**

Attributes:

1. private Node node1,node2 - objects of type Node which belong to the road;
2. private Player owner – a Player type object that contains the player that owns the road

Methods:

1. Setters and Getters

**Ports Class -** class which contains an enumeration of types of ports (simple and resource-specific)

**Program Class -** Main class of the project

**Resources Class -** class which contains an enumeration of types of resources that the tiles contain (including dessert and ocean types of resources)

**Settlement Class** - class which contains an enumeration of types of settlements (village or cities);

**Shuffler Class** -Class that contains the shuffling method which puts the elements of a list on new, randomized positions .

