Final UML Diagram: Due Date 2 **BoardModel** theBoardView: BoardView* - tiles: Tile* [19] - gooseTile: int vertices: Vertex* [53] BoardView - edges: Edge* [71] Controller - builders: Builder* [4] - currBuilder: Builder* theBoardModel: BoardModel* - makeBlank(integer) : string - seed: integer initMethodCall: string - printVertex(Vertex*) : string - rng: random number generator fileName : string printEdge(Edge*): string seedValue : integer - edgeTileNumEdge(ostream, boolean, Edge*, Tile*, Edge*) : ostream - vertexEdgeVertex(ostream, boolean, Vertex*, Edge*, Vertex*): ostream - borderResourceBorder(ostream, boolean, Tile*, boolean, boolean, boolean, prepareBoard(): void + Controller() BoardModel*) : ostream - loadLayout(string, bool): void + playTurn(): void - vertexTileValVertex(ostream, const Vertex*, const Tile*, const Vertex*): loadBuilder(string, integer): void + startGame(): void - makeBuilderDataString(Builder*) : string + initBoard(): void - printTileVal(ostream, Tile*): ostream - makeBoardDataString(): string + initBoard(string) : void + initLoad(string) : void + BoardModel() + printStatus(Builder*[4]) : void + initRandomBoard(): void + initBoard(string) : void + printResidence(const Builder*) : void + setBoardSeed(int) : void + initRandomBoard(): void + printBoard(BoardModel*) : void + setSeedValue(int) : void + initLoad(string): void + printCurrBuilderTurn(const Builder*): void + setInitMethodCall(string) : void + save(string): void + printTradeResource(const Colour, const Colour, const ResourceType, const + setFileName(string): void + buildResidence(integer, bool) : void ResourceType): void + improveResidence(integer) : void + printWhereBuild(const Colour): void + obtainResources(integer): void + beginGameHelp(): void + buildRoad(integer) : void + duringGameHelp(): void + rollDice(): integer + playRoll(int): integer next(): void + prevBuilder() : void + checkWinner() : boolean + tradeResource(Colour, ResourceType, ResourceType): void + playGoose(): void 71 + printBoard(): void Edge Vertex + printResidences(): void + printCurrBuilderTurn(): void edgeNumber: integer - vertexNumber: integer + printStatus() : void - residence: Residence* hasRoad: boolean + beginGameHelp(): void - edges: integer[2...3] - owner: Builder* + duringGameHelp(): void vertices: integer[2] adjacentVertices: integer [2...3] + printTradeResources(const Colour, const ResourceType, const - adjacentEdges: integer[1..3] ResourceType) : void - cost: map<ResourceType, integer> + printWhereBuild(): void + Vertex(int) + buildResidence(Builder*, character, + getCurrBuilder() : Builder* + Edge(int) boolean): void + getDiceType(): character + buildRoad(Builder*, bool) : void + getVertexNum(): integer + getVertexPtr(integer): Vertex* + doesHaveRoad(): boolean + getResidence(): Residence* + getEdgePtr(integer): Edge* + getEdgeNum(): integer + getTilePtr(integer): Tile* + getOwnerColour(): Colour + getGooseTile(): Tile* + setDice(character) : void + setSeed(integer): void Colour ResourceType Residence + {BLUE, RED, ORANGE, YELLOW} + {BRICK, ENERGY, GLASS, HEAT, WIFI, PARK} - reward: integer - type: character owner: Builder* costToImprove: map<ResourceType, int> - cost: map<ResourceType, int> Residence(Builder*, character) improveResidence(int): void + getType(): character getOwner(): Builder* - tileNumber: integer getReward(): integer Builder resourceType: ResourceType getCost(): map<ResourceType, int> - value: integer - colour: Colour - hasGeese: boolean 19 points: integer vertices: integer[6] - resources: map<ResourceType, integer> edges: integer[6] - buildings: map<integer, character> - roads: integer[0..*] - dice: character + Tile(integer) + getTileNum(): integer + getResourceType(): ResourceType + getTileVal(): integer + Builder(Colour) + buildResidence(integer, character, map<ResourceType, integer>: + buildRoad(integer, map<ResourceType, integer>, boolean): void + updateResidence(integer, map<ResourceType, integer>, boolean): + takeResources(ResourceType, integer): void + haveResidence(integer): boolean + haveRoad(integer): boolean

+ checkResources(map<ResourceType, integer>): boolean

+ getResources(): map<ResourceType, integer>

+ getBuildings(): map<integer, character>

+ getColour(): Colour + getPoints(): integer

+ getRoads(): integer[0...*]
+ setDice(character): void

Key:

Red: Dhruv Mittal Blue: Adrian Seto Green: Omar Baranek