Eriantys Model Game GameID: int GameMode : GameMode <<enum>> - ListOfPlayers : Arrayl ist < Player > Board VerifyType - GameState : GameState Player <<enum>> Clouds : Arrayl ist < Cloud > NORMAI - Board : Board PlayerState Archipelago : Archipelago Nickname : String NumOfPlayers : int TWOPOINTS - Bag : ArravList < Student> NOTYOURTURN NOTOWER PlayerID : int VerifyType : VerifyType CoinsReserve : int PlayerState : PlayerState PLAYINGYOURTURN NOCOLOR - setState(State : GameState) : void - Plance : Plance FNDOFTURN - Characters : ArravList < Character > getState(): GameState Wizard · Wizard + getNumOfStudentsBag() : int addPlayer(Player : Player) : void Coins : int + getAndRemoveRandomBagStudent(NumOfStudents : int) : ArravList<Student> getGameMode() : GameMode - AssistantCards : ArravList < Assistant > <<enum>> + getClouds() : ArravList < Cloud> + qetNumOfPlayers() : int + getCoinsReserve() : int Wizard + getNickname() : String + winner() : Plaver + addCoinstoReserve(Coins : int) : void + getPlayerID(): int + verifyProfessorControll() · void WIZARDGREEN + removeCoinsFromReserve(Coins : int) : void <<enum>> + getPlayerState() : PlayerState + verifyIslandInflunce(Island : Island, VerifyTipe : VerifyType) : void WIZARDYELLOW + getCharacters() : ArrayList < Character> GameMode + setPlayerState(NextPlayerState : PlayerState) : void + setVerifyType(VerifyType : VerifyType) : void WIZARDPINK + addStudentsBag(Students : ArrayList <Student>) : void + getPlance() : Plance WIZARDBILLE + startTurn() : void SIMPLEMODE + getArchipelago() : Archipelago + getWizard() : Wizard + endTurn() : void EXPERTMODE + getCoins() : int + moveStudentToEntrance(PlayerID : int, Student : Student) : void + addCoin() : void + moveStudentToIsland(PlayerID: int, Island: Island, Student: Student): void + removeCoins(NumOfCoins : int) : void Archipelago + useAssistant(PlayerID : int, Assistant : Assistant) : void + getAssistantCards() : Arrayl ist < Assistant > + moveMotherNature(Island : Island) : void - Islands · Arrayl ist < Island> + removeAssistant(Assistant : Assistant) : void + useCharacter(PlayerID : int, Character : Character) : void getIslands() : Arrayl ist < Island > getStudentsIslands(IslandID : int) : ArrayList<Student> getTowersIslands(IslandID : int) : ArrayList<Tower> Cloud Plance getMotherNatureIslandID(): int setMotherNatureIslandID(IslandID : int) : void Students : ArravList<Student> - Entrance : ArrayList < Student > Choosen : boolean + mergelslands(IslandID1 : int, IslandID2 : int) : void <<interface>> TableProfessor : ArrayList < Professor > + getNumOfIslands(): int CloudID : int InterfaceToController Hall : Student[][] + getStudentsClound() : ArrayList<Student> + startTurn() · void Towers : ArrayList<Tower> <<enum>> setStudentsCloud(Students : ArravList<Student>) : void Island + endTurn() · void + getEntrance() : ArrayList<Student> GameState + isChoosen(): boolean + moveStudentToEntrance(PlayerID : int, Student : Student) : void + getProfessors() : ArrayList<Professor> <<enum>> Students : ArrayList < Student> + setChoosen(Choosen : boolean) : void WAITINGFORPLAYERS + moveStudentToIsland(PlayerID : int, Island : Island, Student : Student) : void getNumOfTowers(): int Tower : ArrayList<Tower> Tower + getCloudID(): int + useAssistant(PlayerID : int, Assistant : Assistant) : void PLAYING + addProfessor(Professor : Professor) : void - MotherNature : boolean FNDFD + moveMotherNature(Island : Island) : void WHITE + removeProfessor(Professor : Professor) : void Stop: boolean + useCharacter(PlayerID : int, Character : Character) : void BI ACK + removeTower(): void IslandID : int Character GREY + addTower(Tower : Tower) : void getStudents(): ArrayList<Student> Cost : int + addStudentEntrance(Student : Student) : void + getTowers() : ArrayList<Tower> Used · hoolean + removeStudentEntrance(Student : Student) : void + setTower(Tower : Tower) : void Effect : Effect + isMotherNature() : boolean <<enum>> + getCost() : int + setMotherNature(MotherNature : boolean) : void Assistant getUsed() : boolean + addStudentIsland(Student : Student) : void ION(1.1) <<enum>> - setUsed(Used : boolean) : void + isStop(): boolean OSTRICH(2.1) Professor + getEffect() : Effect + setStop(Stop : boolean) : void CAT(3.2) RED DRAGON - getIslandID() : int EAGLE(4.2) PINK FAIRY + setIslandID(IslandID : int) : void FOX(5.3) YELLOW ELF + changeTowers(Tower : Tower) : void SNAKE(6,3) BLUE UNICORN <<enum>> + addTower(Tower : Tower) : void OCTOPUS(7,4) GREEN FROG Student DOG(8,4) YELLOW ELEPHANT(9,5) BI UF <<abstract>> TURTLE(10.5) GREEN Effect + getValue() : int RED + getMovement() : in: PINK + effect(Game : Game, PlayerID : int) : void Effect1 Effect3 Effect5 Effect7 Effect9 Effect11 - Students : ArrayList < Student> NumOfStops : int - Students : ArrayList < Student> - Students : ArrayList < Student> + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void effect(Game : Game, PlayerID : int) : void Effect2 Effect4 Effect6 Effect8 Effect10 Effect12 effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void + effect(Game : Game, PlayerID : int) : void