

## Controller GameController PlanningPhaseController ActionPhaseController + setup(numberOfPlayers): void + playAssistant(int): void + moveMother(steps: int): void + fillCloud(): void + goNextPlayer(): void + moveStudentToIsland(student: Student, island: Island): void + setPlayerOrder(): void + moveStudentToDiningRoom(student: Student): void + endGame(): void + chooseCloud(cloud: Cloud): void IslandController DiningRoomController -ioinableIslands: List<Island> + getJoinableIslands(): joinableIsands + checkProfessor(): List<HasProfessor> + join(joinableIslands): Island + getOwner(Island): Player getOwner() does + conquer(Island, Player): void all the influence calculations and returns the (new) owner of the

## Process:

- 1. GameController.setup()
- a. this fills out the information needed.
- 2. PlanningPhaseController.fillCloud()
- 3. PlanningPhaseController.playAssistant()
- a. for every player in right order
- b. Can end the game
- 4. GameController.setPlayerOrder()
- 5. ActionPhaseController
- a. moveStudentToIsland()
- b. moveStudentToDiningRoom()
- DiningRoomController.checkProfessor()
- ActionPhaseController.moveMother()
- 7. IslandController checks if the island is
- conquered or controlled.
- a. getOwner()
- b. conquer()
- i. Can end the game
- c. getJoinableIslands()
- d. if joinableIsands > 1
- i. join()
- 1. update Game.islandOrder
- 2. Can end the game
- 8. ActionPhaseController.chooseCloud()
- 9. Game.goNextPlayer()
- a. goes to next round if last player's turn

## MEMO

## for list of students and professors association <<enum>> ColorAnimal

- YellowGnome

island

- BlueUnicorn
- GreenFrog
- RedDragon
- PinkFairy

For Assistant: MotherNatureMoves = floor((turnOrder+1)/2)