New UML Diagram(expert)

<<java class>> ConcreteCharacterCreator
+ createCharacter (in name :Name) :Character

<<interface>> CharacterCreator
createCharacter (in name :Name) : Character

<<enumeration>> Color
Black
White
Grey

<<enumeration>> Creature
Yellow Gnoms
Red Dragons
Blue Unicorns
Green Frogs
Pink Faries

<<enumeration>> Name
Herald ; 3
Knight ; 2
Centaur ; 3
Farmer ; 2
Fungaro ; 3
Joker ; 1
Thief ; 3
Minstrel ; 1
Monk ; 1
Herbalist ; 2
Magicpostman ; 1
Princess ; 2

<<java class>> GameModel
- players :Player[]
- table :Table
- numberofplayers :int
- characters :Character[][]
- played_character : Character
- professorchecker : ProfessorChecker
- influencecalculator : InfluenceCalculator
+ GameModel (in advanced_rules :boolean, in players :String[], in numberofplayer :int) :void
+ fillClouds () :void
+ playAssistant () :void
+ establishRoundOrder () : void
+ moveStudents (in source : StudentContainer, in destination : StudentContainer, in creatures :Creature[]) :void
+ moveMotherNature (in jumps :int) :boolean
+ checkEndGame () :boolean
+ findWinner () :Player
+ checkTower () :void
+ checkNeighbourIsland () :void
+ modifyCostOfCharacter (in character :Character) :void

<<java class>> Player
- username :String
- my_schoolboard :SchoolBoard
- my_color :Color
- last_played_card :Assistant
- my_coins :int
- assistant_deck : Assistant[]
- professors :Professor[]
- wizard :String
- towers :int
- entrance :Entrance
- dining_room : Dining_Room
+ Player (in username : String, in coloroftower : Color, in wizard :String, in numberoftower :int, in entrance :Entrance, in dining_room : Dining_Room) :void
+ addCoin () :void
+ removeCoin (in charactercost :int) :void
+ setAssistantCard (in assistant :Assistant) : void
+ modifyTower (in numberoftower :int)

<<java class>> Table
- islands :Island[]
- clouds :Cloud[]
- mother_nature : Mother_Nature
- coin_reserve :int
+ islandFusion (in islands :Island[]) : Island
+ Table (in islands : Island[], in mother_nature : Mother_Nature, in clouds :Cloud[], in characters : Character[])

<<java class>> Professor
- creature :Creature
+ professor (in creature :Creature)

<> Character
- model :GameModel
- updated_cost :int
- name :Name
+ mofidyCost () : void
+ Character (in name :Name)
+ getCost () :int
+ effect () :void

<<java class>> Herbalist
- deactivator :int
+ effect () :void

<<java class>> Postman
+ effect () :void

<<java class>> Herald
+ effect () :void

<<java class>> Thief
+ effect () :void

<<java class>> Assistant
- value :Values
+ Assistant () :void

<<enumeration>> Values
Cheetah1;1
Ostrich2;1
Cat3;2
Eagle4;2
Fox5;3
Lizard6;3
Octopus7;4
Dog8;4
Elefant9;5
Turtle10;5

character's name;value ; mother nature movement

<<java class>> Mother_Nature
- curr_island :int
+ getCurrIsland () : int
+ Mother_Nature (in position :int)
+ setCurrIsland (in currisland :int)

<> MovementCard
- students : Student[]
- max_capacity :int
+ addStudents (in student :Student) : void
+ removeStudent (in creature :Creature) : Student
+ effect () :void

<<java class>> Mover
+ effect () :void

monk and princess

<<java class>> Trader
+ effect () :void

joker and ministrel

<<java class>> Entrance
+ Entrance (in capacity :int) :void

<<java class>> Dining_Room
- diners : map<Creature, Students[]>
+ getNumberofCreature (in creature : Creature) :int
+ Dining_Room () : void

<<java class>> Island
- towers :int
- coloroftowers :Color
- noentry :int
+ changeTower (in color : Color)
+ Island (in students : Student[], in numberoftower :int, in coloroftower :Color) :
+ addNoEntry () :void
+ removeNoEntry () :void

<<java class>> Cloud
- extracted_students :Student[]
+ extractStudent () : Student[]
+ Cloud (in max_students :int) :void

<<java class>> StudentContainer
- students : Student[]
- capacity :int
+ addStudents (in newStudents : Student[]) :void
+ removeStudents (in creature : Creature) :Student

<<java class>> Student
- creature :Creature
+ Student (in creature :Creature) : Student

<<java class>> StudentBucket
- generated_students :Map<Creature, Integer>
- max_students :int

2..4
0..5
0..10
3
3..12
2;3