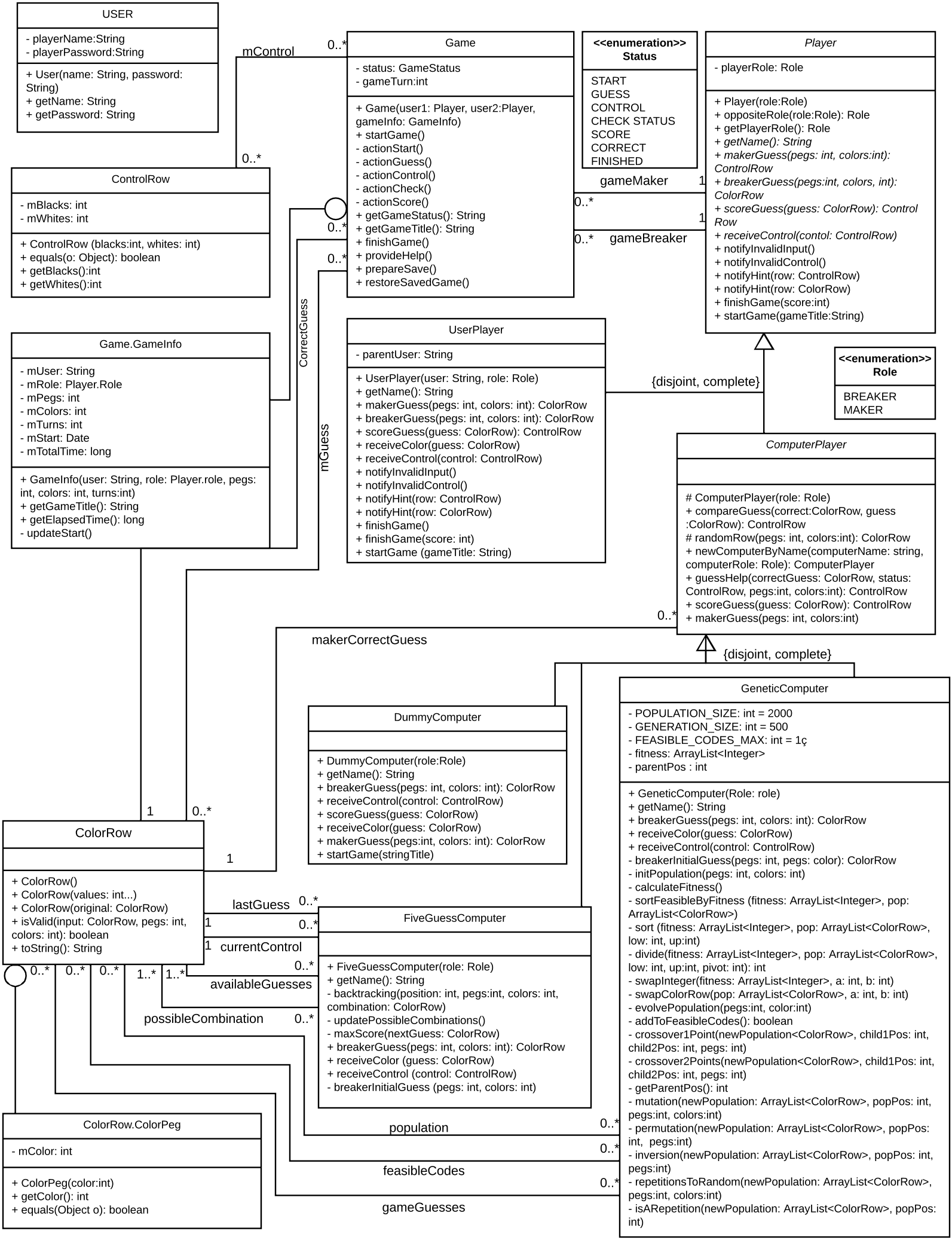
USER
- playerName:String
- playerPassword:String
+ User(name: String, password: String)
+ getName: String
+ getPassword: String
mControl
0..*
Game
- status: GameStatus
- gameTurn:int
+ Game(user1: Player, user2:Player, gameInfo: GameInfo)
+ startGame()
- actionStart()
- actionGuess()
- actionControl()
- actionCheck()
- actionScore()
+ getGameStatus(): String
+ getGameTitle(): String
+ finishGame()
+ provideHelp()
+ prepareSave()
+ restoreSavedGame()
<<enumeration>>
Status
START
GUESS
CONTROL
CHECK STATUS
SCORE
CORRECT
FINISHED
Player
- playerRole: Role
+ Player(role:Role)
+ oppositeRole(role:Role): Role
+ getPlayerRole(): Role
+ getName(): String
+ makerGuess(pegs: int, colors:int): ControlRow
+ breakerGuess(pegs:int, colors, int): ColorRow
+ scoreGuess(guess: ColorRow): Control Row
+ receiveControl(contol: ControlRow)
+ notifyInvalidInput()
+ notifyInvalidControl()
+ notifyHint(row: ControlRow)
+ notifyHint(row: ColorRow)
+ finishGame(score:int)
+ startGame(gameTitle:String)
gameMaker
1
0..*
1
0..*
gameBreaker
ControlRow
- mBlacks: int
- mWhites: int
+ ControlRow (blacks:int, whites: int)
+ equals(o: Object): boolean
+ getBlacks():int
+ getWhites():int
CorrectGuess
mGuess
Game.GameInfo
- mUser: String
- mRole: Player.Role
- mPegs: int
- mColors: int
- mTurns: int
- mStart: Date
- mTotalTime: long
+ GameInfo(user: String, role: Player.role, pegs: int, colors: int, turns:int)
+ getGameTitle(): String
+ getElapsedTime(): long
- updateStart()
UserPlayer
- parentUser: String
+ UserPlayer(user: String, role: Role)
+ getName(): String
+ makerGuess(pegs: int, colors: int): ColorRow
+ breakerGuess(pegs: int, colors: int): ColorRow
+ scoreGuess(guess: ColorRow): ControlRow
+ receiveColor(guess: ColorRow)
+ receiveControl(control: ControlRow)
+ notifyInvalidInput()
+ notifyInvalidControl()
+ notifyHint(row: ControlRow)
+ notifyHint(row: ColorRow)
+ finishGame()
+ finishGame(score: int)
+ startGame (gameTitle: String)
{disjoint, complete}
<<enumeration>>
Role
BREAKER
MAKER
ComputerPlayer
ComputerPlayer(role: Role)
+ compareGuess(correct:ColorRow, guess :ColorRow): ControlRow
# randomRow(pegs: int, colors:int): ColorRow
+ newComputerByName(computerName: string, computerRole: Role): ComputerPlayer
+ guessHelp(correctGuess: ColorRow, status: ControlRow, pegs:int, colors:int): ControlRow
+ scoreGuess(guess: ColorRow): ControlRow
+ makerGuess(pegs: int, colors:int)
0..*
makerCorrectGuess
{disjoint, complete}
GeneticComputer
- POPULATION_SIZE: int = 2000
- GENERATION_SIZE: int = 500
- FEASIBLE_CODES_MAX: int = 1ç
- fitness: ArrayList<Integer>
- parentPos : int
+ GeneticComputer(Role: role)
+ getName(): String
+ breakerGuess(pegs: int, colors: int): ColorRow
+ receiveColor(guess: ColorRow)
+ receiveControl(control: ControlRow)
- breakerInitialGuess(pegs: int, pegs: color): ColorRow
- initPopulation(pegs: int, colors: int)
- calculateFitness()
- sortFeasibleByFitness (fitness: ArrayList<Integer>, pop: ArrayList<ColorRow>)
- sort (fitness: ArrayList<Integer>, pop: ArrayList<ColorRow>, low: int, up:int)
- divide(fitness: ArrayList<Integer>, pop: ArrayList<ColorRow>, low: int, up:int, pivot: int): int
- swapInteger(fitness: ArrayList<Integer>, a: int, b: int)
- swapColorRow(pop: ArrayList<ColorRow>, a: int, b: int)
- evolvePopulation(pegs:int, color:int)
- addToFeasibleCodes(): boolean
- crossover1Point(newPopulation<ColorRow>, child1Pos: int, child2Pos: int, pegs: int)
- crossover2Points(newPopulation<ColorRow>, child1Pos: int, child2Pos: int, pegs: int)
- getParentPos(): int
- mutation(newPopulation: ArrayList<ColorRow>, popPos: int, pegs:int, colors:int)
- permutation(newPopulation: ArrayList<ColorRow>, popPos: int, pegs:int)
- inversion(newPopulation: ArrayList<ColorRow>, popPos: int, pegs:int)
- repetitionsToRandom(newPopulation: ArrayList<ColorRow>, pegs:int, colors:int)
- isARepetition(newPopulation: ArrayList<ColorRow>, popPos: int)
DummyComputer
+ DummyComputer(role:Role)
+ getName(): String
+ breakerGuess(pegs: int, colors: int): ColorRow
+ receiveControl(control: ControlRow)
+ scoreGuess(guess: ColorRow)
+ receiveColor(guess: ColorRow)
+ makerGuess(pegs:int, colors: int): ColorRow
+ startGame(stringTitle)
1
0..*
1
ColorRow
+ ColorRow()
+ ColorRow(values: int...)
+ ColorRow(original: ColorRow)
+ isValid(input: ColorRow, pegs: int, colors: int): boolean
+ toString(): String
lastGuess
0..*
1
0..*
1
currentControl
0..*
availableGuesses
0..*
0..*
0..*
0..*
1..*
1..*
possibleCombination
0..*
FiveGuessComputer
+ FiveGuessComputer(role: Role)
+ getName(): String
- backtracking(position: int, pegs:int, colors: int, combination: ColorRow)
- updatePossibleCombinations()
- maxScore(nextGuess: ColorRow)
+ breakerGuess(pegs: int, colors: int): ColorRow
+ receiveColor (guess: ColorRow)
+ receiveControl (control: ControlRow)
- breakerInitialGuess (pegs: int, colors: int)
ColorRow.ColorPeg
- mColor: int
+ ColorPeg(color:int)
+ getColor(): int
+ equals(Object o): boolean
population
0..*
0..*
feasibleCodes
0..*
gameGuesses