UML Class Diagrams

Joseph Maples CS101
Design for a number guessing game

+	public
-	private
	package
#	protected

legend

+	PlayGame
+	main(args:String [])

+	GamePlayer
-	guessingGame:GuessingGame
-	player:Scanner
+	getGuessingGame()
-	setGuessingGame(guessingGame:GuessingGame)
+	getPlayer()
-	setPlayer(player:Scanner)
+	GamePlayer(player:Scanner)
+	play()

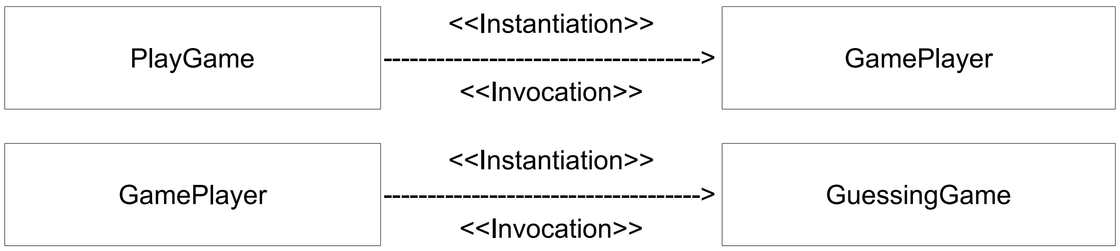
+	GuessingGame
+	EASY_GAME:boolean
+	DIFFICULT_GAME:boolean
+	DEFAULT_MAXIMUM_RANGE:int
+	GAME_WON:int
+	GAME_LOST:int
+	GAME_IN_PROGRESS:int
-	gameState:int
-	numberToGuess:int
-	numberOfGuesses:int

```
-   largestPossibleNumber:int
-   maximumNumberOfGuesses:int
-   currentMinimumRange:int
-   currentMaximumRange:int
-   guessToolow:boolean
-   easyGame:boolean
```

```
+   GuessingGame(largestPossibleNumber:int, difficulty:boolean
+   GuessingGame(largestPossibleNumber:int)
+   GuessingGame(difficulty:boolean)
+   GuessingGame()
+   getGameState()
-   setGameState(gameState:int)
+   getNumberToGuess()
-   setNumberToGuess(numberToGuess:int)
+   getNumberOfGuesses()
-   setNumberOfGuesses(numberOfGuesses:int)
+   getLargestPossibleNumber()
-   setLargestPossibleNumber(largestPossibleNumber:int)
+   getMaximumNumberOfGuesses()
-   setMaximumNumberOfGuesses(maximumNumberOfGuesses:int)
+   getCurrentMinimumRange()
-   setCurrentMinimumRange(currentMinimumRange:int)
+   getCurrentMaximumRange()
-   setCurrentMaximumRange(currentMaximumRange:int)
+   getGuessTooLow()
-   setGuessTooLow(tooLow:boolean)
+   getGameEasy()
-   setGameEasy(gameEasy:boolean)
-   computeMaxNumberOfGuesses()
+   makeGuess(guess:int)
+   hint()
+   quit()
+   toString()
```

UML Class Interaction Diagram

Joseph Maples CS101
Design for a number guessing game



Joseph Maples CS101
Design for a number guessing game

Data Table for Class GuessingGame

Variable or Constant	Type	Purpose
EASY_GAME	boolean	Default to easy game
DIFFICULT_GAME	boolean	Difficult game
DEFAULT_MAXIMUM_RANGE	int	Default to 10 as largest number
GAME_WON	int	The game state value that is winning
GAME_LOST	int	The game state value that is losing
GAME_IN_PROGRESS	int	Value of the in progress game state
gameState	int	The current state of the game
numberToGuess	int	The number to guess
numberOfGuesses	int	The number of times the user has guessed
largestPossibleNumber	int	The largest number to guess
maximumNumberOfGuesses	int	Max number of times the user can guess
currentMinimumRange	int	Lowest number in guessing range
currentMaximumRange	int	Current largest number in guessing range
guessTooLow	boolean	Guess is lower than numberToGuess
easyGame	boolean	Game difficulty level

Data Table for GuessingGame(largestPossibleNumber:int, difficulty:boolean)

Variable or Constant	Type	Purpose
largestPossibleNumber	int	The largest number to guess
difficulty	boolean	Easy or difficult
rand	Random	Generate random number

Data Table for GuessingGame(largestPossibleNumber:int)

Variable or Constant	Type	Purpose
largestPossibleNumber	int	The largest number to guess
EASY_GAME	boolean	Default to easy game

Data Table for GuessingGame(difficulty:boolean)

Variable or Constant	Type	Purpose
DEFAULT_MAXIMUM_RANGE	int	Default to 10 as largest number
difficulty	boolean	Easy or difficult

Data Table for GuessingGame()

Variable or Constant	Type	Purpose
DEFAULT_MAXIMUM_RANGE	int	Default to 10 as largest number
EASY_GAME	boolean	Default to easy game

Data Table for setGameState(gameState:int)

Variable or Constant	Type	Purpose
gameState	int	Current game state
gameState	int	Parameter, new state

Data Table for setNumberToGuess(numberToGuess:int)

Variable or Constant	Type	Purpose
numberToGuess	int	The number to guess
numberToGuess	int	Parameter, new number to guess

Data Table for setNumberOfGuesses(numberOfGuesses:int)

Variable or Constant	Type	Purpose
numberOfGuesses	int	The number of times the user has guessed
numberOfGuesses	int	Parameters, the new number of times the user has guessed

Data Table for setLargestPossibleNumber(largestPossibleNumber:int)

Variable or Constant	Type	Purpose
largestPossibleNumber	int	The largest number to guess
largestPossibleNumber	int	Parameter, new largest number

Data Table for setMaximumNumberOfGuesses(maximumNumberOfGuesses:int)

Variable or Constant	Type	Purpose
maximumNumberOfGuesses	int	Max number of times the user can guess
maximumNumberOfGuesses	int	Parameter, new max number of guesses

Data Table for setCurrentMinimumRange(currentMinimumRange:int)

Variable or Constant	Type	Purpose
currentMinimumRange	int	Lowest number in guessing range
currentMinimumRange	int	Parameter, new lowest number in range

Data Table for setCurrentMaximumRange(currentMaximumRange:int)

Variable or Constant	Type	Purpose
currentMaximumRange	int	Current largest number in guessing range
currentMaximumRange	int	Parameter, new largest number in range

Data Table for setGuessTooLow(guessTooLow:boolean)

Variable or Constant	Type	Purpose
guessTooLow	boolean	Guess is lower than numberToGuess
guessTooLow	boolean	Parameter, new guess is too low

Data Table for setEasyGame(easyGame:boolean)

Variable or Constant	Type	Purpose
easyGame	boolean	Current game difficulty level
easyGame	boolean	Parameter, new difficulty level

Data Table for computeMaxNumberOfguesses()

Variable or Constant	Type	Purpose
easyGame	boolean	Game difficulty level
iterations	int	Number of times that the computation loop iterates
maxGuessFactor	int	Number that will be modified to convert largestPossibleNumber to the max number of guesses

largestPossibleNumber	int	The largest number to guess

Data Table for makeGuess(guess:int)

Variable or Constant	Type	Purpose
guess	int	The number the user guessed
numberOfGuesses	int	Number of times the user has guessed
numberToGuess	int	The number the user needs to guess
GAME_WON	int	The game state value that is winning
gameState	int	The current state of the game
GAME_LOST	int	The game state value that is losing

Data Table for hint()

Variable or Constant	Type	Purpose
hint	String	The users hint
currentMinimumRange	int	The lowest number in guessing range
currentMaximumRange	int	The highest number in guessing range

Data Table for quit()

Variable or Constant	Type	Purpose
GAME_LOST	int	Value of the losing game state

Data Table for toString()

Variable or Constant	Type	Purpose
guessingGame	String	The object as a string
gameState	int	The current game state
GAME_IN_PROGRESS	int	Value of the in progress game state
maximumNumberOfGuesses	int	number of guesses the user is allowed
numberOfGuesses	int	Number of times the user has guessed
GAME_WON	int	Value of the winning game state
numberToGuess	int	The number the user needs to guess

Joseph Maples CS101
Design for a number guessing game

Data Table for Class GamePlayer

Variable or Constant	Type	Purpose
guessingGame	GuessingGame	Current guessingGame object
player	Scanner	Get user input

Data table for setGuessingGame(guessingGame)

Variable or Constant	Type	Purpose
guessingGame	GuessingGame	Current guessingGame object
guessingGame	GuessingGame	Parameter, new object

Data Table for setPlayer(player:Scanner)

Variable or Constant	Type	Purpose
player	Scanner	Current Scanner object
player	Scanner	Parameter, new object

Data Table for GamePlayer(ScannerPlayer)

Variable or Constant	Type	Purpose
player	Scanner	Get user input
option	int	User selected menu option
difficulty	boolean	User selected game difficulty
upperBound	int	User selected highest number
guessingGame	GuessingGame	Object to play game

Data Table for play()

Variable or Constant	Type	Purpose
option	int	User selected menu option

Data Table for PlayGame

Joseph Maples CS101
Design for a number guessing game

Data Table for main(args)

Variable or Constant	Type	Purpose
args	String []	parameter, unused
scan	Scanner	Read user input
playAgain	boolean	Whether the user wants to play again
games	int	Number of games the user has played
player	GamePlayer	GamePlayer object for playing the game

Algorithms for GuessingGame

Joseph Maples CS101
Design for a number guessing game

GuessingGame Class Algorithms

Algorithms for GuessingGame Constructors

```
GuessingGame(largestPossibleNumber, difficulty)
    setLargestPossibleNumber(largestPossibleNumber)
    setEasyGame(difficulty)
    Random rand equals new Random()
    setCurrentMaximumRange(largestPossibleNumber)
    setCurrentMinimumRange(0)
    setNumberToGuess(rand.nextInt(largestPossibleNumber))
    setMaximumNumberOfGuesses(computeMaxNumberOfGuesses())

GuessingGame(largestPossibleNumber)
    this(largestPossibleNumber, EASY_GAME)

GuessingGame(difficulty)
    this(DEFAULT_MAXIMUM_RANGE, difficulty)

GuessingGame()
    this(DEFAULT_MAXIMUM_RANGE, EASY_GAME)
```

Algorithms for instance variable mutators

```
setGameState(gameState)
    this.gameState equals gameState

setNumberToGuess(numberToGuess)
    this.numberToGuess equals numberToGuess

setNumberOfGuesses(numberOfGuesses)
    this.numberOfGuesses equals numberOfGuesses

setLargestPossibleNumber(number)
    this.largestPossibleNumber equals
                  Math.max(number, DEFAULT_MAXIMUM_RANGE)

setMaximumNumberOfGuesses(maximumNumberOfGuesses)
    this.maximumNumberOfGuesses equals maximumNumberOfGuesses

setCurrentMinimumRange(min)
    if (min is greater than currentMinimumRange)
        currentMinimumRange equals min

setCurrentMaximumRange(max)
    if (max is less than currentMaximumRange)
        currentMaximumRange equals max
    else
```

```
        currentMaximumRange equals largestPossibleNumber

setGuessTooLow(tooLow)
    this.guessTooLow equals guessTooLow
```

Algorithm for computeMaxNumberOfGuesses()

```
computeMaxNumberOfGuesses()
    if (easyGame)
        return Math.ceil(largestPossibleNumber divided by 2.0)
    else
        iterations equals 0
        maxGuessFactor equals largestPossibleNumber
        while (maxGuessFactor is greater than 1)
            maxGuessFactor equals maxGuessFactor divided by 2
            iterations plus 1
        iterations plus 1
        return iterations
```

Algorithm for makeGuess(guess)

```
makeGuess(guess)
    numberOfGuesses plus 1
    if (guess is equal to numberToGuess)
        setGameState(GAME_WON)
    else if (guess is less than numberToGuess)
        setGuessTooLow(true)
        setCurrentMinimumRange(guess)
    else if (guess is greater than numberToGuess)
        setGuessTooLow(false)
        setCurrentMaximumRange(guess)
    if (numberOfGuesses is equal to 0 AND NOT
                    (gameState is equal to GAME_WON))
        setGameState(GAME_LOST)
```

Algorithm for hint()

```
hint()
    String hint equals "Guess a number between " + currentMinimumRange
      + " and " + currentMaximumRange
    return hint
```

Algorithm for quit()

```
quit()
    setGameState(GAME_LOST)
```

Algorithm for toString()

```
toString()
    String guessingGame
    if (gameState is equal to GAME_IN_PROGRESS)
        guessingGame equals "You have "
        + (maximumNumberOfGuesses - numberOfGuesses) + " guesses remaining."
```

```
else if (gameState is equal to GAME_WON)
   guessingGame equals "It took " + numberOfGuesses + " guesses to get "
     + numberToGuess + ". Congratulations!"
else
   guessingGame equals "You used all " + numberOfGuesses
     + " guesses. The correct number was " + numberToGuess + "."
return guessingGame
```

Joseph Maples CS101
Design for a number guessing game

GamePlayer Class Algorithms

Algorithms for GamePlayer Constructor

```
GamePlayer(Scanner player)
    setPlayer(player)
    option equals 0
    difficulty equals GuessingGame.EASY_GAME
    upperBound equals GuessingGame.DEFAULT_MAXIMUM_RANGE
    while (option does not equal 3)
        print "Select a menu option:\n1. choose difficulty level”
            + “\n2. pick upper bound for guess\n3. play game"
        option equals player.nextInt()
        switch (option) :
            case 1:
                print "Choose game difficulty: easy or difficult"
                if (player.next().toLowerCase().contains("d"))
                    difficulty equals GuessingGame.DIFFICULT_GAME
                break
            case 2:
                print "Set the highest number you want to guess:"
                upperBound equals player.nextInt()
                break
            case 3:
                print "Starting game"
                break
            default:
                print "Invalid option"
                break
    guessingGame equals new GuessingGame(upperBound, difficulty)
```

Algorithms for accessor and mutator methods

```
getGuessingGame()
    return guessingGame

setGuessingGame(GuessingGame guessingGame)
    this.guessingGame equals guessingGame

getPlayer()
    return player

setPlayer(Scanner player)
    this.player equals player
```

Algorithm for play()

```
play()
    option equals 0
```

```
while (option does not equal 4)
    print "Select a menu option:" + "\n1. make a guess" + "\n2. get a hint"
         + "\n3. print statistics" + "\n4. quit this game"
    option equals player.nextInt()
    switch (option) :
        case 1:
            print "Make a guess:"
            guessingGame.makeGuess(player.nextInt())
            if (GuessingGame.GAME_IN_PROGRESS does not equal
                        guessingGame.getGameState())
                print guessingGame.toString()
            break
        case 2:
            print guessingGame.hint()
            break
        case 3:
            print guessingGame.toString()
            break
        case 4:
            guessingGame.makeGuess(player.nextInt())
            if (GuessingGame.GAME_IN_PROGRESS does not equal
                        guessingGame.getGameState())
                guessingGame.quit()
            print guessingGame.toString()
            break
        default:
            print "Invalid option"
            break
```

Algorithms for PlayGame

Joseph Maples CS101
Design for a number guessing game

PlayGame Class Algorithms

Algorithm for main(args)

```
main(String[] args)
    instantiate Scanner scan
    playAgain equals false
    games equals 0
    do
        GamePlayer player equals new GamePlayer(scan)
        player.play()
        games plus 1
        print "Would you like to play again? [y/N]"
        playAgain equals scan.next().toLowerCase().contains("y")
    while (playAgain)
    if (games equals 1)
        print "Thanks for playing!"
    else
        print "Thanks for playing " + games + " times!"
```