UCLA

CS 31 : Introduction To Computer Science I

Howard A. Stahl

Project 6

- The Goal: A Working ShutTheBox Game
- Background: Please Play A Few Games With This Free Game
  - http://www.playonlinedicegames.com/shutthebox
- Truth In Advertising:
  - We'll Only Be Dealing With The Following Concepts: Die, Player, Board, ShutTheBox
  - No Need To Worry Sound, Graphics

Project 6

- Unlike Earlier Assignments, I Am Supplying You With A Partial "Skeleton" Of The Code Solution
- It Will Run Right Out Of The Box
  - Some Important Pieces Are Stubbed Out…
  - These Are The Parts You Need To Complete
- Hint 1: Acquire The Skeleton!
- Hint 2: Build And The Run The Skeleton!
  - Look At What Is Working And What Is Not

Project 6

- The Work Product: The Implementation Of The Public API Of The Classes Described Here And In The Assignment.
- You Are Free To Do It However You Like, But You Must Provide The Public API I Am Looking For..
  - You Can Add Classes, Methods, Members As You Feel Appropriate
  - But I Honestly Don't Think You'll Need To…
- In What Follows, It Is The **Bolded** Portions That You Need To Complete

Introducing The Die Class

- Using The Die Class, We'll Have Random Play, Like In The Real World…
- We'll Be Using Six-Sided Dies
  - mSides=6!
- `roll()` tosses the Die
- `getValue()` retrieves what was rolled

```
Die
---------------------
- mSides : int
- mValue : int
---------------------
+ Die( sides : int )

+ roll( ) : void
+ getValue( ) : int
```

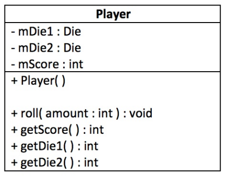
Introducing The Die Class

- Using The Die Class, We'll Have Random Play, Like In The Real World…
- We'll Be Using Six-Sided Dies
  - mSides=6!
- `roll()` tosses the Die
- `getValue()` retrieves what was rolled

```
Die
---------------------
- mSides : int
- mValue : int
---------------------
+ Die( sides : int )

+ roll( ) : void
+ getValue( ) : int
```

The Player Class

- Manages Two Dies



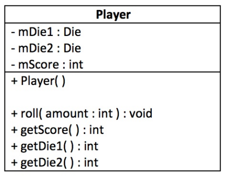
Player
- mDie1 : Die
- mDie2 : Die
- mScore : int

+ Player( )

+ roll(amount : int) : void
+ getScore( ) : int
+ getDie1( ) : int
+ getDie2( ) : int

The Player Class

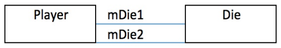
- Manages Two Dies



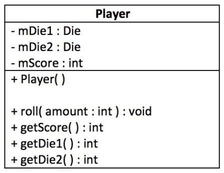
Player
- mDie1 : Die
- mDie2 : Die
- mScore : int

+ Player( )

+ roll(amount : int) : void
+ getScore( ) : int
+ getDie1( ) : int
+ getDie2( ) : int



The Player Class



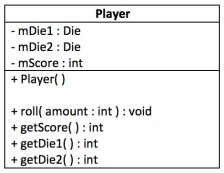
Player
- mDie1 : Die
- mDie2 : Die
- mScore : int

+ Player( )

+ roll(amount : int) : void
+ getScore( ) : int
+ getDie1( ) : int
+ getDie2( ) : int

- Manages Two Dies And A Score
- `roll( )` Tosses The Two Dies Unless… A Non-Zero Amount Is Supplied
  - When Passing A Zero Amount, The Game Proceeds Randomly
  - When Passing A Non-Zero Amount, You Can Force Certain Game Behavior AKA Cheating! But Very Useful For Testing Purposes…
- `getDie1( )` Get The Die That Was Just Rolled
  - Or A Value That Fits If Cheating Was Desired…
- `getDie2( )` Gets The Die That Was Just Rolled
  - Or A Value That Fits If Cheating Was Desired…
- `getScore( )` Returns The Total Of The Two Tossed Dies

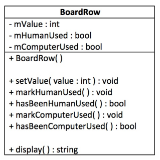
The Player Class



- Manages Two Dies And A Score
- `roll( )` Tosses The Two Dies Unless… A Non-Zero Amount Is Supplied
  - When Passing A Zero Amount, The Game Proceeds Randomly
  - When Passing A Non-Zero Amount, You Can Force Certain Game Behavior AKA Cheating! But Very Useful For Testing Purposes…
- `getDie1( )` Get The [illegible]ie Th[illegible]at [illegible]as Just Rolled
  - Or A Value That Fits If Cheat[illegible]sired…
- `getDie2( )` Gets The D[illegible]ust Rolled
  - Or A Value That Fits If Cheati[illegible]sired…
- `getScore( )` Returns The [illegible]otal Of The Two Tossed Dies

YAY! Ain't Nothing To Do Here…

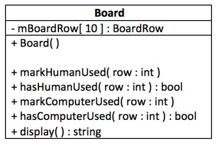
The BoardRow Class

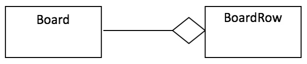

BoardRow
- mValue : int
- mHumanUsed : bool
- mComputerUsed : bool
+ BoardRow( )
+ setValue(value : int) : void
+ markHumanUsed( ) : void
+ hasBeenHumanUsed( ) : bool
+ markComputerUsed( ) : void
+ hasBeenComputerUsed( ) : bool
+ display( ) : string

- Manages One Row Of The Game Table Output
- `setValue( )` Provides The Row Number
  - Should Be A Value Between 1 And 9…
- **`markHumanUsed( )`** Is A Mutator/"Setter" Method
- **`hasBeenHumanUsed( )`** Is An Accessor/"Getter" Method
- **`markComputerUsed( )`** Is A Mutator/"Setter" Method
- **`hasBeenComputerUsed( )`** Is An Accessor/"Getter" Method
- `display( )` Stringifies This Object So It Can Be Printed In The Game Table

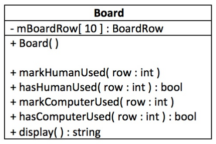
The Board Class

- Manages A Set Of BoardRow


Board
- mBoardRow[10] : BoardRow
+ Board( )
+ markHumanUsed(row : int)
+ hasHumanUsed(row : int) : bool
+ markComputerUsed(row : int)
+ hasComputerUsed(row : int) : bool
+ display( ) : string



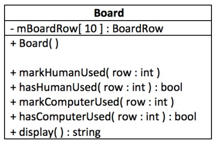
The Board Class



```
Board
- mBoardRow[ 10 ] : BoardRow
+ Board( )

+ markHumanUsed( row : int )
+ hasHumanUsed( row : int ) : bool
+ markComputerUsed( row : int )
+ hasComputerUsed( row : int ) : bool
+ display( ) : string
```

- Manages A Set Of BoardRow
  - For Simplicity Sake, We'll Be Using Array Elements At Index 1 Thru 9, Ignoring Index 0…
- **`markHumanUsed( row )`** Is A Mutator/"Setter" Method
- **`hasBeenHumanUsed( row )`** Is An Accessor/"Getter" Method
- **`markComputerUsed( row )`** Is A Mutator/"Setter" Method
- **`hasBeenComputerUsed( row )`** Is An Accessor/"Getter" Method
- `display( )` Stringifies This Object So It Can Be Printed

The Board Class



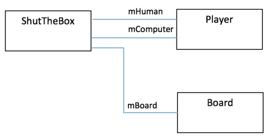
```
Board
- mBoardRow[ 10 ] : BoardRow
+ Board( )

+ markHumanUsed( row : int )
+ hasHumanUsed( row : int ) : bool
+ markComputerUsed( row : int )
+ hasComputerUsed( row : int ) : bool
+ display( ) : string
```

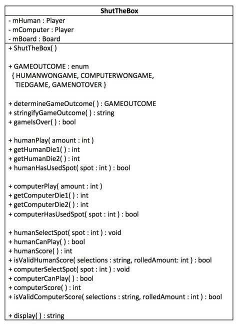
- Manages A Set Of BoardRow
  - For Simplicity Sake, We'll Be Using Array Elements At Index 1 Thru 9, Ignoring Index 0…
- **`markHumanUsed( row )`** Is A Mutator/"Setter" Method
- **`hasBeenHumanUsed( row )`** Is An Accessor/"Getter" Method
- **`markComputerUsed( row )`** Is A Mutator/"Setter" Method
- **`hasBeenComputerUsed( ro`** Method
- `display( )` Stringifies This Obj

Be Careful Not To Go Outside The Bounds of The BoardRow Array…

The ShutTheBox Class

- Manages Two Players And A Board





```
ShutTheBox
- mHuman : Player
- mComputer : Player
- mBoard : Board
+ ShutTheBox( )

+ GAMEOUTCOME : enum
  { HUMANWONGAME, COMPUTERWONGAME,
    TIEDGAME, GAMENOTOVER }

+ determineGameOutcome( ) : GAMEOUTCOME
+ stringifyGameOutcome( ) : string
+ gameIsOver( ) : bool

+ humanPlay( amount : int )
+ getHumanDie1( ) : int
+ getHumanDie2( ) : int
+ humanHasUsedSpot( spot : int ) : bool

+ computerPlay( amount : int )
+ getComputerDie1( ) : int
+ getComputerDie2( ) : int
+ computerHasUsedSpot( spot : int ) : bool

+ humanSelectSpot( spot : int ) : void
+ humanCanPlay( ) : bool
+ humanScore( ) : int
+ isValidHumanScore( selections : string, rolledAmount: int ) : bool
+ computerSelectSpot( spot : int ) : void
+ computerCanPlay( ) : bool
+ computerScore( ) : int
+ isValidComputerScore( selections : string, rolledAmount : int ) : bool

+ display( ) : string
```

The GAMEOUTCOME Enumeration

- I Am Very Partial To Enumerations….
- `GAMEOUTCOME` Lists The Possible Results Of Playing A Game:
  - `HUMANWON` – No Moves Left And Human Had The Smaller Score! Yay!
  - `COMPUTERWON` – No Moves Left And Computer Had The Smaller Score! Boo!
  - `TIEDGAME` – No Moves Left And The Players Both Had The Same Score
  - `GAMENOTOVER` – Based On The Board And The Current Rolls, One (Or Both) Of The Players Still Has A Possible Valid Move That Can Be Made

The ShutTheBox Class

- Manages Two Players And A Board
- Many Methods. I Am Only Describing The Ones You Need To Change
- **`getHumanDie1( )`** returns int
  Return The Human's First Die
- **`getHumanDie2( )`** returns int
  Return The Human's Second Die
- **`getComputerDie1( )`** returns int
  Return The Computer's First Die
- **`getComputerDie2( )`** returns int
  Return The Computer's Second Die

The ShutTheBox Class

- Manages Two Players And A Board
- **`hasHumanUsedSpot( int spot )`** returns bool
  Has The Human Already Used This Spot On The Board?
  HINT: Check The Board's BoardRow For This Spot
- **`hasComputerUsedSpot( int spot )`** returns bool
  Has The Computer Already Used This Spot On The Board?
  HINT: Check The Board's BoardRow For This Spot

The ShutTheBox Class

- Manages Two Players And A Board
- **`determineGameOutcome( )`** returns GAMEOUTCOME
  If Neither Player Can Play, Then Consider The Score Of Each Player
  To Determine The Outcome
  HINT: Call .humanCanPlay( ) And .computerCanPlay( )
  HINT: Call .humanScore( ) And .computerScore( )
- **`gameIsOver()`** returns bool
  Is The Current GameOutcome != GAMENOTOVER?
- **`humanPlay( int )`** returns int
  Allow The Human Player To Play
  If The Argument Is Non-Zero, Then Force The Human To Roll That Amount
  If The Argument Is Zero, Force A Random Roll For The Human Player
  Returns The Human Player's Score

Driver Code Says:

```
• ShutTheBox   game;
  do
  {
     do  {
        game.humanPlay();
        if  (game.humanCanPlay())
        {
           //  select   spots
           game.humanSelectSpots(   value   );
        }
        else
        {
           game.computerPlay();
           if  (game.computerCanPlay())
           {
              //  select   spots
              game.computerSelectSpots(   value   );
           }
           else
           {
              break;
           }
        }
      }  while(   true   );
  }  while(   !game.gameIsOver()   );
```

Driver Code Says:

```
• ShutTheBox   game;
  do
  {
     do  {
        game.humanPlay();
        if  (game.humanCanPlay())
        {
           //  select   spots
           game.humanSelectSpots(   value   );
        }
        else
        {
           game.computerPlay();
           if  (game.computerCanPlay())
           {
              //  select   spots
              game.computerSelectSpots(   value
           }
           else
           {
              break;
           }
        }
      }  while(   true   );
  }  while(   !game.gameIsOver()   );
```

For Testing Purposes, Wouldn't It Be Great If We Could CHEAT…

Cheating Driver Code Says:

```
• ShutTheBox  game;
  game.humanPlay(  7 );
  game.humanSelectSpot(  1 );
  game.humanSelectSpot(  2 );
  game.humanSelectSpot(  4 );
  game.humanPlay(  8 );
  game.humanSelectSpot(  8 );
  game.humanPlay(  6 );
  game.humanSelectSpot(  6 );
  game.humanPlay(  6 ); // cannot  be played...
  game.humanScore()  == 3 + 5 + 7 + 9;
```

Cheating Driver Code Says:

```
• ShutTheBox  game;
  game.computerPlay(  6 );
  game.computerSelectSpot(   1 );
  game.computerSelectSpot(   5 );
  game.computerPlay(  8 );
  game.computerSelectSpot(   8 );
  game.computerPlay(  7 );
  game.computerSelectSpot(   3 );
  game.computerSelectSpot(   4 );
  game.computerPlay(  5 ); // cannot  be played...
  game.computerScore()  == 2  + 6  + 7  + 9;
```