James Sandoval
Lab 3 – Completion of lab and Reflection

The initial design process proved to be quite a bit different from the final design, primarily due to my own misunderstanding of inheritance and the "is-a" and "has-a" relationships of classes and objects. I've color-coded the items in the pseudo code to show which elements of the design I did not need (red), and which items I added throughout the design. The biggest issue I faced was in understanding how the inherited class functions might be accessed from the gameplay class. After developing a stronger understanding of how the constructors need to be initialized in an inherited class I found a solution. Additionally, I need to be more thorough in my design process. For example, I should have thought about things like creating member variables for each players score beforehand. Overall, this was a great learning experience, both in developing a better design process, as well as a thorough working knowledge inheritance.

Class Die
- Private:
  - Int numberOfSides
- Public:
  - Getter/Setter functions for numberOfSides
  - dieRoll Function

Class LoadedDie : Public Die
- Private:
  - Int numberOfSides
- Public:
  - Getter/Setter functions for numberOfSides
  - LoadedDieRollFunction

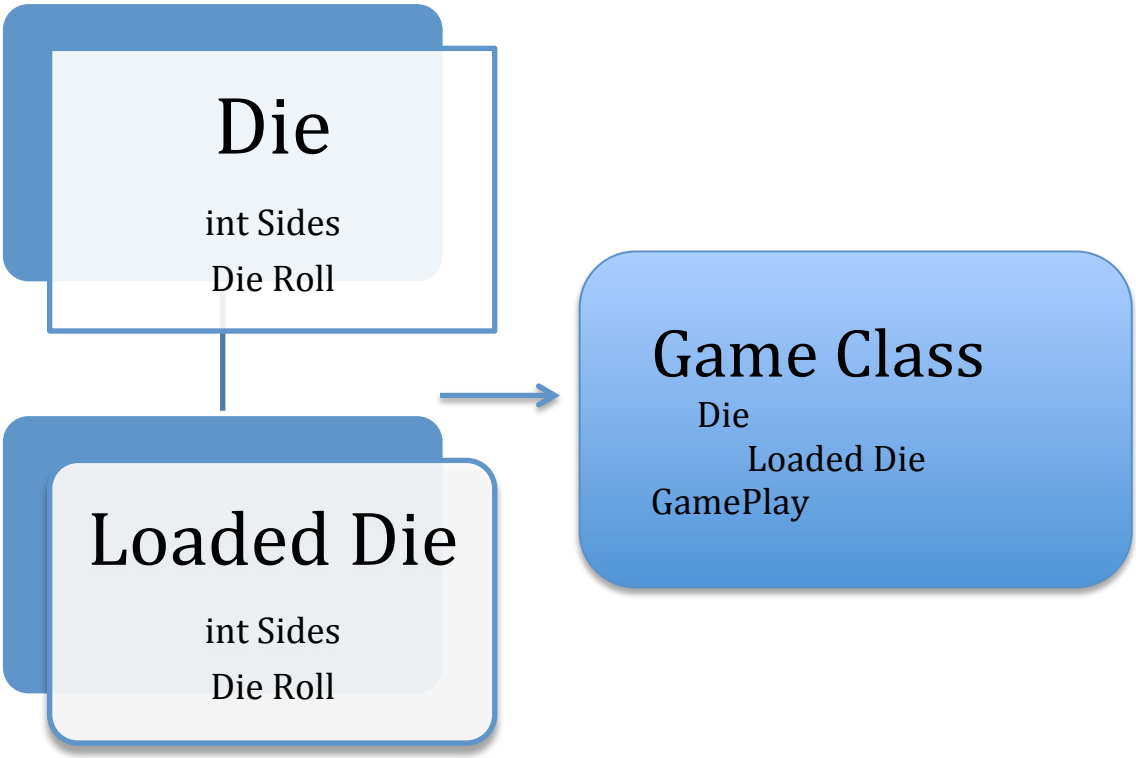
Class Game(has-a-die/loaded die)
- Private:
  - Rounds
  - Die1
  - Die2
  - Bool – loaded die or regular
  - Playerscore1
  - Playerscore2
  - Players
  - scoreCounter
- Public:
  - Set dieValue
  - Setloaded dieValue
  - Play function

Main Function
- Create Game Object
- Create Die Object

Create Loaded Die Object1
Create Loaded Die object 2
Greet user
Ask user how many rounds to play
Ask user to define player 1 die sides
Is die normal or loaded?
Ask user to define player 2 die sides
Is die normal or loaded?
Loop(defined by how many rounds to play)
Player 1 turn
Player 2 turn
Higher value wins
Equal value is a draw
Store round status in player score
After rounds are complete, print out congratulations to winning player

Test Plan(Initial):

Input:	Expected Result	Actual Result:
Number of rounds(1-100)	Loops gameplay 1-100x	Loops as expected
Define the sides of the Die(1-100)	User should be able to select 1-100 and game should function as intended.	Die functions as expected, and die returns random value within range.
Select Normal die/loaded	Correct die functions as Defined by its class	Loaded Die is slightly skewed to favor higher numbers. Loaded Die almost always wins except in very rare cases.
Player 1, sides 6, Loaded Die Player 2, sides 6, Regular Die 10 Rounds	Player 1 wins almost everytime	Player 1 wins everytime, although some cases are very close. If given enough tries, player 1 might eventually lose.
Player 1, sides 6, Loaded Die Player 2, sides 6, Loaded Die 10 Rounds	Appears to have somewhat of a random result, with each player winning at random.	Appears to have somewhat of a random result, with each player winning at random.
Player 1, sides 6, Regular Die Player 2, sides 6, Regular Die 10 Rounds	Appears to have somewhat of a random result, with each player winning at random.	Appears to have somewhat of a random result, with each player winning at random.


Die
int Sides
Die Roll
Loaded Die
int Sides
Die Roll
Game Class
Die
Loaded Die
GamePlay