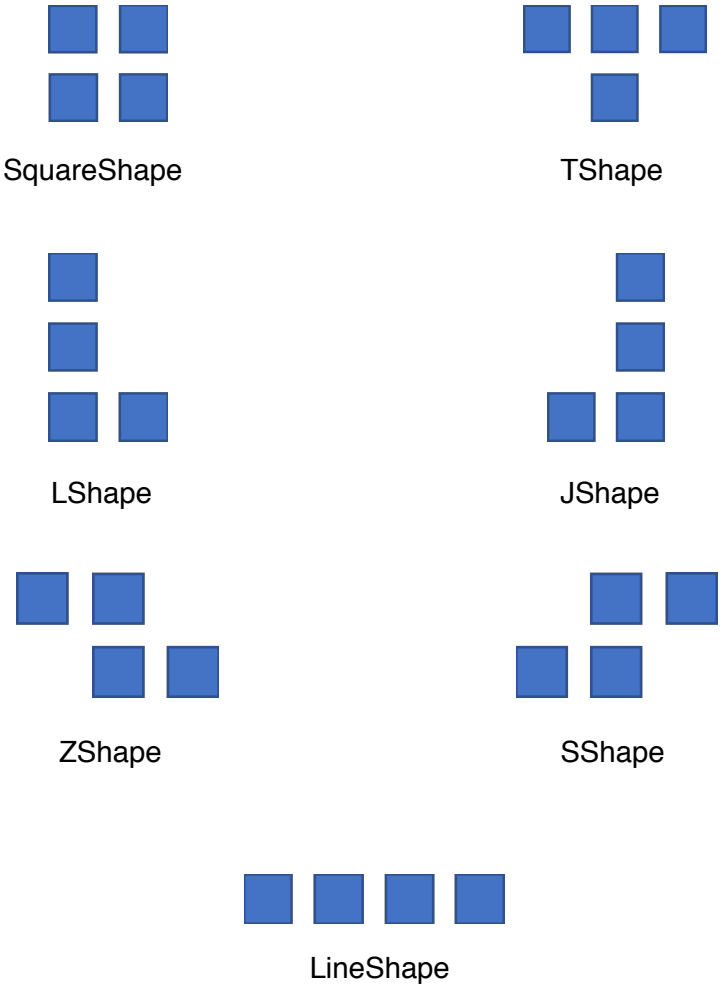
**Anna Systaliuk**
**PIC 10C: Advanced Programming**
**Final Project Report**

I made Tetris game that is objectively played as the following. The user will be presented with a block (different blocks have different shapes, I called them: TShape, SquareShape, LineShape, LShape, JShape, SShape, ZShape).




I developed the game using Swift language (version 3). I used Xcode as my IDE (only native swift IDE, other IDEs may exist but none are fully compatible like Xcode ), which functionality wise resemble QT in many ways, that is both IDEs have a very easy interface to work with, and we can use a special “tools” that

help us design the user interface in both IDE's (Qt Designer in Qt and Storyboard in Xcode).
Furthermore, I used the concepts described in the class discussing Inheritance and Polymorphism by constructing a class Shape (Parent), which hold the properties the all other shapes hold, ranging from behavioral properties like (moveTo function which notifies a block to move to a given row, and columns ) to appearance properties like (blocks color, and blocks order). Hence, all other shapes mentioned above inherit from the Shape class, with overriding some functions that distinguish shapes from one another.
An interesting distinction I noticed while working with swift is its inability to work with pointers ( just like java, swift doesn't allow user to create/handle pointers ).
I have also got a chance to use lambda functions in my program (known as closure expression in swift), an example of that is the function that creates the "collapse" animation after the blocks are destroyed, after this function is executed, another set of statements get called in the same closure.
I have indirectly used the built in iterators during the process of creating my game, and I used the word indirectly because Swift is one of the languages that automatically detect the needed type of a variable, for example:
`var name = "Anna"` in swift is equivalent to `string name = "Anna"` in C++. So to iterate through a built in container, all what we need to do is type `for block in container.blocks { … }`, which will iterate through all the elements inside of containers.blocks and call them block, which we can operate on at each iteration.
I also made a GitHub repository for the game that have the code and the proper description for the game, and a demonstration of how the game works.