Professor: Ph.D Salima Hassaine
LaSalle College Montreal

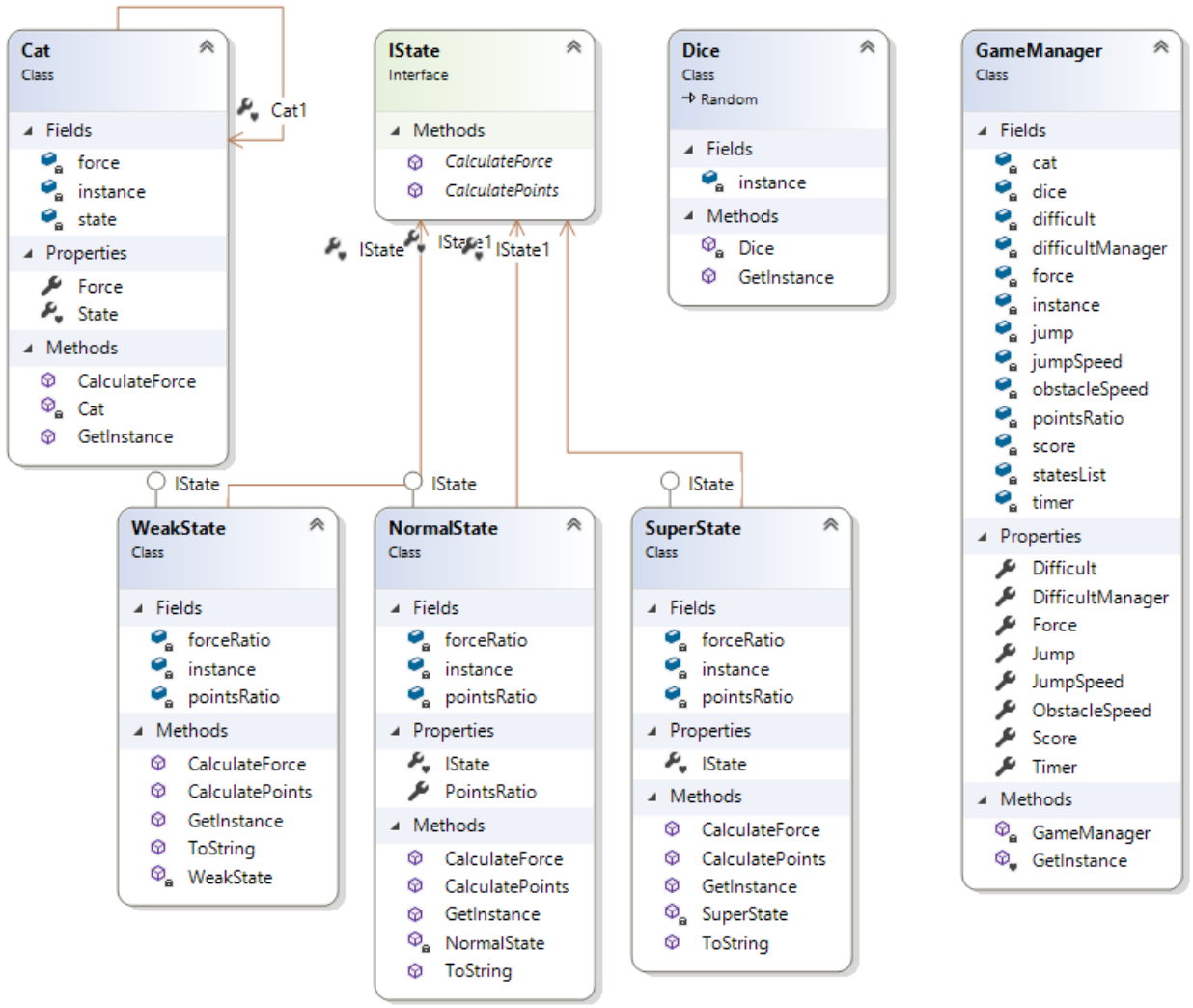
CAT ENDLESS RUN

Object Oriented Programming II

by Philipe Gouveia
Fall 2019 Project

ABOUT THE GAME / SOURCE CODE:

- **Class Diagram:**



- **Explanation:**

The game is based on a singleton class that handles the player character (a cat). This class instance is called in the GameManager Class, which is another Singleton Class. The class GameManager handles all the main functions of the game. However, those functions are called in the Form1 class, which is used to update the WinForm.

Other than above stated, the game structure has another singleton Class which is the Dice Class. The Dice instance inherits from the .Net Random Class; therefore, we can use within the instance of the Dice Class Random Class' methods.

In addition, there is the classes which implements the State Pattern, such as the interface IState and the childs classes “weakstate”, “normalstate” and “superstate”. Those classes are all singleton and implemented inside the Cat Class, as they represent states of the Cat Class’ instance.

- **About the Game Play:**

As the concept of the game is based on an Endless Run, there is no need to use arrow keys, due to the character auto acceleration.

The only keyboard keys necessary are the following:

1. **“Space” key:** is used to make the Cat instance jump above the obstacles.
2. **“R” key:** is used to reset the game.
3. **“Escape” key:** is used to quit the game.

At the beginning of the game the GameManager randomly chooses a state for the player (between the 3 possible states). Each state alters the player’s jump height and the score ratio, as below:

1. **Weak State:** can jump **9** canvas measurements and has a score ratio of **+1**
2. **Normal State:** can jump **12** canvas measurements and has a score ratio of **+2**
3. **Super State:** can jump **16** canvas measurements and has as core ratio of **+3**

Aside from randomly choosing the initial state at the beginning of the game, after every 10 score points the GameManager increases the speed of the obstacles in 20% and alters (randomly) the Cat instance’s state.

- **Object Oriented Programming Implementation:**

Abstraction: implemented on the classes Cat and GameManager as fields, such as most attributes were hidden from other classes.

Encapsulation: such as the abstraction the encapsulation was implemented in both the GameManager and the Cat classes where the attributes are hidden and they can be accesses throw the public properties.

Inheritance: implemented in the State Pattern and in the Dice class.

Polymorphism: implemented in the IState childs, where they override the ToString() method.

Singleton Pattern: implemented in the Cat, GameManager, Dice, NormalState, WeakState, SuperState classes.

State Pattern: implemented on the chain of the parent IState interface and its childs classes: WeakState, NormalState, SuperState.