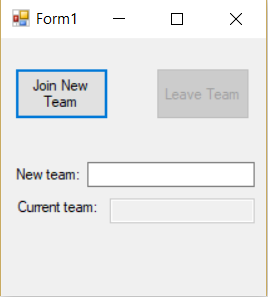
Jacob Scholl

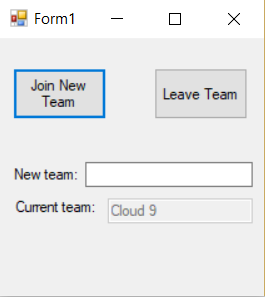
Prof. Retterer, Retired

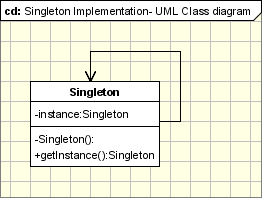
Design Patterns

27 August 2017

Introduction: Do the Singleton Pattern.

Narrative: This project illustrates the Singleton Pattern with a simulation of being a member of an esports team. Since players can only be on one team at a time, they must first leave their current team before joining a new one. This is illustrated by using a Singleton object that holds the name of the team that the user is currently a “member” of.

When a name is typed into the “New team:” text box and the “Join New Team” button is clicked, that information is stored in a Singleton object. If the user attempts to then join a different team, the “Current team” text box will not update. Instead, the original team Singleton’s name remains displayed. This is because the Singleton class does not allow more than one Singleton object to exist at any one time. For example, if “Cloud 9” is joined, a Singleton that holds the name “Cloud 9” is created, that name will appear in the “Current team” box until the “Leave Team” button is clicked. The “Leave Team” button operates by deleting the existing Singleton object, allowing a new one to be created. Then, a new Singleton can be created.

The defining characterics of the Singleton Pattern are that it only allows one Singleton to exist at any given time and that this Singleton can be accessed globally. This functionality is illustrated in this example program because the user can only join one team at a time (create one Singleton at a time). The program refuses to let the user join another team (create another Singleton) until the existing team is left (original Singleton is deleted).

Observations: This pattern is a good introductory pattern for this class. It’s a very basic class with a relatively straightforward implementation but an interesting curveball thrown in. I had issues finding the correct variables and methods to define as static, but otherwise, creating this program went smoothly.

**Form1.cs**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** SingletonPattern

**{**

**public** **partial** class Form1 **:** Form

**{**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

**private** void joinButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**newTextBox**.**Text **==** ""**)**

**{**

**return;**

**}**

string newTeam **=** newTextBox**.**Text**;**

currentTextBox**.**Text **=** nsSingleton**.**Singleton**.**getInstance**(**newTeam**).**getTeamName**();**

leaveButton**.**Enabled **=** **true;**

**}**

**private** void leaveButton\_Click**(object** sender**,** EventArgs e**)**

**{**

nsSingleton**.**Singleton**.**getInstance**(**""**).**delete**();**

currentTextBox**.**Text **=** ""**;**

leaveButton**.**Enabled **=** **false;**

**}**

**}**

**}**

**Singleton.cs**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** nsSingleton

**{**

**public** class Singleton

**{**

**private** static Singleton instance**;** // our lonely Singleton object

**private** string teamName **=** **null;**

// constructor is private

**private** Singleton**(**string newName**)**

**{**

teamName **=** newName**;**

**}**

// allow access to the Singleton when trying to join a team

**public** static Singleton getInstance**(**string newName**)**

**{**

// ensure only one Singleton object exists (thus instance = null)

**if** **(**instance **==** **null)**

**{**

instance **=** **new** Singleton**(**newName**);**

**}**

// we now have a Singleton

**return** instance**;**

**}**

// get the current team name stored in the Singleton

**public** string getTeamName**()**

**{**

**return** instance**.**teamName**;**

**}**

// delete the Singleton, allowing for a new Singleton to be made

**public** void delete**()**

**{**

instance **=** **null;**

**}**

**}**

**}**