```csharp
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace coursework
{
    class Textbox : InteractiveObject
    {
        private KeyboardState currentKeyboardState;
        string dataType;
        int maxLength;

        Keys[] keysToCheckStr = new Keys[] {
        Keys.A, Keys.B, Keys.C, Keys.D, Keys.E,
        Keys.F, Keys.G, Keys.H, Keys.I, Keys.J,
        Keys.K, Keys.L, Keys.M, Keys.N, Keys.O,
        Keys.P, Keys.Q, Keys.R, Keys.S, Keys.T,
        Keys.U, Keys.V, Keys.W, Keys.X, Keys.Y,
        Keys.Z, Keys.Back, Keys.Space,
        Keys.D0, Keys.D1, Keys.D2, Keys.D3, Keys.D4,
        Keys.D5, Keys.D6, Keys.D7, Keys.D8, Keys.D9 };

        Keys[] keysToCheckInt = new Keys[] {
        Keys.D0, Keys.D1, Keys.D2, Keys.D3, Keys.D4,
        Keys.D5, Keys.D6, Keys.D7, Keys.D8, Keys.D9,
        Keys.Back };

        private KeyboardState lastKeyboardState;

        public Textbox(Rectangle rectangle, Texture2D texture, SpriteFont font, string text, string dataType, int maxLength)
        {
            this.rectangle = rectangle;
            this.texture = texture;
            this.font = font;
            this.text = text;
            this.dataType = dataType;
            this.maxLength = maxLength;
        }
        public override void Update(MouseState mouseState, SpriteBatch spriteBatch)
        {
            if (rectangle.Contains(mouseState.X, mouseState.Y))
            {
                if (mouseState.LeftButton == ButtonState.Pressed)
                {
                    state = State.down;
                }
                else if (state == State.none)
                {
                    state = State.hover;
                }
                else
                {
                    state = State.up;
                }
            }
            else if (state == State.up && mouseState.LeftButton == ButtonState.Pressed)
            {
                state = State.none;
            }
            else if (state != State.up)
            {
                state = State.none;
            }
            Draw(spriteBatch);

            if (state == State.up)
            {
                Keys[] keysToCheck;
                currentKeyboardState = Keyboard.GetState();
                if (dataType == "str")
                {
                    keysToCheck = keysToCheckStr;
                }
                else
                {
                    keysToCheck = keysToCheckInt;
                }
                foreach (Keys key in keysToCheck)
                {
                    if (CheckKey(key) && (text.Length < maxLength || key == Keys.Back))
                    {
                        AddKeyToText(key);
                        break;
                    }
                }


                lastKeyboardState = currentKeyboardState;
            }
        }
        private void AddKeyToText(Keys key)
        {
            string newChar = "";

            if (text.Length >= 20 && key != Keys.Back)
                return;

            switch (key)
            {
                case Keys.A:
                    newChar += "a";
                    break;
                case Keys.B:
                    newChar += "b";
                    break;
                case Keys.C:
                    newChar += "c";
                    break;
                case Keys.D:
                    newChar += "d";
                    break;
                case Keys.E:
                    newChar += "e";
                    break;
                case Keys.F:
                    newChar += "f";
                    break;
                case Keys.G:
                    newChar += "g";
                    break;
                case Keys.H:
                    newChar += "h";
                    break;
                case Keys.I:
                    newChar += "i";
                    break;
                case Keys.J:
                    newChar += "j";
                    break;
                case Keys.K:
                    newChar += "k";
                    break;
                case Keys.L:
                    newChar += "l";
                    break;
                case Keys.M:
                    newChar += "m";
                    break;
                case Keys.N:
                    newChar += "n";
                    break;
                case Keys.O:
                    newChar += "o";
                    break;
                case Keys.P:
                    newChar += "p";
                    break;
                case Keys.Q:
                    newChar += "q";
                    break;
                case Keys.R:
                    newChar += "r";
                    break;
                case Keys.S:
                    newChar += "s";
                    break;
                case Keys.T:
                    newChar += "t";
                    break;
                case Keys.U:
                    newChar += "u";
                    break;
                case Keys.V:
                    newChar += "v";
```

```
                        break;
                    case Keys.W:
                        newChar += "w";
                        break;
                    case Keys.X:
                        newChar += "x";
                        break;
                    case Keys.Y:
                        newChar += "y";
                        break;
                    case Keys.Z:
                        newChar += "z";
                        break;
                    case Keys.D0:
                        newChar += "0";
                        break;
                    case Keys.D1:
                        newChar += "1";
                        break;
                    case Keys.D2:
                        newChar += "2";
                        break;
                    case Keys.D3:
                        newChar += "3";
                        break;
                    case Keys.D4:
                        newChar += "4";
                        break;
                    case Keys.D5:
                        newChar += "5";
                        break;
                    case Keys.D6:
                        newChar += "6";
                        break;
                    case Keys.D7:
                        newChar += "7";
                        break;
                    case Keys.D8:
                        newChar += "8";
                        break;
                    case Keys.D9:
                        newChar += "9";
                        break;
                    case Keys.Space:
                        newChar += " ";
                        break;
                    case Keys.Back:
                        if (text.Length != 0)
                            text = text.Remove(text.Length - 1);
                        return;
                }
                if (currentKeyboardState.IsKeyDown(Keys.RightShift) ||
                    currentKeyboardState.IsKeyDown(Keys.LeftShift))
                {
                    newChar = newChar.ToUpper();
                }
```

```
            text += newChar;
        }

        private bool CheckKey(Keys theKey)
        {
            return lastKeyboardState.IsKeyDown(theKey) && currentKeyboardState.IsKeyUp(theKey);
        }

    }
}

```