Advanced Programming C#

Lecture 2

dr inż. Małgorzata Janik

malgorzata.janik@pw.edu.pl

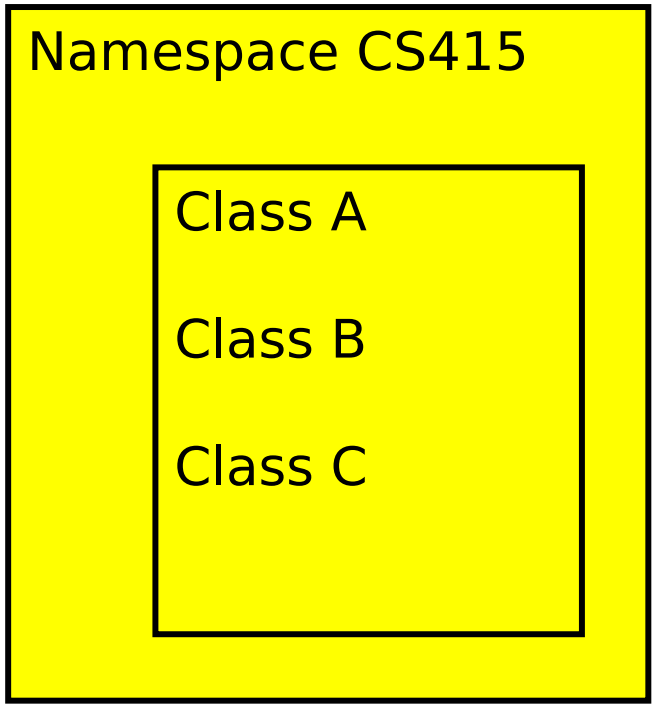
Winter Semester 2018/2019

C# Classes, Properties, Controls

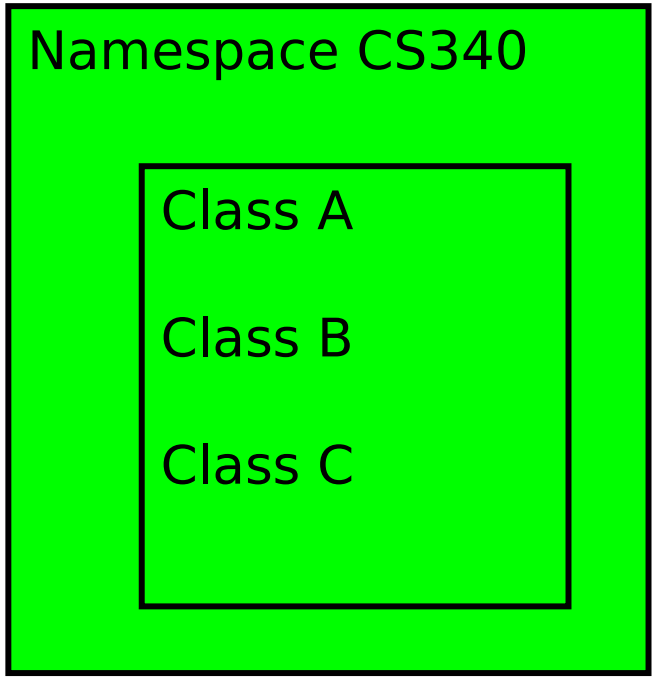
Constructions of Note

- `using`
  - like import in Java: bring in namespaces
- `namespace`
  - disambiguation of names
  - like Internet hierarchical names and Java naming
- `class`
  - like in Java
  - single inheritance up to object

Namespaces



- Permits isolation of names
- Can be nested
- Access via fully qualified names




```
CS415.A…

CS340.A…
```

Language Features : Classes

- Single inheritance
- Multiple interface implementation
- Class members
  - Constants, **fields**, methods, **properties**, indexers, **events**, operators, constructors, destructors
  - Static and instance members
  - Nested types
- Member access
  - Public, protected, internal, private

Properties

- Properties are “smart fields”
  - Natural syntax, accessors, inlining

```
public class Button: Control
{
   private string caption;

   public string Caption {
      get {
         return caption;
      }
      set {
         caption = value;
         Repaint();
      }
   }
}
```

```
Button b = new Button();
b.Caption = "OK";
String s = b.Caption;
```

foreach loop

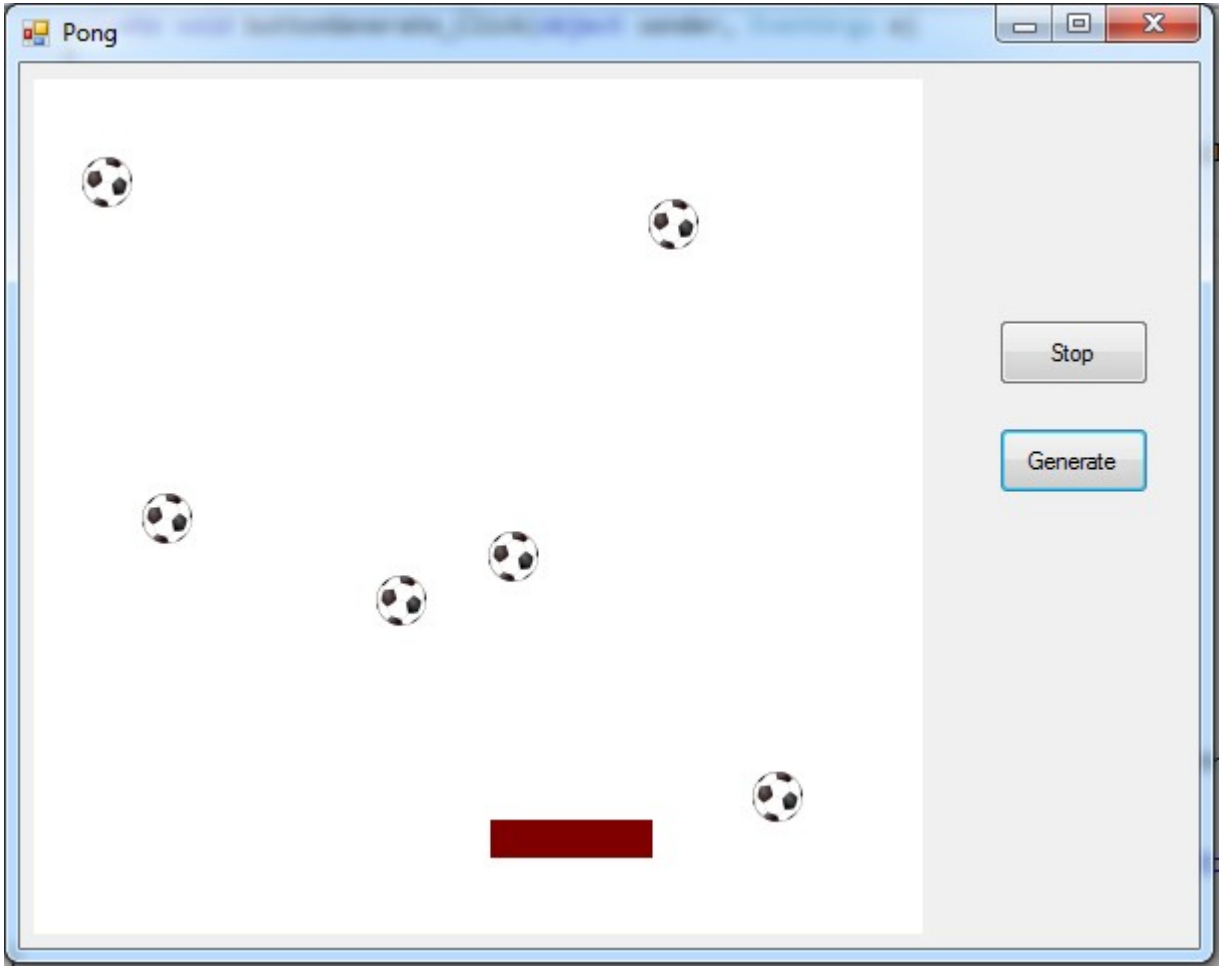
- Iteration of arrays

```
public static void Main(string[] args) {
   foreach (string s in args) Console.WriteLine(s);
}
```

- Iteration of user-defined collections

```
foreach (Customer c in customers.OrderBy("name")) {
   if (c.Orders.Count != 0) {
      ...
   }
}
```

Pong multi-ball game

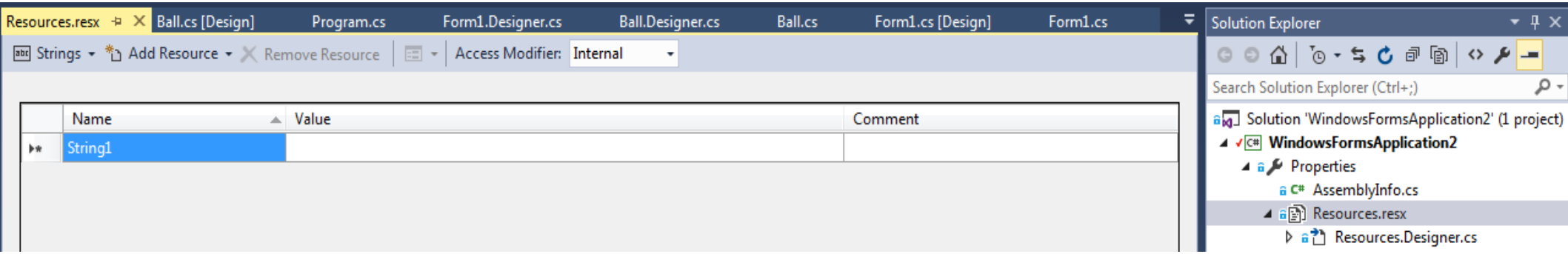


Windows Forms

Adding external resources

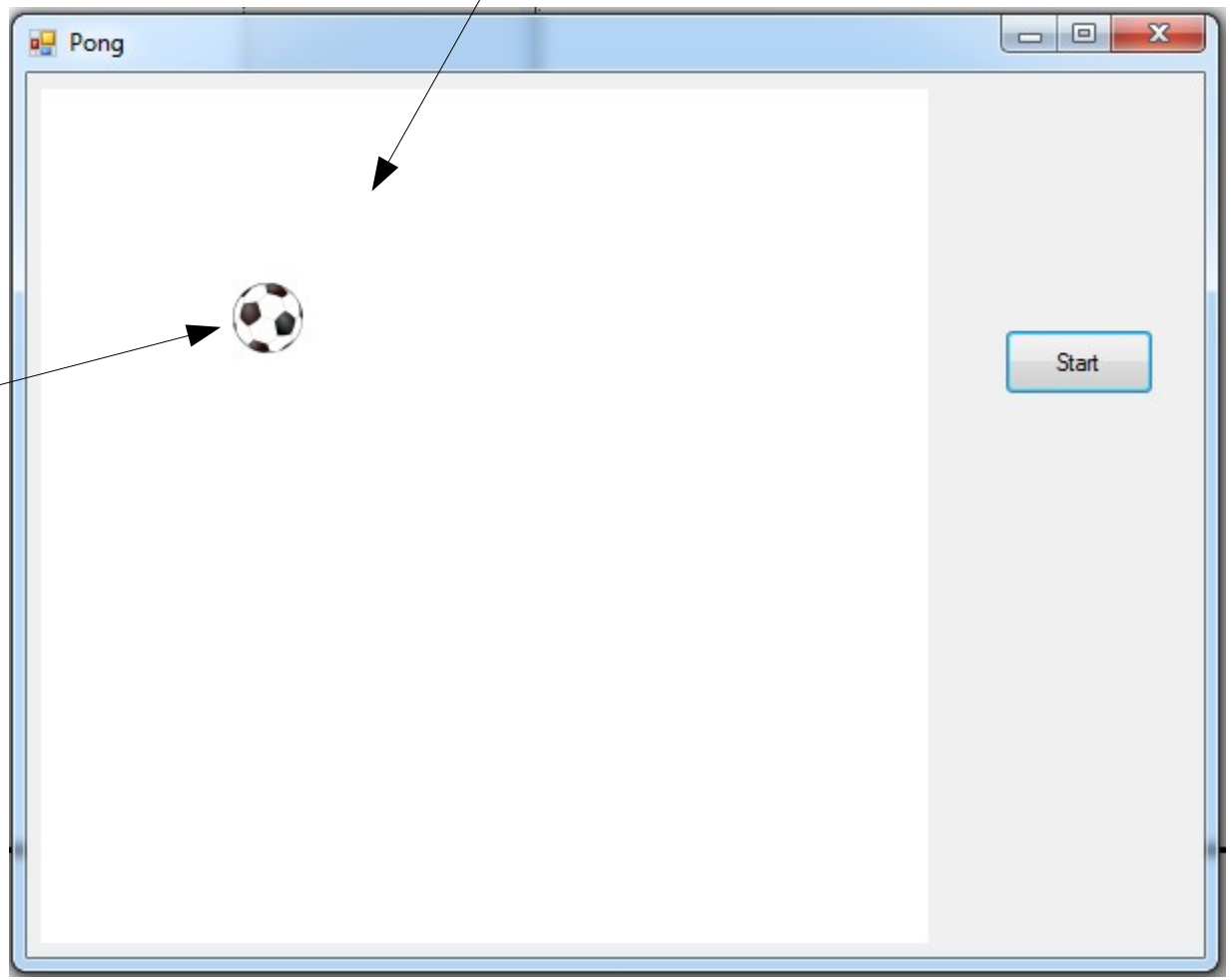
- Solution Explorer
  - Properties
    - Double click Resources.resx
      - Add Resource...
        - Add Existing File...

http://images.all-free-download.com/images/graphiclarge/soccer_ball_clip_art_13012.jpg



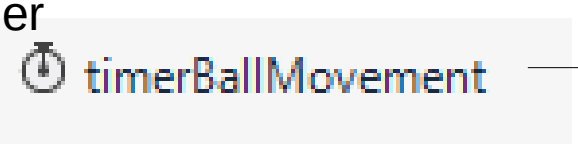
Task 1: PictureBox Movement

Panel



PictureBox
- Image (loaded as a resource)
- SizeMode
- Should move after clicking „Start”

+ timer

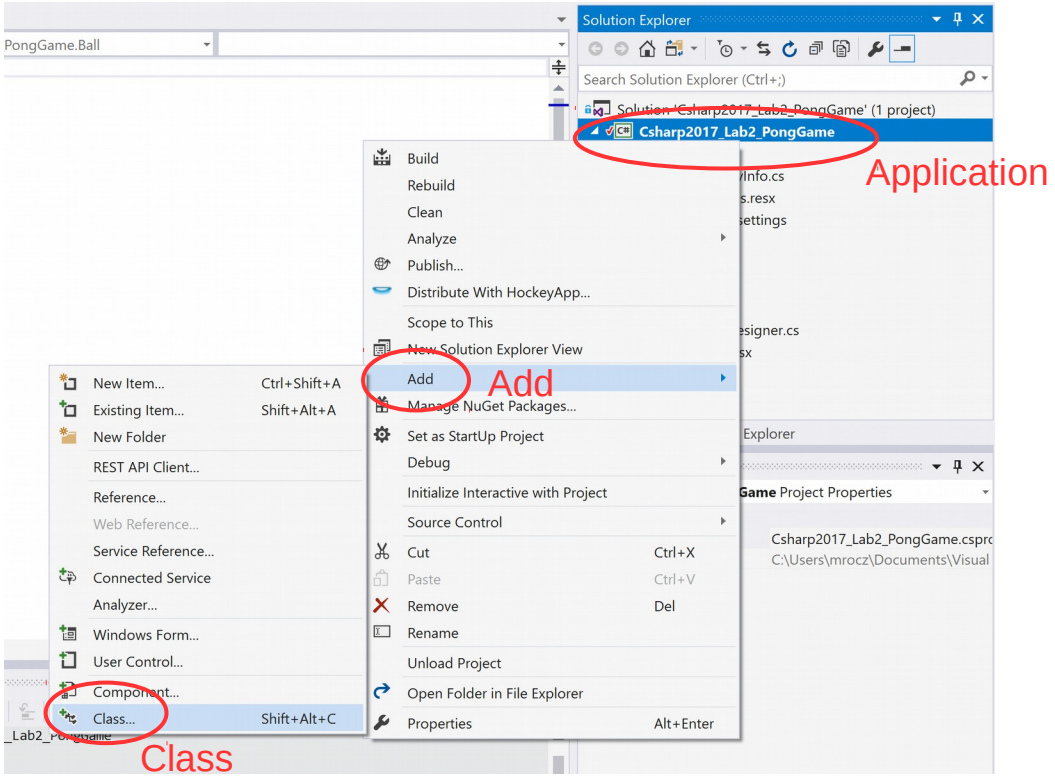


→ only one event

```
Change of the control position:
ball.Location = new Point(10,10);
```

Creating new class

- We want the ball to have additional property: velocity (Vx, Vy).
  - We want to extend PictreBox class, adding this additional property.
- Application → Add → Class...
  - Ball.cs




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication_Lab2_Pong
{
    class Ball
    {
    }
}
```

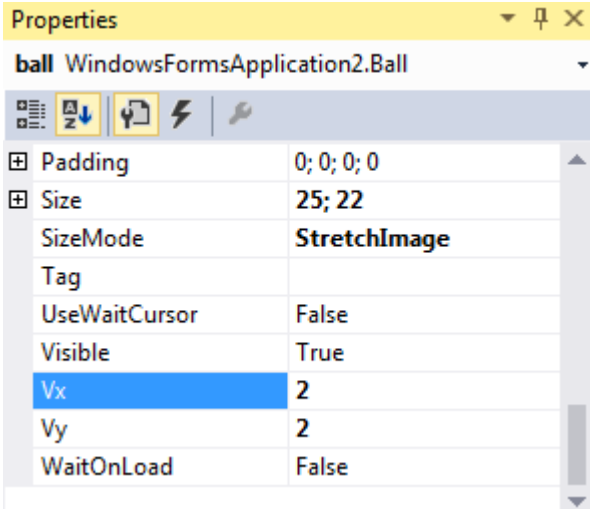
Class inheritance

- Inheritance **`Ball : PictureBox`**
- PictureBox exists in `System.Windows.Forms` namespace.
- Fields can be created and immidately initialized.

```
using System.Windows.Forms;

namespace WindowsFormsApplication_Lab2_Pong
{
    class Ball :  PictureBox
    {
        private double vx = 2;
        private double vy = 2;


    }
}
```

- Build & Start your project
- You can now add Ball control from the Toolbox!

Properties

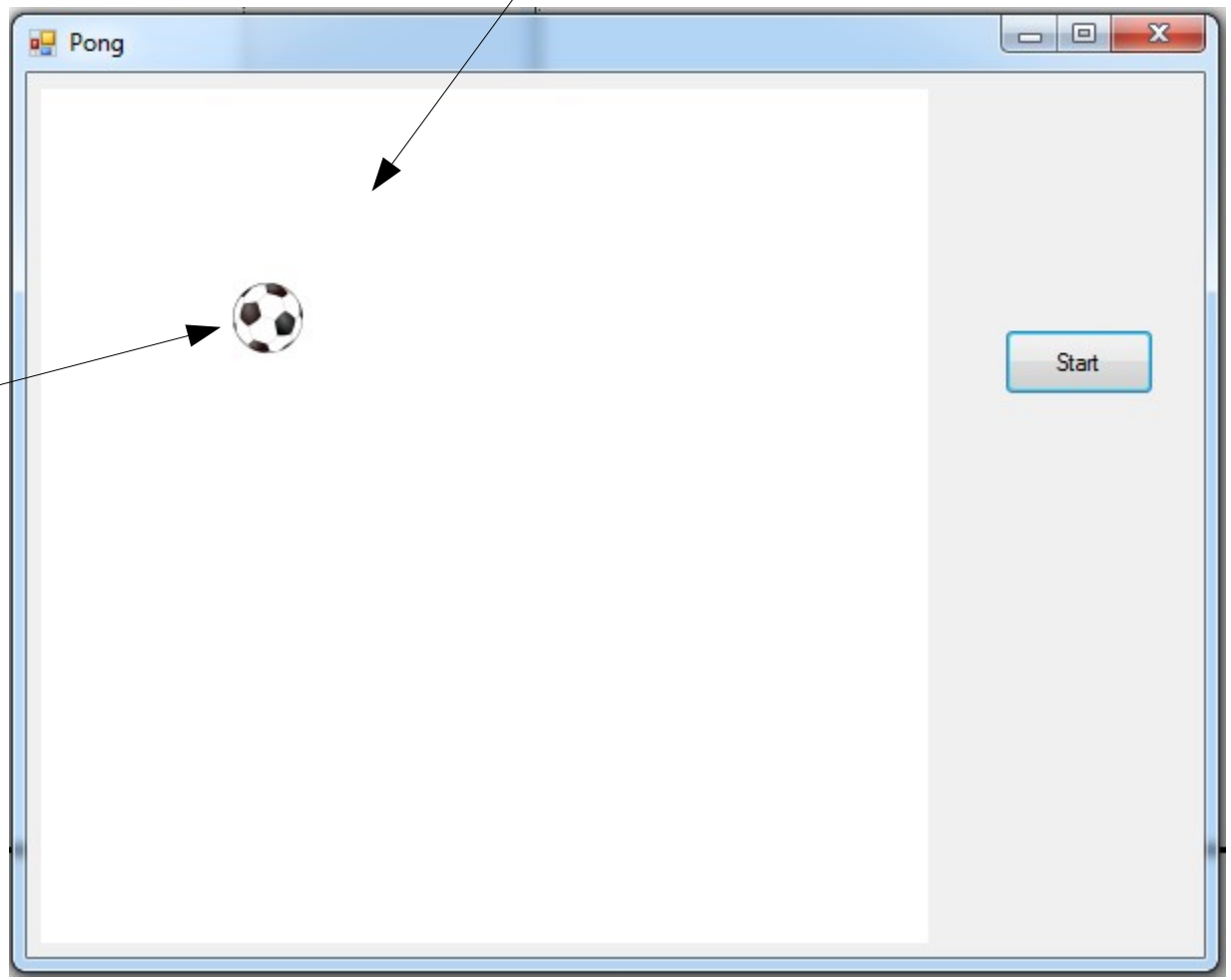


Remark: Use Invalidate(); Repaint(); and Update(); methods in case of redrawing.

```
class Ball :  PictureBox
{
  private double vx = 2;
  private double vy = 2;

  public double Vx
        {
            get
            {
                return vx;
            }
            set
            {
                vx = value;
            }
        }

        public double Vy
        {
            get
            {
                return vy;
            }
            set
            {
                vy = value;
            }
        }

}
```

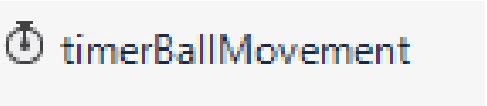
Task 2: Ball Movement

Panel

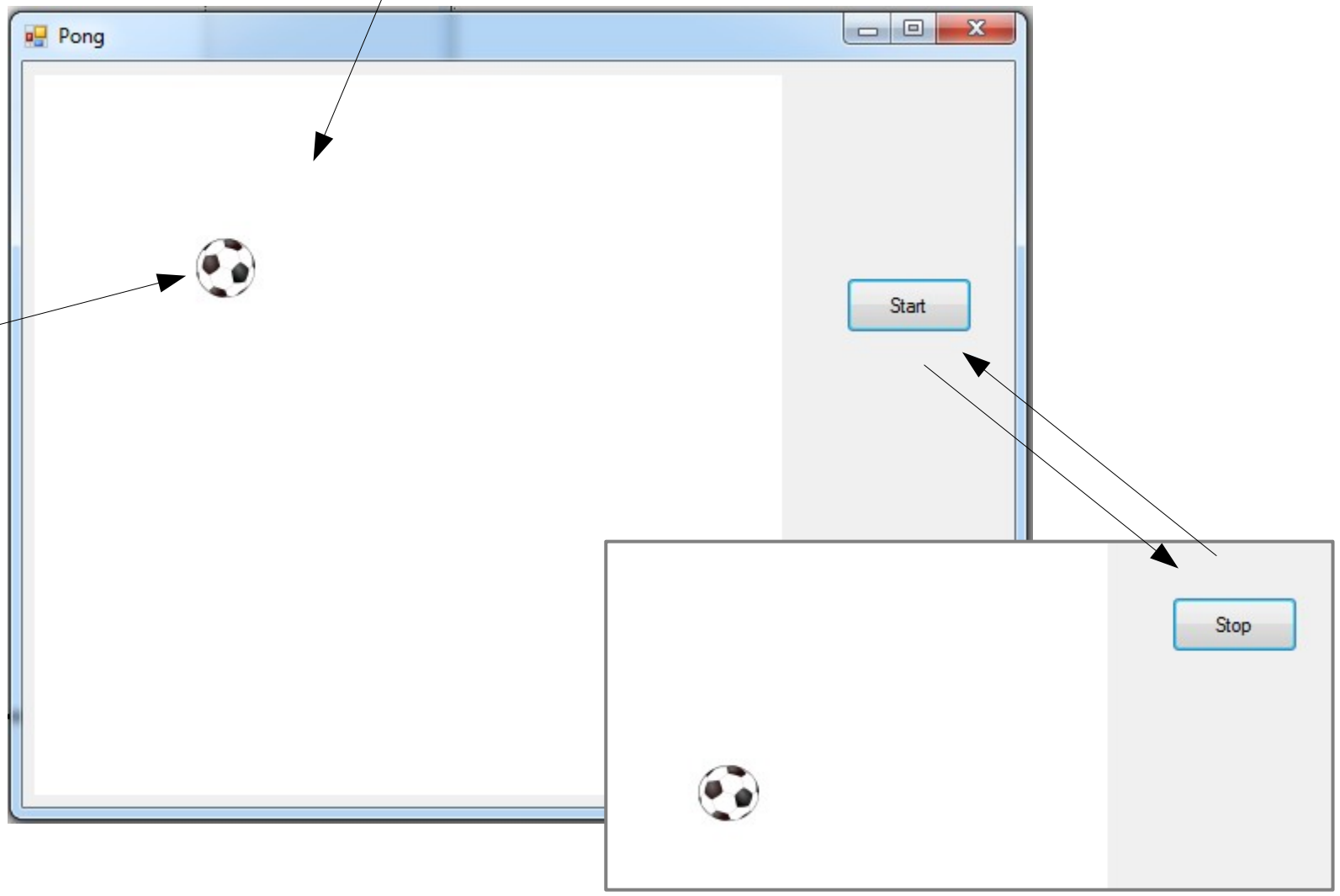


Ball
- Image (loaded as a resource)
- SizeMode
- **Should move after clicking „Start”, Vx pixels right and Vy pixels down**

+ timer



Panel



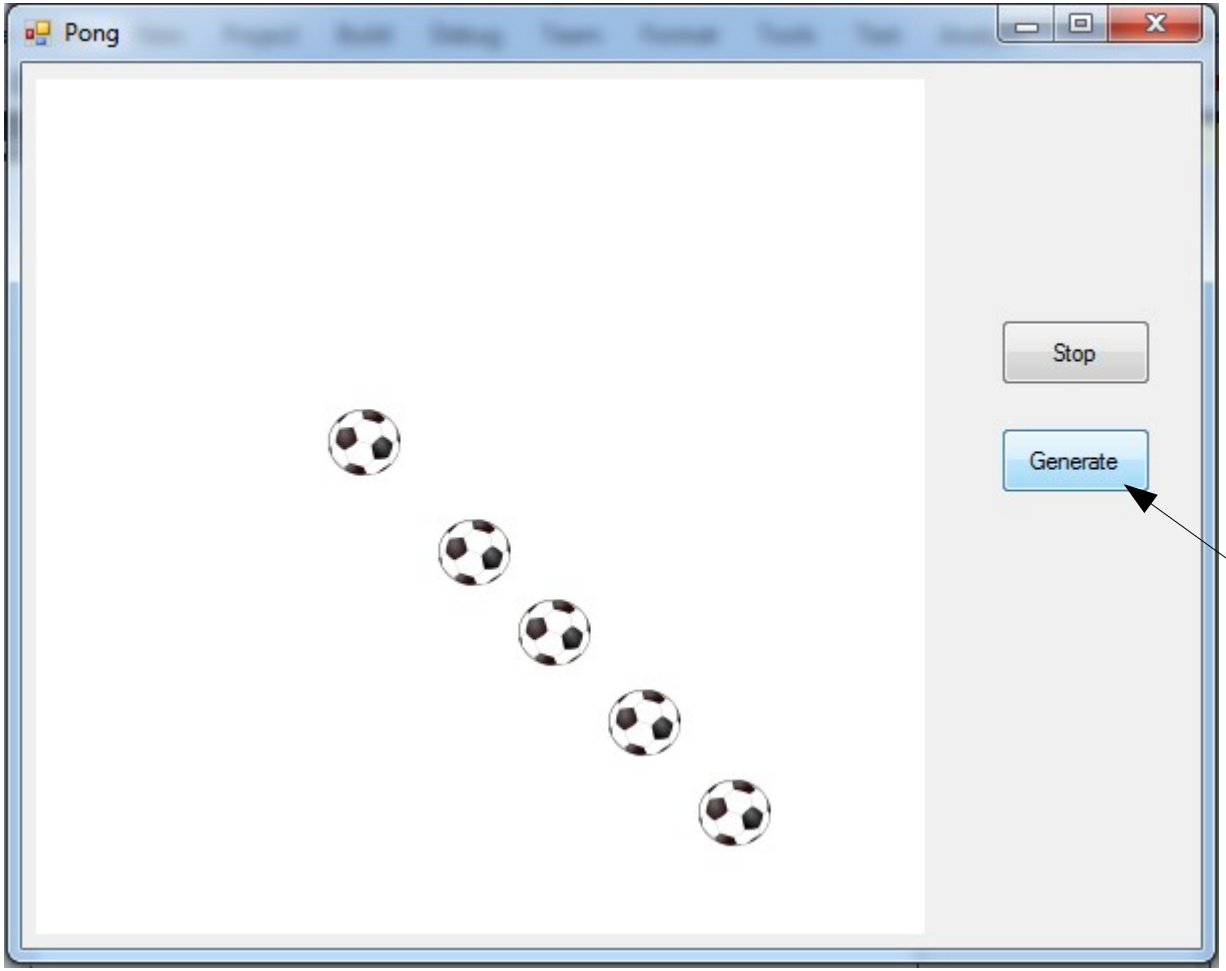
Ball
- Image (loaded as a resource)
- SizeMode
- **Should move after clicking „Start”, Vx pixels right and Vy pixels down**

+ timer

timerBallMovement

Possibility to Stop and Start ball movement.

Task 3: Ball Generation



Generates new ball

Collections (List)

- We can create empty list with:

```
List<int> listInt = new List<int>();

List<Ball> listBalls = new List<Ball>();
```

- Adding to the list:

```
listInt.Add(3);
listBalls.Add(ballFirst);
```

Collections (List)

- We can create empty list with:

```
List<int> listInt = new List<int>();

List<Ball> listBalls = new List<Ball>();
```

- Adding to the list:

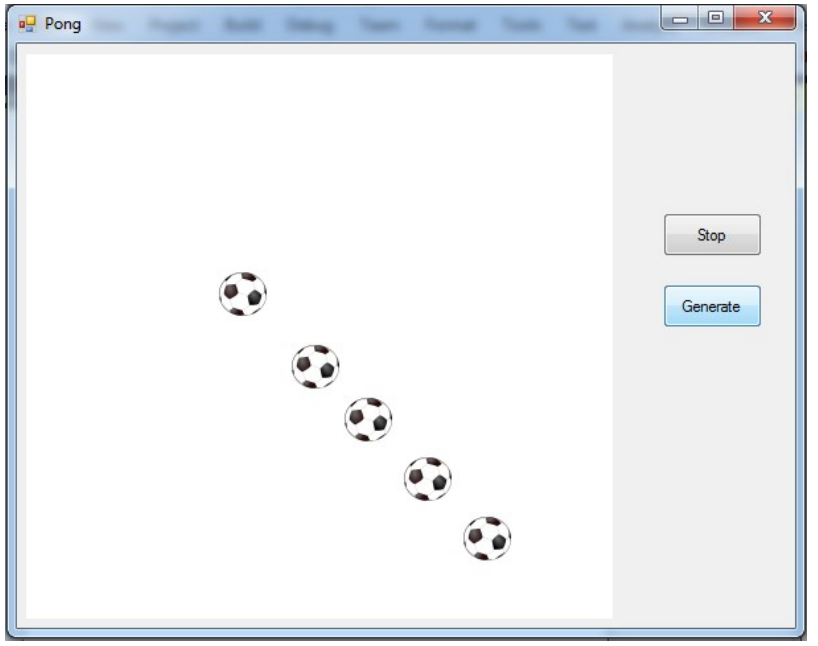
```
listInt.Add(3);
listBalls.Add(ballFirst);
```

- Iterating:

```
foreach (Ball ball in listBalls)
{

}
```

Suggestions

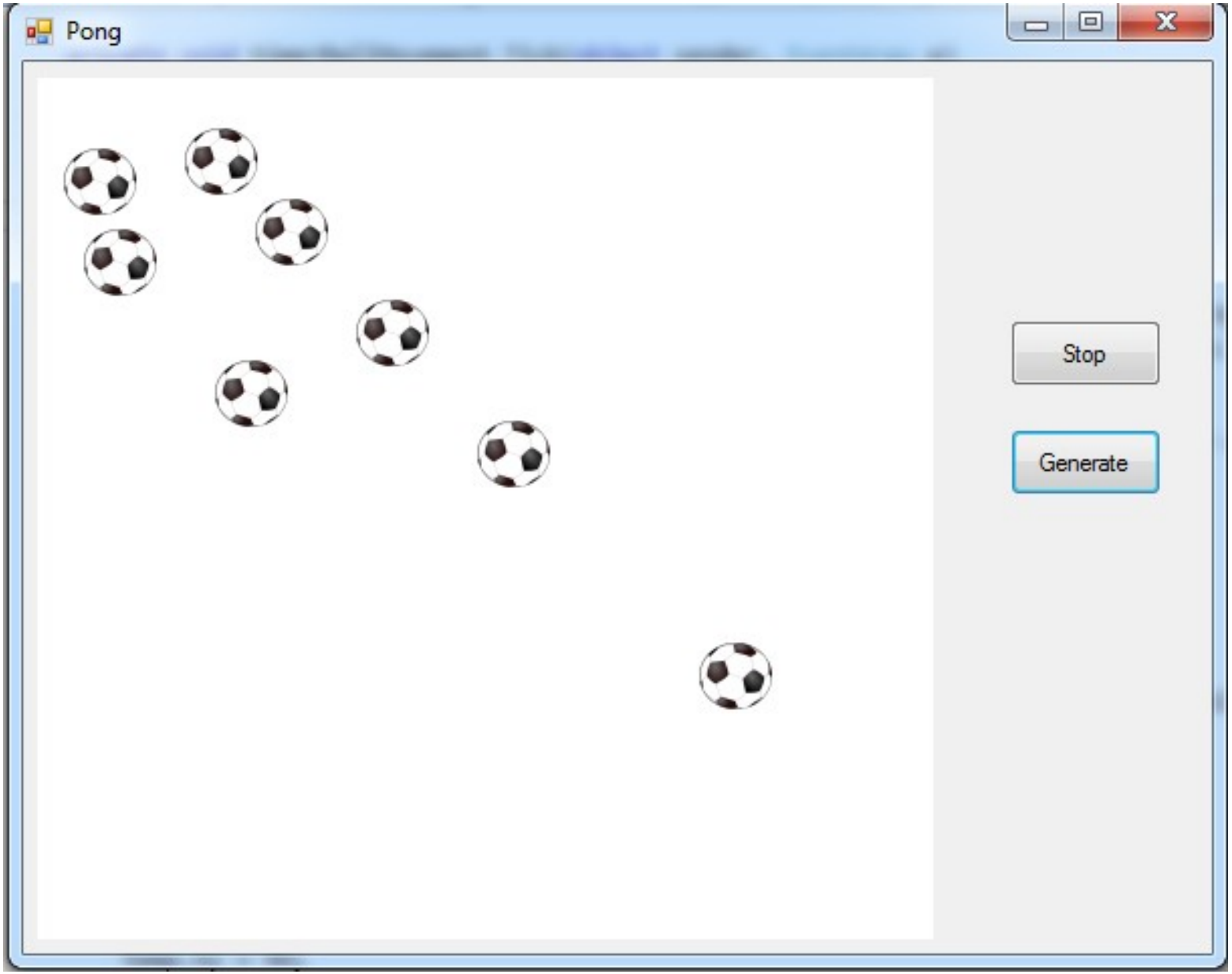
- Create new global Ball list
- Create new temp ball (new)
- Set its properties
  - Copy from generated code of Form.Designer.cs
- Add to the balls list
- Add to the panel control list



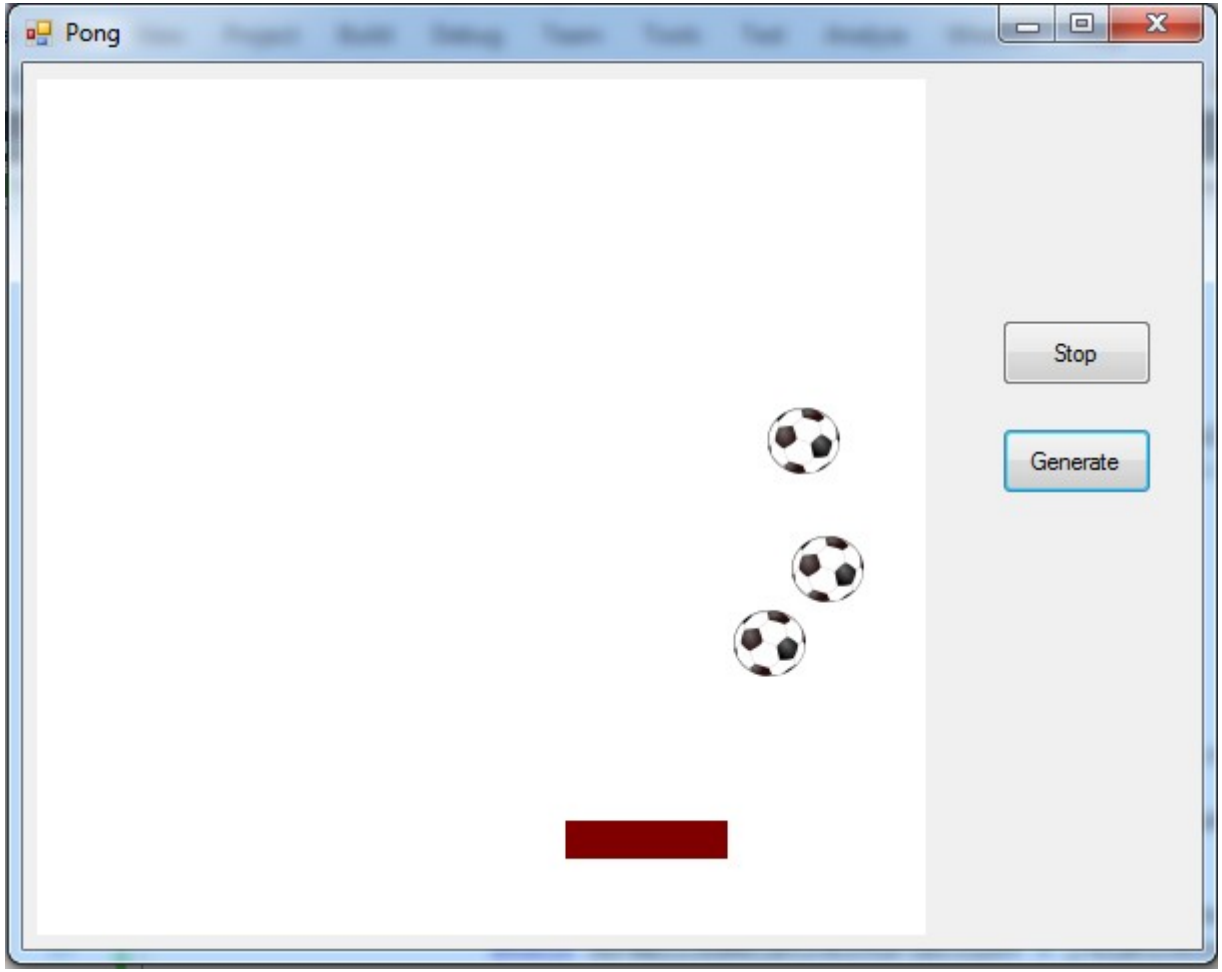

```
panelPole.Controls.Add(temp);
```

- Change location of balls using foreach loop

Collisions with walls



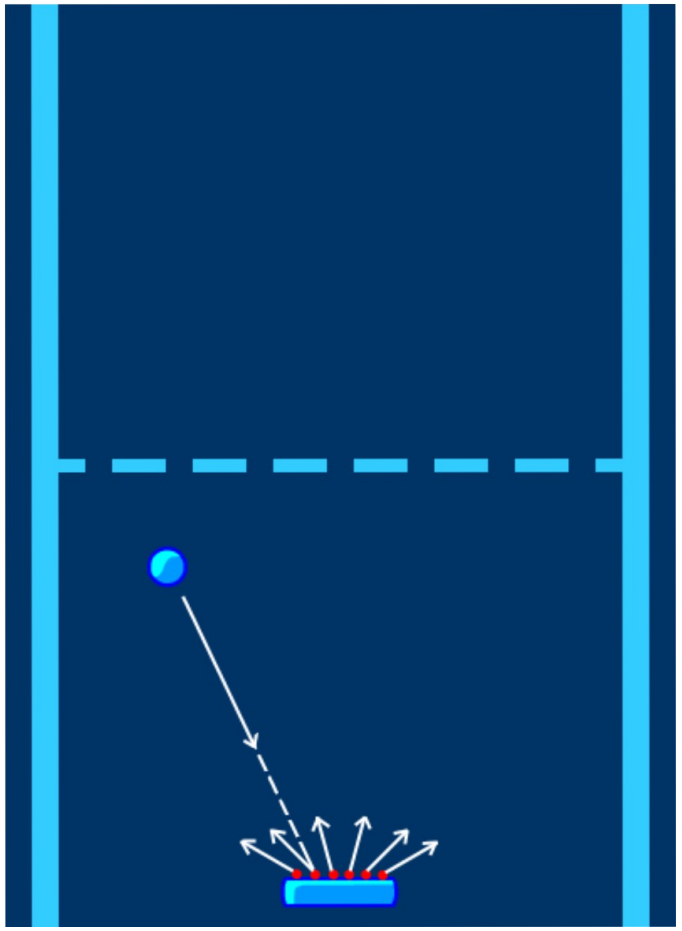
Collisions with the paddle



MouseMove

Collisions with the paddle

Ball direction after collision with paddle → only the place where ball hit the paddle (relative intersection point) is taken into account



```
double relativeIntersectY = (paddleX + (PADDLEWIDTH / 2)) - ballX;
double normalizedRelativeIntersectionY = (relativeIntersectY / (PADDLEWIDTH / 2));
double bounceAngle = normalizedRelativeIntersectionY * MAXBOUNCEANGLE;
ball.Vx = BALLSPEED * -Math.Sin(bounceAngle);
ball.Vy = BALLSPEED * Math.Cos(bounceAngle);
```

THE END

dr inż. Małgorzata Janik

malgorzata.janik@pw.edu.pl