Below is what the form looks like with Name, Text or Value for the elements from the toolbox
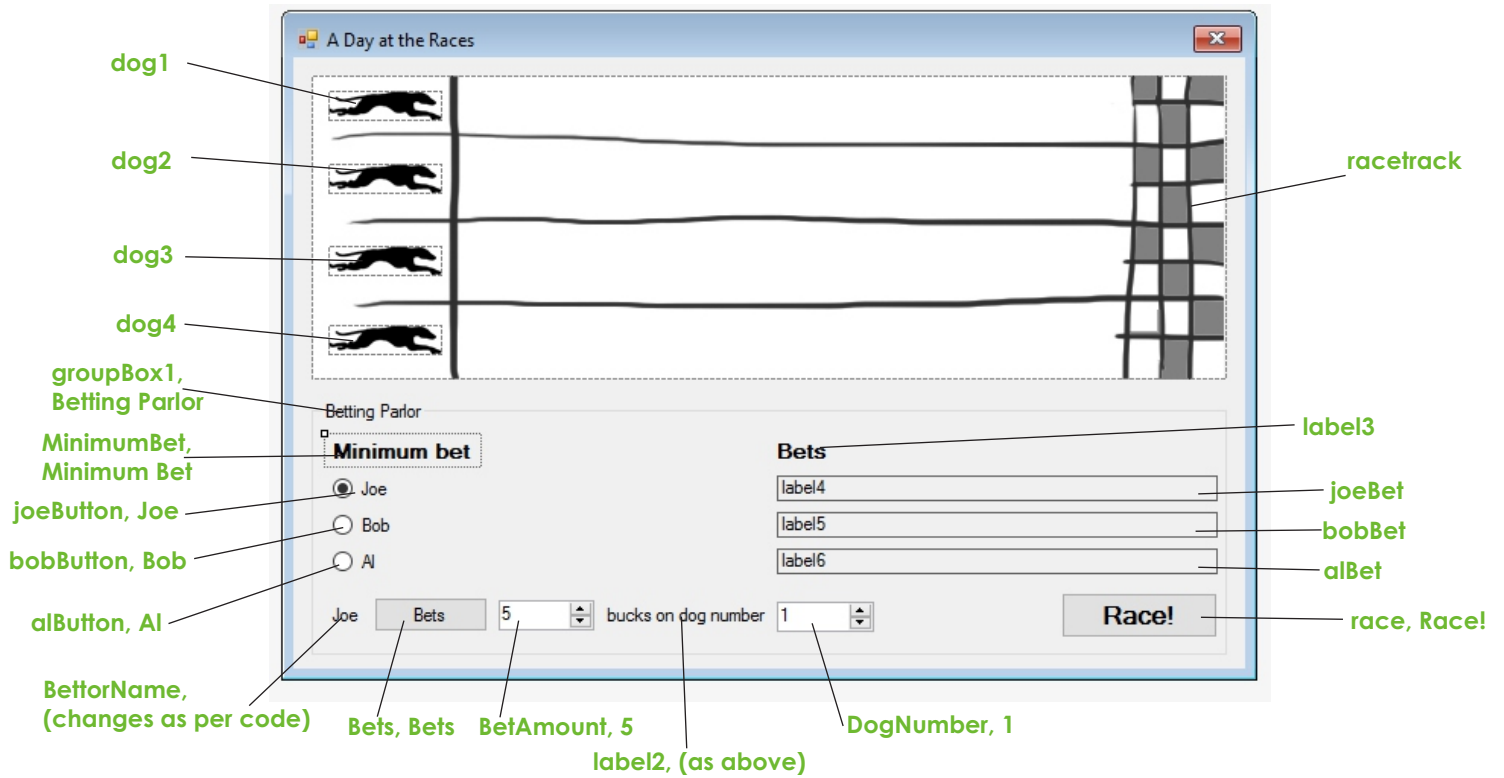
dog1 · **dog2** · **dog3** · **dog4** · **racetrack**

groupBox1, Betting Parlor
MinimumBet, Minimum Bet
joeButton, Joe
bobButton, Bob
alButton, Al
BettorName, (changes as per code)
Bets, Bets · **BetAmount, 5** · **label2, (as above)** · **DogNumber, 1**
label3 · **joeBet** · **bobBet** · **alBet** · **race, Race!**

Class greyhound

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing;

namespace RaceTrackSimulator {
    public class Greyhound {
        public int StartingPosition;
        public int RacetrackLength;
        public PictureBox MyPictureBox = null;          // What is null doing here ?
        public int Location = 0;
        public Random MyRandom;      // There is no class named Random created, Why is it mentioned here?

        public bool Run() {
            MyRandom = new Random();
            int distance = MyRandom.Next(1, 5);    // How is Next moving picturebox randomly from 1 - 5 ?

            MoveMyPictureBox(distance);     // Shouldn't the method at the bottom come before this line ?

            Location += distance;
            if (Location >= (RacetrackLength - StartingPosition)) {
                return true;
            }
            return false;
        }

        public void TakeStartingPosition() {     // Is this method resetting all the dog's picturebox to startingpostion?
            MoveMyPictureBox(-Location);
            Location = 0;
        }

        public void MoveMyPictureBox(int distance) {
            Point p = MyPictureBox.Location;      // Haven't explained in the book how point works
            p.X += distance;
            MyPictureBox.Location = p;
        }
    }
}
```

Class Guy

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RaceTrackSimulator {
    public class Guy {
        public string Name;
        public Bet MyBet;
        public int Cash;

        public RadioButton MyRadioButton;
        public Label MyLabel;

        public Guy(string Name, Bet MyBet, int Cash, RadioButton MyRadioButton,
Label MyLabel) {
            this.Name = Name;
            this.MyBet = MyBet;
            this.Cash = Cash;
            this.MyRadioButton = MyRadioButton;
            this.MyLabel = MyLabel;
        }

        public void UpdateLabels() {
            if (MyBet == null) {
                MyLabel.Text = String.Format("{0} hasn't placed any bets", Name);
            } else {
                MyLabel.Text = MyBet.GetDescription();
            }
            MyRadioButton.Text = Name + " has " + Cash + " bucks";
        }

        public void ClearBet() {
            MyBet.Amount = 0;
        }

        public bool PlaceBet(int Amount, int Dog) {
            if (Amount <= Cash) {
                MyBet = new Bet(Amount, Dog, this);
                return true;
            }
            return false;
        }

        public void Collect(int Winner) {
            Cash += MyBet.PayOut(Winner);
        }
    }
}
```

What is happening in this line ?

Haven't really seen "this" being used outside designer.cs what does these lines mean ?

What is happening in this line ?
I know an instance MyBet is created, what is 'this' doing ?

```csharp
public class Bet {
    public int Amount;
    public int Dog;
    public Guy Bettor;

    public Bet(int Amount, int Dog, Guy Bettor) {
        this.Amount = Amount;
        this.Dog = Dog;
        this.Bettor = Bettor;
    }

    public string GetDescription() {
        string description = "";

        if(Amount > 0) {
            description = String.Format("{0} bets {1} on dog #{2}",
                Bettor.Name, Amount, Dog);
        } else {
            description = String.Format("{0} hasn't placed any bets",
                Bettor.Name);
        }
        return description;
    }

    public int PayOut(int Winner) {
        if (Dog == Winner) {
            return Amount;
        }
        return -Amount;
    }
}
```

Again what is happening here ?

Is this line Necessary ?

Is it possible to mention these methods and variables in Guy Class so that there is no need for Bet class to be created ?

Why aren't there not more than one instance for Guy or Bet or Random Class are we dealing with 3 objects joe, bob and al ?

```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RaceTrackSimulator {
    public partial class Form1 : Form {
        Greyhound[] dogs = new Greyhound[4];
        Guy[] guys = new Guy[3];

        public Form1() {
            InitializeComponent();
            SetupRaceTrack();
        }

        private void SetupRaceTrack() {
            MinimumBet.Text = string.Format("Minimum bet {0:c}",
(int)BetAmount.Minimum);

            int startingPosition = dog1.Right - racetrack.Left;
            int raceTrackLength = racetrack.Size.Width;

            dogs[0] = new Greyhound() { MyPictureBox = dog1,
                                RacetrackLength = raceTrackLength,
                                StartingPosition = startingPosition };
            dogs[1] = new Greyhound() { MyPictureBox = dog2,
                                RacetrackLength = raceTrackLength,
                                StartingPosition = startingPosition };
            dogs[2] = new Greyhound() { MyPictureBox = dog3,
                                RacetrackLength = raceTrackLength,
                                StartingPosition = startingPosition };
            dogs[3] = new Greyhound() { MyPictureBox = dog4,
                                RacetrackLength = raceTrackLength,
                                StartingPosition = startingPosition };

            guys[0] = new Guy("Joe", null, 50, joeButton, joeBet);
            guys[1] = new Guy("Bob", null, 75, bobButton, bobBet);
            guys[2] = new Guy("Al", null, 45, alButton, alBet);

            foreach (Guy guy in guys) {
                guy.UpdateLabels();
            }
        }

        private void joeButton_CheckedChanged(object sender, EventArgs e) {
            SetBettorNameTextLabel("Joe");
        }

        private void bobButton_CheckedChanged(object sender, EventArgs e) {
            SetBettorNameTextLabel("Bob");
        }

        private void alButton_CheckedChanged(object sender, EventArgs e) {
            SetBettorNameTextLabel("Al");
        }

        private void SetBettorNameTextLabel(string Name) {
            BettorName.Text = Name;
```

Annotations:
- What is happening here ?
- Why is this all null ?
- foreach (Guy guy in guys) lol What ? Ok how does this line function. What is the function of 'in'? Why is guy there when it is never mentioned before?
- Is CheckedChanged an event ?

```csharp
        }

        private void Bets_Click(object sender, EventArgs e) {
            int GuyNumber = 0;

            if (joeButton.Checked) {
                GuyNumber = 0;
            }
            if (bobButton.Checked) {
                GuyNumber = 1;
            }
            if (alButton.Checked) {
                GuyNumber = 2;
            }

            guys[GuyNumber].PlaceBet((int)BetAmount.Value, (int)DogNumber.Value);
            guys[GuyNumber].UpdateLabels();
        }

        private void race_Click(object sender, EventArgs e) {
            bool NoWinner = true;       // Shouldn't this be true, because on click the button should be enabled right ?
            int winningDog;
            race.Enabled = false;
                                        // What events ?Where is DoEvents() method ? Is Application a class ?
            while (NoWinner) {
                Application.DoEvents();
                for (int i = 0; i < dogs.Length; i++ ) {
                    if (dogs[i].Run()) {
                        winningDog = i + 1;
                        NoWinner = false;
                        MessageBox.Show("We have a winner - dog #" + winningDog);
                        foreach (Guy guy in guys) {
                            if (guy.MyBet != null) {          // This part is really confusing me,  idk what is happening in
                                guy.Collect(winningDog);      // foreach loop.
                                guy.MyBet = null;
                                guy.UpdateLabels();
                            }
                        }
                        foreach (Greyhound dog in dogs) {
                            dog.TakeStartingPosition();
                        }
                        break;
                    }
                }
            }
                                        // Shouldn't this be false, we don't want to enable the race after the picture resets ?
            race.Enabled = true;
        }
    }
}
```

Ok here's how I think the loop functions

When NoWinner = true, run for loop

Under for loop all dogs are looped at the same time — Does this mean for loop loop all the dogs at the same time
and runs if loop — or loops them one after the other

Under if loop randomly move picture box
until the location >= racetracklength - Startingpoint

If end is reached which ever dog reaches first its value 'i' is recorded

If the winning dog is dog[0] i+1 = 0+1 =1 and dog 1 wins

So now while loop is false .

Message box of the winner displayed