Laboratório C#

Um dia de corridas

Este laboratório dá uma especificação que descreve um programa para você desenvolver usando o conhecimento obtido nos últimos capítulos.
Este projeto é maior do que os já vistos até agora. Então, leia tudo antes de começar e reserve algum tempo para pensar. E não se preocupe se ficar preso em uma parte – não há nada novo aqui.

Preenchemos alguns poucos detalhes do projeto e certificamo-nos de que você tenha todas as peças necessárias... e mais nada.

É sua responsabilidade terminar o trabalho.

Especificações: desenvolva um simulador de pista de corridas

Joe, Bob e Al gostam de apostar em corridas, mas estão cansados de perder dinheiro. Eles querem que você construa um simulador, permitindo-lhes determinar os vencedores *antes* de colocar dinheiro na coisa. E, se você fizer um bom trabalho, eles dividirão com você os lucros.

Eis como você vai desenvolver para eles...

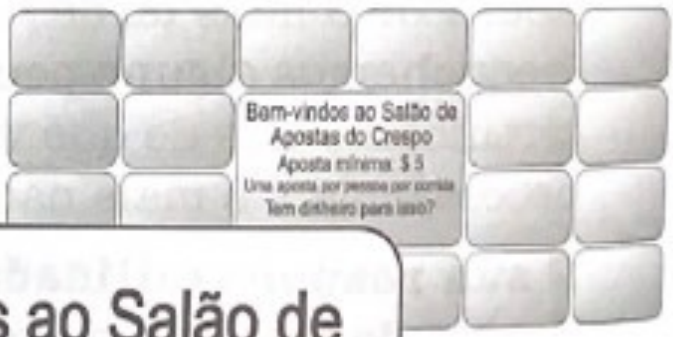
Os caras

Joe, Bob e Al querem apostar numa corrida de cachorros. Joe começa com $ 50, Bob com $ 75 e Al com $ 45. Antes de cada corrida, eles decidem se querem apostar e quanto cada um vai investir. Eles podem mudar as apostas até o início da corrida... mas, depois que ela começar, as apostas são encerradas.

O salão de apostas

O salão de apostas manterá registro do dinheiro de cada um e quais apostas fizeram. Existe um valor mínimo de apostas de $ 5. O sistema aceita somente uma aposta por pessoa em cada corrida.

O sistema checa para se certificar de que cada apostador tenha dinheiro o suficiente para cobrir sua aposta – portanto, eles não podem apostar se não tiverem dinheiro suficiente.



Bem-vindos ao Salão de Apostas do Crespo
Aposta mínima: $ 5
Uma aposta por pessoa por corrida.
Tem dinheiro para isso?

Apostando

Todas as apostas são o dobro ou nada – o vencedor dobra seu dinheiro ou perde o que apostou. Existe um limite mínimo de apostas de $ 5 e cada um pode apostar até $ 15 num único cão. Se ele ganhar, o apostador terminará com o dobro da quantidade que apostou (depois da corrida terminar). Se ele perder, aquela quantidade desaparecerá de seu total.

Todas as apostas: dobro ou nada
Aposta mínima: $ 5
Até $ 15 por cão
Vencedores: $$ adicionado
Perdedores: $$ removido

Digamos que um cara aposte £ 10 na janela. No final da corrida, se o cachorro vencer, seu dinheiro aumentará em £ 10 (porque ele mantém os £ 10 originais apostados e ganha mais £ 10 por ter vencido). Se perder, seu total diminui em £ 10.

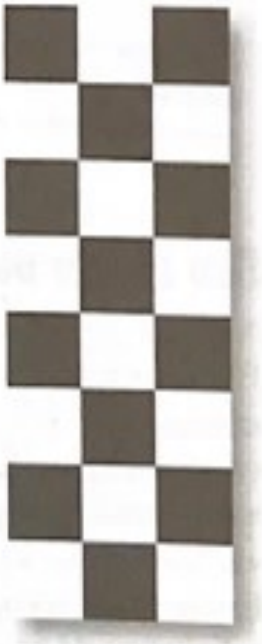
A corrida

Existem quatro cães que correm em linha reta. O vencedor da corrida é o primeiro a cruzar a linha de chegada. Uma corrida é totalmente aleatória, não existem obstáculos ou vantagens e um cão não tem mais chance de vencer as próximas corridas baseando-se no seu histórico passado.

Se você quer desenvolver um sistema com obstáculos, por favor faça isto! Será uma ótima oportunidade de praticar a escrita de um código e ainda se divertir.







Você precisará de três classes e um formulário

Você terá que desenvolver três classes principais no seu projeto, bem como uma GUI para o simulador. Você deve ter um array de três objetos `Guy` para controlar os três caras e seus resultados, e um array de quatro objetos `GreyHound` que disputarão as corridas. Além disso, cada instância de `Guy` deve ter seu próprio objeto `Bet`, que controla as apostas e paga ou toma dinheiro no final de cada corrida.

Adiantamos o seu trabalho com descrições de classes e alguns fragmentos de código. Você terá de terminar de desenvolvê-los.

Será preciso adicionar `using System.Windows.Forms` no topo das classes GreyHound e Guy. E você precisará adicionar `using System.Drawing` a Greyhound (porque o código no final dessa página usa Point, que é o que está naquele namespace). Você também precisará adicionar a palavra-chave pública na frente de cada declaração de classe.

Fornecemos o esqueleto da classe que Você precisa desenvolver. Sua tarefa é completar os métodos.

```
public class Greyhound {
  public int StartingPosition; // Onde PictureBox inicia
  public int RacetrackLength; // O tamanho da pista de corrida
  public PictureBox MyPictureBox = null; // Meu objeto PictureBox
  public int Location = 0; // Minha posição na pista
  public Random Randomizer; // Uma instância de Random

  public bool Run( ) {
    // Avance 1, 2, 3 ou 4 espaços aleatoriamente
    // Atualize a posição de PictureBox no formulário
    // Retorna true se eu ganhei a corrida
  }

  public void TakeStartingPosition( ) {
    // Redefina minha posição para a linha de partida
  }
}
```

Você só precisa de uma instância de Random – a referência Randomizer de cada Greyhound deve apontar para o mesmo objeto Random.

Adicionamos comentários para dar a você uma ideia do que fazer.

Greyhound
StartingPosition RacetrackLength MyPictureBox Location Randomizer
Run() TakeStartingPosition()

Vê como o diagrama de classe corresponde ao código?

O inicializador do objeto Greyhound é bem simples. Apenas passe uma referência para o PictureBox certo no formulário para cada objeto Greyhound.

Não quebre muito a cabeça com isso... algumas vezes, só é preciso definir uma variável e pronto.

Seu objeto pode controlar coisas em seu formulário...

A classe `Greyhound` controla a posição na pista durante a corrida. Ela também atualiza o local de `PictureBox` que representa o cão se movendo pela pista. Cada instância de `Greyhound` usa um campo chamado `MyPictureBox` para a referência de PictureBox no formulário que mostra a imagem de um cão. Suponha que a variável `distance` contenha a distância que um cão vai avançar. Então, este código vai atualizar a posição de `MyPictureBox` ao adicionar `distance` ao seu valor `X`

Você terá que ter certeza que o formulário passe o PictureBox correto para o inicializador de objeto de cada Greyhound.

Você obtém a posição atual da imagem...

... adiciona o valor para avançar para sua coordenada X...

... e, então, atualiza

```
Point p = MyPictureBox.Location;
```

Guy

Name
MyBet
Cash
MyRadioButton
MyLabel

UpdateLabels()
PlaceBet()
ClearBet()
Collect()

```
class Guy {
   public string Name; // O nome do cara
   public Bet MyBet; // Uma instância de Bet( ) que tem sua aposta
   public int Cash; // Quanto dinheiro ele tem

   // Os últimos dois campos são os controles no formulário da GUI dos caras
   public RadioButton MyRadioButton; // Meu RadioButton
   public Label MyLabel; // Minha Label

   public void UpdateLabels( ) {
     // Defina minha etiqueta para a descrição da minha aposta e a etiqueta em meu
     // meu botão de rádio para mostrar meu dinheiro ("João tem 43 reais")
   }

   public void ClearBet() { } // Redefina minha aposta para que seja zero

   public bool PlaceBet(int Amount, int Dog) {
     // Faça uma nova aposta e armazene-a no meu campo de aposta
     // Retorne true se o cara teve dinheiro suficiente para apostar
   }

   public void Collect(int Winner) { } // Cobre minha aposta se eu ganhei
}
```

Assim que você definir MyLabel para uma etiqueta no formulário, será capaz de mudar o texto da etiqueta usando MyLabel.Text. E o mesmo ocorre para MyRadioButton!

Quando você inicializar o objeto Guy, defina seu campo MyBet para null e chame seu método UpdateLabels() assim que terminar de inicializar.

Adicione seu código aqui.

Lembre-se que as apostas são representadas por instâncias de Bet.

A chave aqui é usar o objeto Bet... deixe que ele faça o trabalho.

Este é o objeto que a classe Guy usa para representar apostas na aplicação.

Bet

Amount
Dog
Bettor

GetDescription
PayOut

O inicializador de objeto para Bet apenas define a quantia, o cão e o apostador.

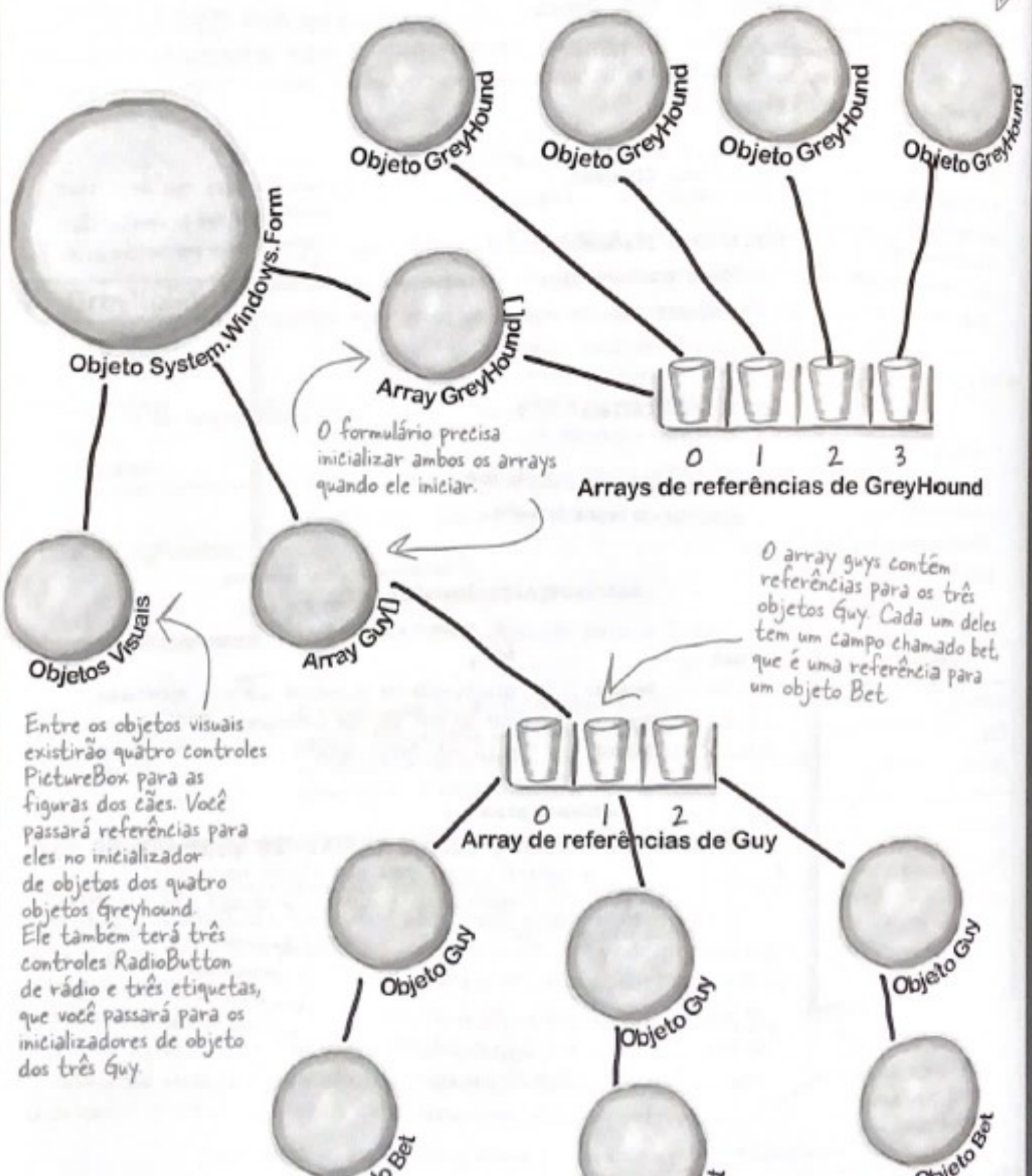
```
class Bet {
   public int Amount; // A quantidade de dinheiro que foi apostada
   public int Dog; // O número do cão em que apostamos
   public Guy Bettor; // O cara que fez a aposta

   public string GetDescription( ) {
     // Retorne uma string que diga quem fez a aposta, quanto
     // dinheiro foi apostado e em qual cão ("João apostou 8
     // no cão #4"). Se a quantidade for zero, a aposta não foi feita
     // ("João não apostou").
   }

   public int PayOut(int Winner) {
     // O parâmetro é o vencedor da corrida. Se o cão venceu,
     // retorne a quantia apostada. De outra forma, retorne um valor
     // negativo do valor apostado.
   }
}
```

Esta é uma tarefa de programação bem comum: montar uma string ou mensagem a partir de vários bits de dados individuais.

Dica: Você instanciará Bet no código de Guy, que usará a palavra-chave **this** para passar uma referência dele mesmo para o

Eis aqui a arquitetura de sua aplicação

Passe algum tempo examinando cuidadosamente a arquitetura. Ela parece bem complicada à primeira vista, mas não há nada aqui que você não saiba. Sua tarefa é implementar essa arquitetura você mesmo, começando com os arrays `GreyHound` e `Guy` no seu formulário principal.



Quando um cara faz uma aposta, ele cria um novo objeto Bet

Primeiro, o formulário informa a Guy #2 para fazer uma aposta de $ 7 no cão #3...

```
Guy[1].PlaceBet(7, 3)
```

...então, Guy # 2 cria uma nova instância de Bet, usando a palavra-chave this para informar ao objeto Bet que ele é o apostador...

```
MyBet = new Bet( )
{ Amount = 7, dog = 3, Bettor = this };
```


```
true
```

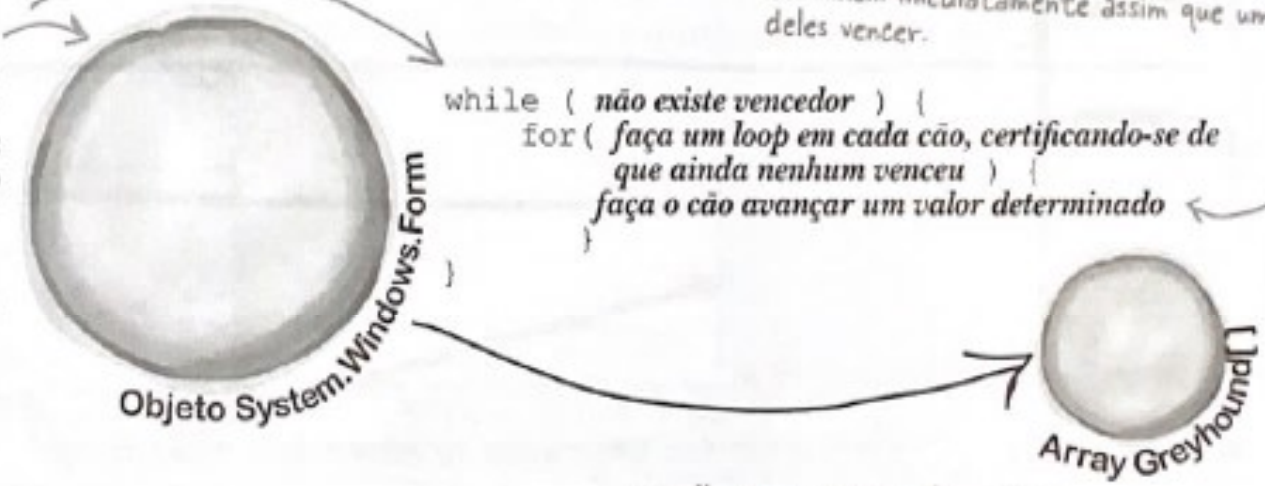
...e como Guy tem dinheiro suficiente para apostar, PlaceBet( ) retorna true.

O formulário diz aos cães para continuarem correndo até que um vença

O método Run( ) de cada cão checa se ele ganhou a corrida, para que os loops terminem imediatamente assim que um deles vencer.

Quando o usuário diz ao formulário para começar a corrida, ele inicia um loop para animar cada cão correndo na pista.

```
while ( não existe vencedor ) {
    for( faça um loop em cada cão, certificando-se de
         que ainda nenhum venceu ) {
        faça o cão avançar um valor determinado
    }
}
```



O objeto Bet descobre se deve ser feito algum pagamento

O salão de apostas no formulário diz para cada Guy qual cão venceu para que ele possa recolher qualquer vencedor a partir de sua aposta.

```
Guy[1].Collect(winningDog)
```

```
MyBet.PayOut(winningDog)
```

Guy vai ter seu dinheiro aumentado pelo resultado em Bet.Payout(). Portanto, se o cão ganhou, ele deverá retornar Amount. De outra forma, retornará -Amount.

```
if ( meu cão ganhou ) {
    return Amount;
} else {
    return -Amount;
}
```

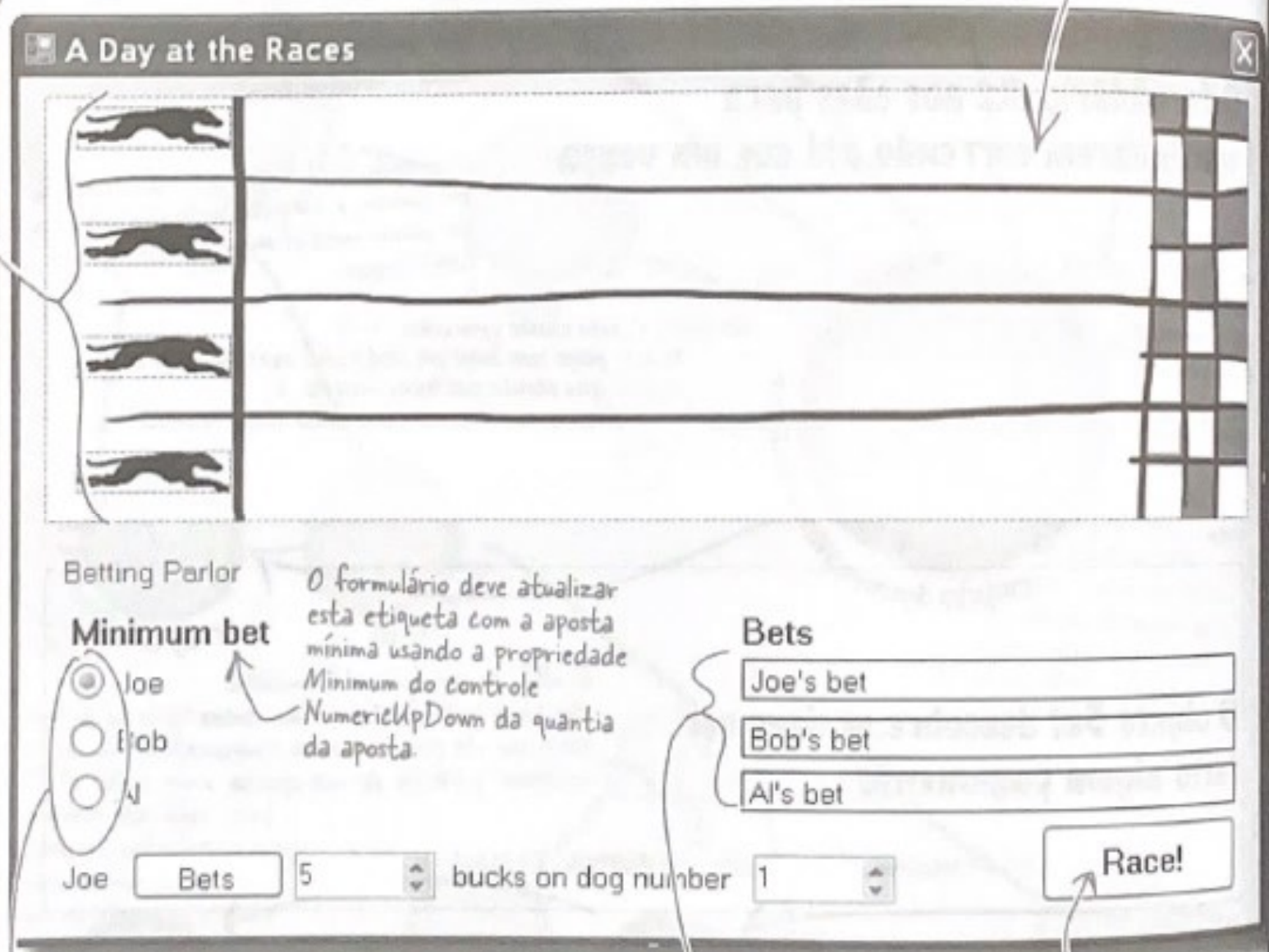
Aparência de sua GUI

A interface gráfica do usuário para a aplicação "Dia de Corrida" consiste num formulário dividido em duas seções. A parte de cima é a pista de corridas: um controle `PictureBox` para a pista e quatro outros para os cães. A metade inferior do formulário mostra o salão de apostas, onde três caras (Joe, Bob e Al) podem apostar no resultado da corrida.

Você usará a propriedade Length do controle PictureBox da pista para determinar o comprimento da pista no objeto Greyhound, que ele usará para descobrir quem ganhou a corrida.

Cada um dos quatro cães tem seu próprio controle PictureBox. Quando você inicializar cada um dos quatro objetos Greyhound, o campo MyPictureBox de cada um terá uma referência para um desses objetos. Você passará a referência (juntamente com o tamanho da pista de corrida e a posição inicial) para o inicializador de objeto de Greyhound.

Defina a propriedade SizeMode de cada PictureBox para Zoom.



O formulário deve atualizar esta etiqueta com a aposta mínima usando a propriedade Minimum do controle NumericUpDown da quantia da aposta.

Todos os três caras podem apostar na corrida, mas apenas uma única janela de apostas existe, assim, apenas um pode apostar de cada vez. Estes botões de rádio são usados para selecionar qual cara faz a aposta.

Quando um cara aposta, ele apaga qualquer aposta anterior. A aposta atual aparece nestes controles de label. Cada um deles tem AutoSize definido para False e BorderStyle definido para FixedSingle.

Uma vez que as apostas tenham sido feitas, clique neste botão para iniciar a corrida.

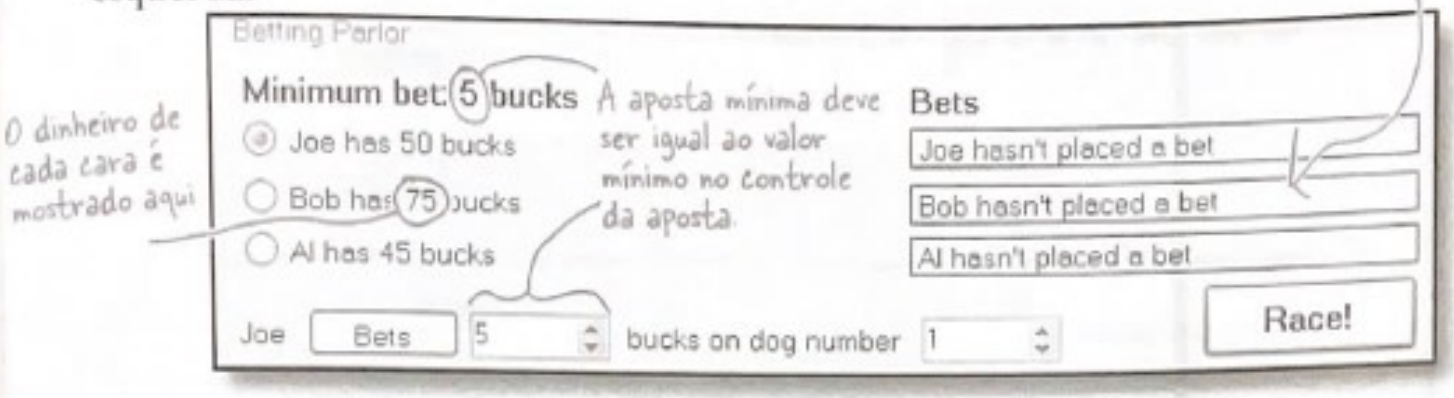
Apostando

Use os controles na caixa de agrupamento do Salão de Apostas para fazer a aposta de cada rapaz. Existem três estágios distintos aqui:

Quando um cara aposta, seu objeto Guy atualiza esta etiqueta usando a referência MyLabel. Ele também atualiza o dinheiro que ele tem usando sua referência MyRadioButton.

1 **Nenhuma aposta ainda foi feita**

Quando o programa inicia, ou se uma corrida terminou, nenhuma aposta estará registrada no Salão. Você verá o total de dinheiro de cada cara ao lado de seu nome, à esquerda.



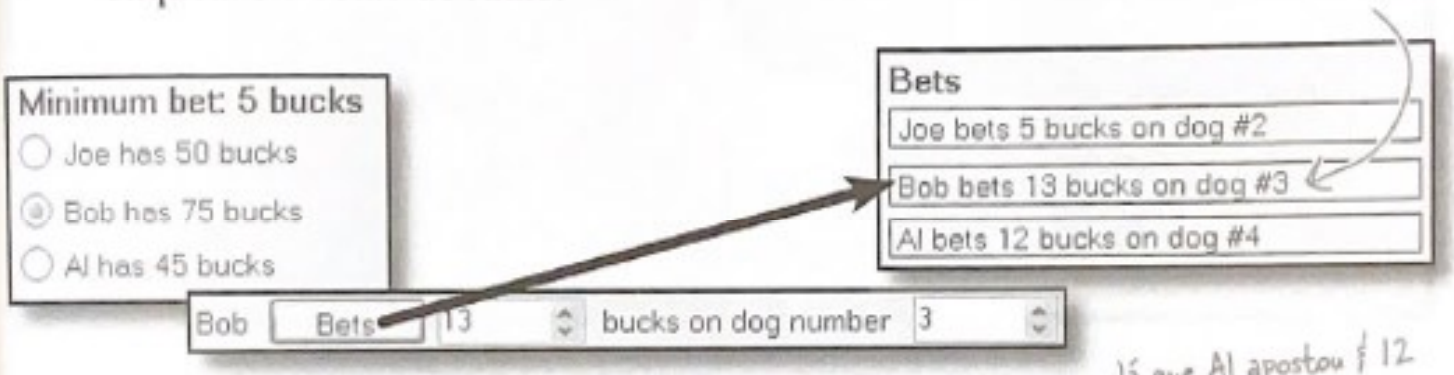
O dinheiro de cada cara é mostrado aqui

A aposta mínima deve ser igual ao valor mínimo no controle da aposta.

2 **Todos apostam**

Para apostar, selecione o botão de rádio de um cara, selecione uma quantidade e um cão, e clique no botão Bets (Apostas). Seu método `PlaceBet()` atualizará a etiqueta e o botão de rádio.

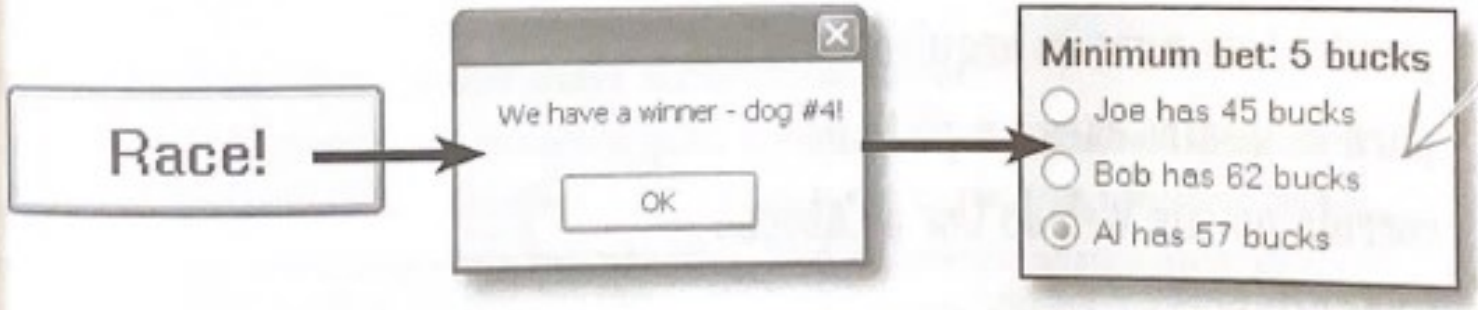
Já que Al apostou $ 12 no cão vencedor, seu dinheiro sobe em 12. Os outros dois caras perdem o dinheiro que apostaram.



3 **Depois da corrida, cada cara recebe seus lucros (ou paga os prejuízos!)**

Uma vez que a corrida tenha terminado e existe um vencedor, cada objeto Guy chama seu método **`Collect()`** e adiciona seu lucro ou prejuízo ao seu dinheiro.

Já que Al apostou $ 12 no cão vencedor, seu dinheiro sobe em 12. Os outros dois caras perdem o dinheiro que apostaram.

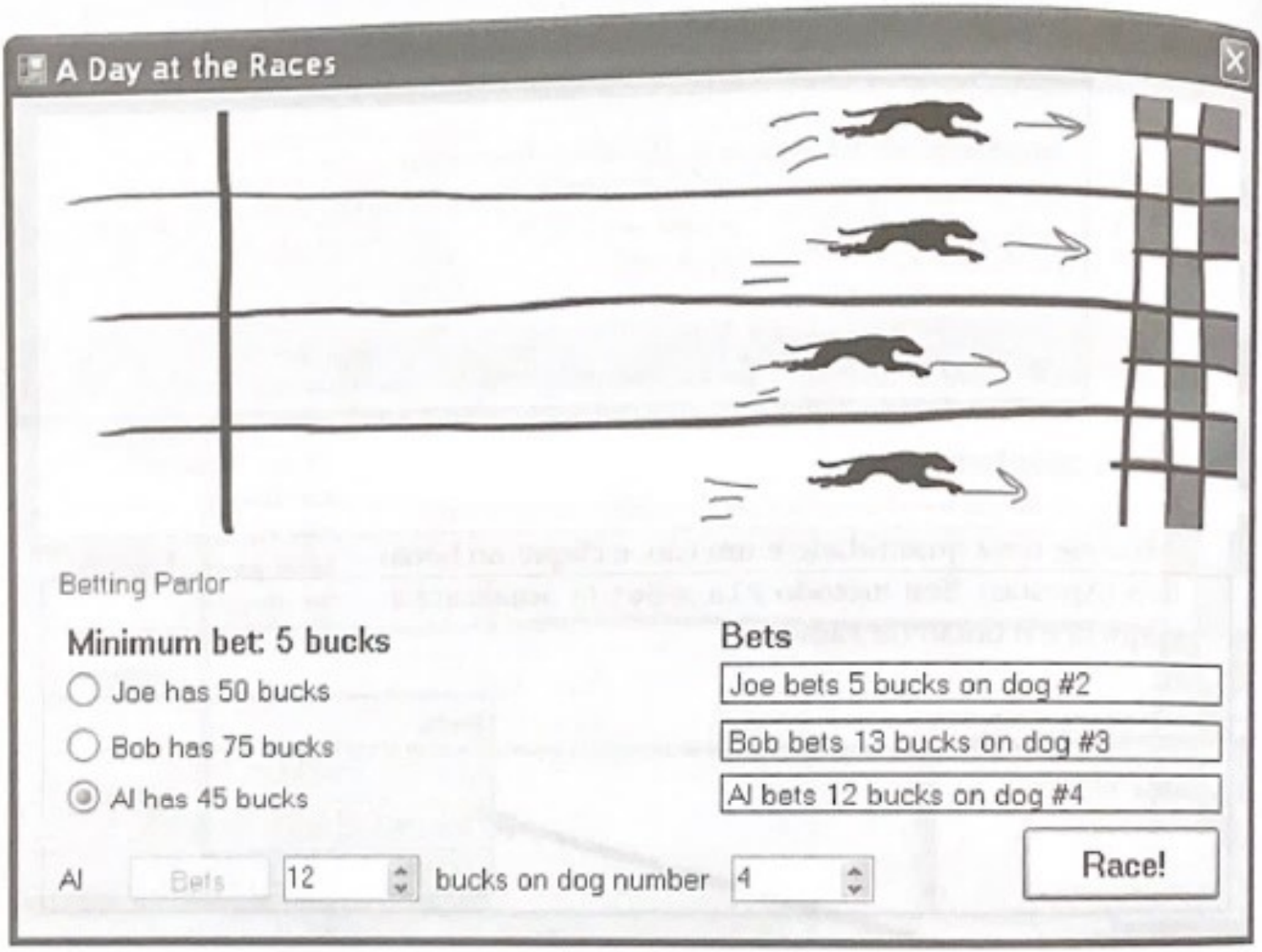


Certifique-se de que todos os objetos Greyhound compartilhem um objeto Random! Se cada cão criar sua própria instância nova de Random, você poderá ver um erro onde todos os cães geram a mesma sequência de números aleatórios.

O produto final

Você saberá que sua aplicação "Um Dia de Corrida" está pronta quando os caras puderem apostar e a corrida puder ser vista.

Durante a corrida, as imagens dos quatro cães correm na pista de corrida até que um deles vença.



Durante a corrida, nenhuma aposta pode ser feita... e certifique-se de que uma nova corrida não possa ser iniciada enquanto os cães estiverem correndo!