Nome:

Data:

Laboratório C# 1

Um dia de corridas

Este laboratório dá-lhe as especificações de um programa que você deve 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; você pode continuar lendo o livro e voltar ao laboratório mais tarde.

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

É sua responsabilidade terminar o trabalho. Você pode baixar o executável deste laboratório na nossa página na Internet... mas não lhe daremos o código da resposta.

Especificações: Desenvolva um simulador de pista de corridas

João, Beto e Alfredo 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

João, Beto e Alfredo querem apostar numa corrida de cachorros. João começa com R$ 50, Beto com R$ 75 e Alfredo com R$ 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 estarão 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 R$ 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 – eles não podem apostar se não tiverem dinheiro suficiente.

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

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

Apostando

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

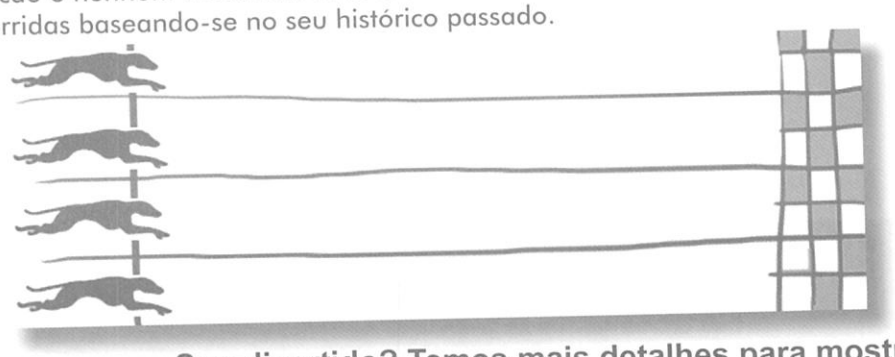
Todas as apostas: dobro-ou-nada
Aposta mínima: R$ 5
Até R$ 15 por cão
Vencedores: $$ adicionado
Perdedores: $$ removido

Digamos que um cara aposte R$ 10. No final da corrida, se o cachorro escolhido vence, ele tem mais R$ 10 (porque ele mantém os R$ 10 originais apostados e ganha mais R$ 10 por ter vencido). Se ele perder, seu total diminui R$ 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 características boas ou ruins para um cão e nenhum deles tem mais chance de vencer as próximas corridas baseando-se no seu histórico passado.

Se você quer desenvolver um sistema com vantagens e desvantagens para os cães, por favor faça! Será uma ótima oportunidade de praticar escrever código e ainda se divertir.



Soa divertido? Temos mais detalhes para mostrar... →

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 uma matriz de três objetos Guy (cara) para manter registro dos três caras e seus resultados e uma matriz de quatro objetos GreyHound (nome de raça de cão de corrida, literalmente "cão cinza") que disputarão as corridas. Além disso, cada instância de Guy deve ter seu próprio objeto Bet (aposta, ou apostar) que mantém registro das apostas e paga ou toma dinheiro no final de cada corrida.

Será preciso adicionar "using System.Windows.Forms" no topo das classes GreyHound e Guy.

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

Eis para você o esqueleto da classe que precisará desenvolver. Sua tarefa será completar os métodos.

Greyhound
StartingLocation
RacetrackLength
MyPictureBox
Location
MyRandom
Run()
TakeStartingPosition()

```
public class Greyhound {
  public int StartingPosition; // Onde a minha caixa de imagem inicia
  public int RacetrackLength; // O quanto a pista de corrida tem de comprimento
  public PictureBox MyPictureBox = null; // Meu objeto caixa de imagem
  public int Location = 0; // Minha posicao na pista
  public Random MyRandom; // Uma instancia de Random
  public bool Run() {
    // Mova-se para frente 1, 2, 3 ou 4 espacos aleatoriamente
    // Atualize a posicao da minha caixa de imagem no formulario
    // Retorna true se eu ganhei a corrida
  }
  public void TakeStartingPosition() {
    // Volte minha posicao para a linha de partida
  }
}
```

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

Vê como o diagrama de classe corresponde ao código?

O inicializador do objeto Greyhound é bem simples. Apenas se certifique de passar a referência para a PictureBox certa no formulário para cada um dos objetos.

Não quebre muito a cabeça com isso... algumas vezes só é preciso atribuir um valor pra uma variável e pronto.

Seu objeto pode controlar coisas em seu formulário...

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

A classe GreyHound mantém registro da posição na pista durante a corrida. Ela também atualiza a posição da caixa de imagem que representa o cão se movendo pela pista. Cada instância de GreyHound tem um campo chamado MyPictureBox com a referência ao controle de caixa de imagem no formulário que mostra a imagem de um cão. Suponha que a variável distance contenha a distância que um cão vai percorrer para frente. Então o seguinte código vai atualizar a posição de MyPictureBox ao adicionar distance ao valor de sua propriedade X:

```
Point p = MyPictureBox.Location;
p.X += distance;
MyPictureBox.Location = p;
```

Você recupera a posição atual da imagem...

... adiciona o valor de deslocamento para frente na sua coordenada X...

... e então atualiza a posição da caixa de imagem no formulário.

Guy
Name
MyBet
Cash
MyRadioButton
MyLabel
UpdateLabels()
PlaceBet()
ClearBet()
Collect()

```
public class Guy {
  public string Name; // O nome do cara
  public Bet MyBet; // Uma instancia de Bet() que mantem as apostas
  public int Cash; // Quanto dinheiro resta
  // Os ultimos dois campos sao os controles no formulario da GUI dos caras
  public RadioButton MyRadioButton; // Meu botao de radio
  public Label MyLabel; // Meu rotulo
  public void UpdateLabels() {
    // Atribua ao meu rotulo a descricao da minha aposta, e ao rotulo do
    // meu botao de radio o meu dinheiro ("Joao tem 43 reais")
  }

  public void ClearBet() { } // Reinicialize minha aposta para que ela zere
  public bool PlaceBet(int Amount, int Dog) {
    // Crie uma nova aposta e armazene-a no meu campo bet
    // Retorne verdadeiro se o cara tem dinheiro suficiente para apostar
  }
  public void Collect(int Winner) { } // Cobre minha aposta se eu ganhei
}
```

Adicione seu código aqui.

Quando inicializar o objeto Guy, certifique-se de atribuir o campo MyBet para null, e chame o método UpdateLabels() (atualiza rótulos) assim que terminar de inicializar.

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

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

Este é o objeto que a classe Guy usa para representar apostas no aplicativo.

O inicializador de objeto para Bet atribui o valor, o cão e o apostador.

Bet
Amount
Dog
Bettor
GetDescription
PayOut

```
public class Bet {
  public int Amount; // A quantidade de dinheiro que foi apostada
  public int Dog; // O numero do cao em que apostamos
  public Guy Bettor; // O cara que fez a aposta
  public string GetDescription() {
    // Retorna uma sequencia de caracteres que diz quem fez a aposta, quanto
    // dinheiro foi apostado e em qual cao ("Joao apostou 8
    // no cao 4"). Se a quantidade for zero, a aposta nao foi feita
    // ("Joao nao apostou").

  }
  public int PayOut(int Winner) {
    // O parametro deve receber o vencedor da corrida. Se o cao venceu,
    // retorna a quantidade apostada. De outra forma, retorne um valor
    // negativo correspondente ao valor apostado
  }
}
```

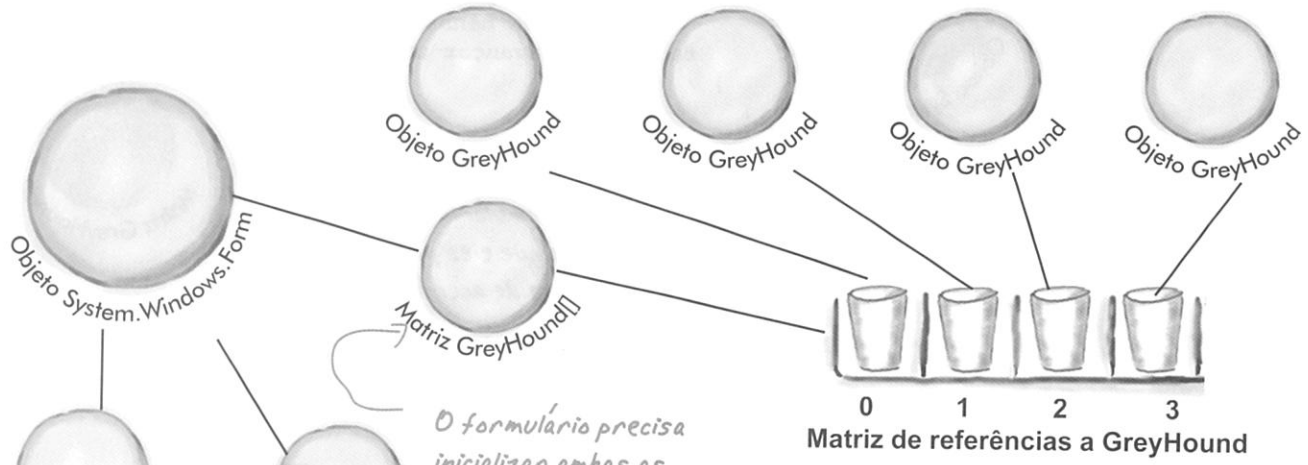
Esta é uma tarefa de programação bem comum: montar uma seqüência de caracteres usando muitos dados de diferentes origens.

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

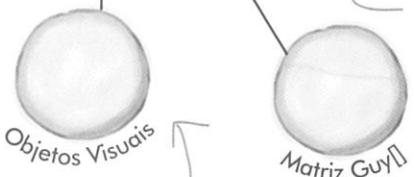
Eis aqui a estrutura de seu aplicativo

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

A matriz dogs contém quatro referências, cada uma apontando para uma instância em separado da classe GreyHound.

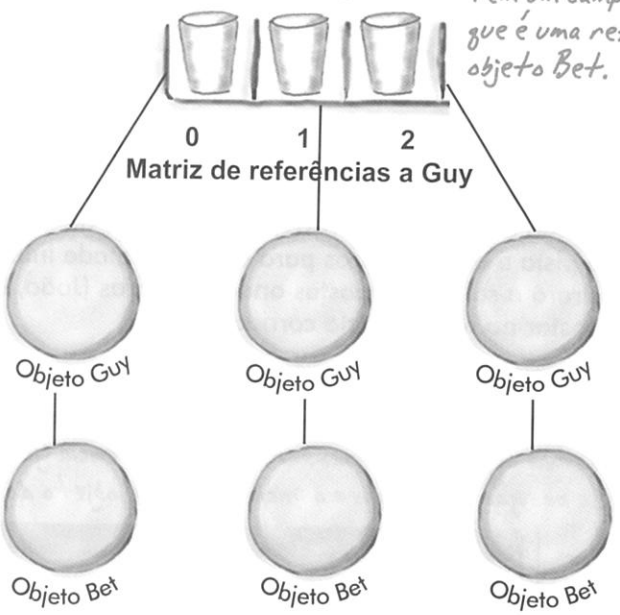


O formulário precisa inicializar ambas as matrizes quando ele iniciar.



Entre os objetos visuais existirão quatro controles de caixa de imagem para as figuras dos cães. Você passará referências para eles no inicializador de objetos dos quatro GreyHounds. Também existem três botões de rádio e três rótulos, que serão passados aos inicializadores de objeto dos três Guy.

A matriz guys conterá referências aos três objetos Guy. Cada um deles tem um campo chamado bet, que é uma referência a um objeto Bet.



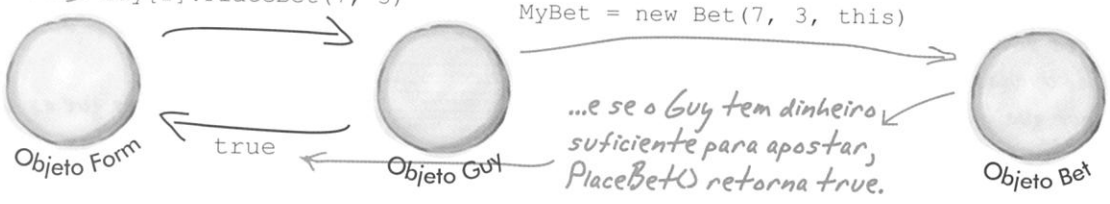
Quando um cara faz uma aposta, ele cria um novo objeto Bet

Primeiro o formulário diz, por exemplo, ao Guy #2 para apostar R$ 7 no cão 3...

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

...então o Guy 2 cria uma nova instância de Bet, usando a palavra chave this para dizer a ele quem é o apostador...

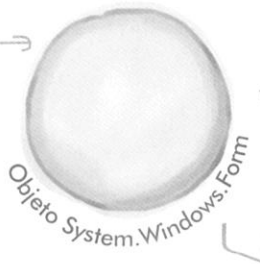
```
MyBet = new Bet(7, 3, this)
```



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

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

O método Run() de cada cão checa se ele ganhou a corrida, para que o laço termine imediatamente assim que um deles o fizer.




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

O objeto Bet determina se deve ser feito algum pagamento

O salão de apostas no formulário diz para cada Guy qual cão venceu para que eles possam recolher qualquer valor decorrente de acertar as apostas.

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

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

O Guy vai ter seu dinheiro aumentado pelo resultado em BetPayout(). Se o cão em que apostou ganhou, ele deve retornar Quantidade. De outra forma, -Quantidade.

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

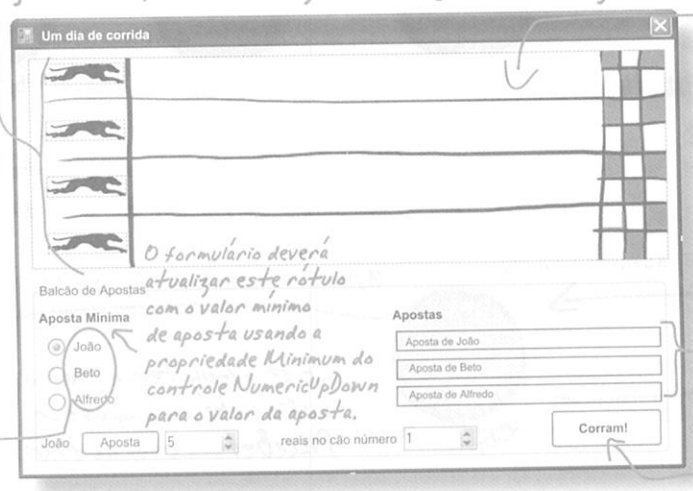
Aspecto de sua GUI

A interface gráfica de usuário (GUI, Graphical User Interface) para o aplicativo "Dia de Corrida" consistirá num formulário dividido em duas seções. A parte de cima é a pista de corridas. Um controle de `caixa de imagem` para a pista e quatro outros para cães. A metade inferior do formulário mostrará o salão de apostas onde três caras (João, Beto e Alfredo) podem apostar no resultado da corrida.

Usaremos a propriedade Length do controle de caixa de imagem da pista para determinar o comprimento da pista de corrida para os objetos GreyHound, que eles usam para descobrir quem ganhou a corrida.

Cada um dos quatro cães tem seu próprio controle de caixa de imagem. Quando você inicializa cada um dos quatro objetos GreyHound, o campo MyPictureBox de cada um terá uma referência a um destes objetos. Você passará essa referência (juntamente com o tamanho da pista de corrida e a posição inicial) para o inicializador de objeto dos GreyHound.

Todos os três caras podem apostar na mesma corrida, mas apenas uma única janela de apostas existe. Assim, eles têm que apostar um de cada vez. Estes botões de rádio são usados para selecionar qual dos caras está fazendo a aposta.



Quando um cara aposta, ele apaga qualquer aposta anterior. A aposta atual aparece nestes controles de rótulo. Cada um deles deverá ter AutoSize como False e BorderStyle como FixedSingle.

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

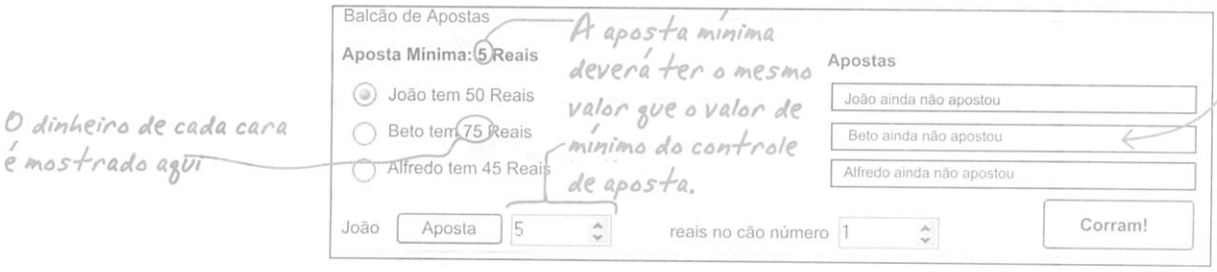
Você pode baixar os arquivos gráficos de `www.altabooks.com.br`

Apostando

Use os controles na caixa de agrupamento do Salão de Apostas para fazer com que cada um dos caras aposte. Existem três estágios distintos aqui:

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.

Quando um cara aposta, seu objeto Guy atualiza este rótulo usando a referência MyLabel. Ele também atualiza o dinheiro que se tem usando a referência MyRadioButton.

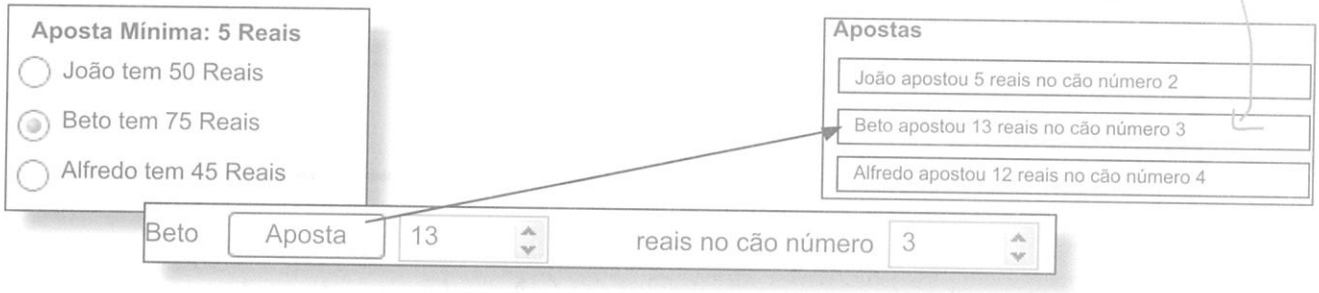


O dinheiro de cada cara é mostrado aqui

A aposta mínima deverá ter o mesmo valor que o valor de mínimo do controle de aposta.

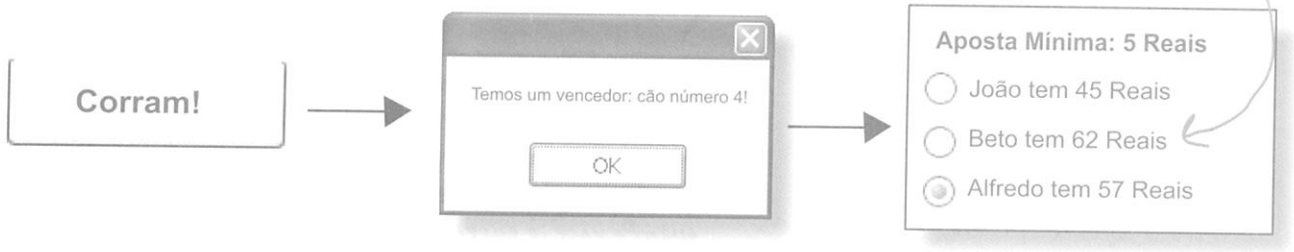
2. **Todos apostam**
   Para apostar, selecione o botão do rádio de um cara qualquer, selecione uma quantidade e um cão e clique o botão "aposta". O método `PlaceBet()` vai atualizar o rótulo e o botão de rádio.

Uma vez que Beto tenha apostado, seu objeto Guy atualiza este rotulo e o texto do botão de rádio.



3. **Depois da corrida, cada cara recebe seus lucros (ou paga os prejuízos!)**
   Uma vez que a corrida tenha se completado e exista um vencedor, cada objeto `Guy` chama seu método `Collet()` e adiciona seu lucro ou prejuízo ao seu total em dinheiro.

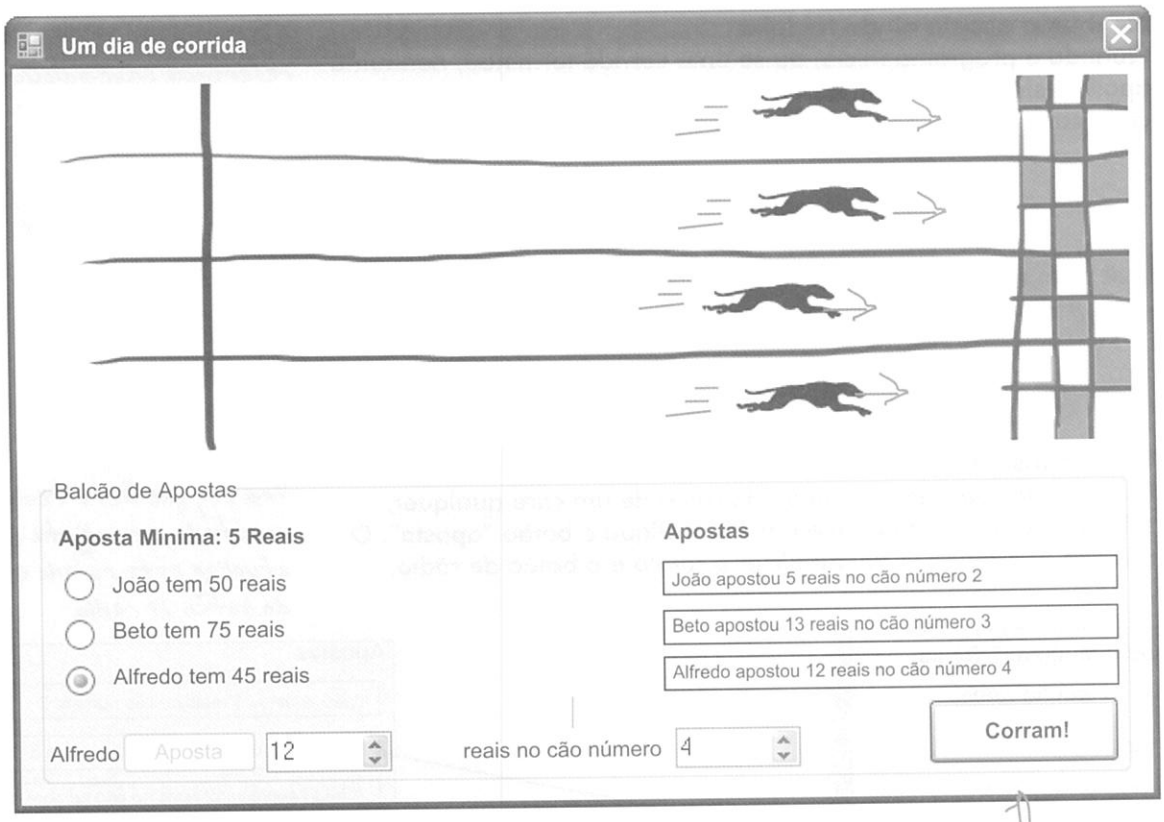
Já que Alfredo apostou 12 Reais no cão vencedor, seu total sobe 12. Os outros dois caras perdem o dinheiro que apostaram.



O Produto Final

Você saberá que seu aplicativo "Um Dia de Corrida" estará pronto quando seus caras puderem apostar e a corrida puder ser vista.

Durante a corrida, as imagens dos quatro cães correm pela 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 em movimento!

Você pode baixar um executável pronto, bem como os arquivos gráficos para os quatro cães e a pista de corrida na página de Internet do Use a Cabeça:
`www.altabooks.com.br`

Mas você não vai encontrar o código fonte! Na vida real, você não terá uma solução pronta para seus problemas de programação. Eis a sua chance de testar realmente seu conhecimento em C# e ver o quanto você já aprendeu até agora!