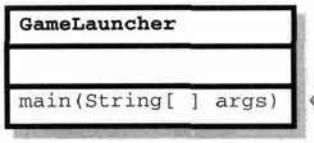
examinamos o uso de um método main( ) em uma classe TestDrive separada para criar e testar os métodos e variáveis de outra classe. No Capítulo 6 examinaremos o uso de uma classe com um método main( ) para iniciar um aplicativo Java *real* (criando objetos e, em seguida, deixando-os livres para interagir com outros objetos, etc.)

No entanto, como uma prévia de como um aplicativo Java real pode se comportar, aqui está um pequeno exemplo. Já que ainda estamos nos estágios iniciais do aprendizado de Java, trabalharemos com um pequeno kit de ferramentas , portanto, você achará esse programa um pouco complicado e ineficiente. Talvez pense no que poderia fazer para aperfeiçoá-lo, e em capítulos posteriores é exatamente isso que faremos. Não se preocupe se parte do código for confusa; o ponto-chave desse exemplo é que os objetos se comunicam entre si.

O jogo de adivinhação

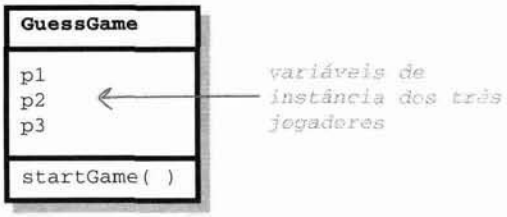
Resumo:

O jogo de adivinhação envolve um objeto 'game' e três objetos 'player'. O jogo gera um número aleatório entre 0 e 9 e os três objetos player tentam adivinhá-lo. (Não dissemos que seria um jogo *divertido.*)



Classes:

GuessGame.class Player.class GameLauncher.class

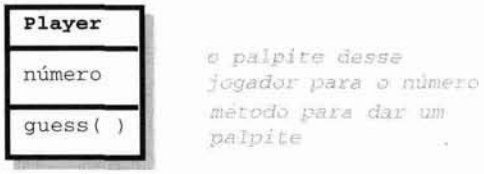


A lógica:

1) É na classe GameLauncher que o aplicativo é iniciado; ela tem o método main( ).

2) No método main( ), um objeto GuessGame é criado e seu método *startGame( ) é chamado.*

3) É no método startGame( ) do objeto GuessGame que o jogo inteiro se desenrola. Ele cria três jogadores e, em seguida, "pensa" em um número aleatório (aquele que os jogadores têm que adivinhar). Depois solicita a cada jogador que adivinhe, verifica o resultado e exibe informações sobre o(s) jogador(es) vencedor(es) ou pede que adivinhem novamente.




```
public class GuessGame {
   Player p1;
   Player p2;
   Player p3;

   public void startGame() {
      p1 = new Player();
      p2 = new Player();
      p3 = new Player();

      int guessp1 = 0;
      int guessp2 = 0;
      int guessp3 = 0;

      boolean p1isRight = false;
      boolean p2isRight = false;
      boolean p3isRight = false;

      int targetNumber = (int) (Math.random() * 10);
      System.out.println("Estou pensando em um número entre 0 e 9...");

      while(true) {
         System.out.println("O número a adivinhar é " + targetNumber);

         p1.guess();
         p2.guess();
         p3.guess();
```

GuessGame tem três variáveis de instância para os três objetos Player

cria três objetos Player e atribui a eles as três variáveis de instância Player

declara três variáveis para armazenar os três palpites que os jogadores fornecerão

declara três variáveis para armazenar um valor verdadeiro ou falso baseado na resposta do jogador

cria um número "alvo" que os jogadores terão que adivinhar

chama o método guess( ) de cada jogador

```
            guessp1 = p1.number;
            System.out.println("O jogador um forneceu o palpite " + guessp1);
            guessp2 = p2.number;
            System.out.println("O jogador dois forneceu o palpite " + guessp2);
            guessp3 = p3.number;
            System.out.println("O jogador três forneceu o palpite " + guessp3);

            if (guessp1 == targetNumber) {
               p1isRight = true;
            }
            if (guessp2 == targetNumber) {
               p2isRight = true;
            }
            if (guessp3 == targetNumber) {
               p3isRight = true;
            }

            if (p1isRight || p2isRight || p3isRight) {

               System.out.println("Temos um vencedor!");
               System.out.println("O jogador um acertou? " + p1isRight);
               System.out.println("O jogador dois acertou? " + p2isRight);
               System.out.println("O jogador três acertou? " + p3isRight);
               System.out.println("Fim do jogo.");
               break; // fim do jogo, portanto saia do loop
            } else {
               // devemos continuar porque ninguém acertou!
               System.out.println("Os jogadores terão que tentar novamente.");
            } // fim de if/else
         } // fim do loop
      } // fim do método
   } // fim da classe
```

obtem o palpite de cada jogador (o resultado da execução de seu método guess( )) acessando a variável numérica de cada um

verifica o palpite de cada jogador para ver se é igual ao número-alvo. Se um jogador acertar, sua variável será configurada com true (lembre-se de que configuramos false como o padrão)

se o jogador um OU o jogador dois OU o jogador três acertar... (O operador || significa OU)

caso contrário, fique no loop e peça aos jogadores outro palpite.

Executando o jogo de adivinhação

```
public class Player {
   int number = 0;  // onde entra o palpite

   public void guess() {
      number = (int) (Math.random() * 10);
      System.out.println("Estou pensando em " + number);
   }
}

public class GameLauncher {
   public static void main (String[] args) {
      GuessGame game = new GuessGame();
      game.startGame();
   }
}
```

Saída (será diferente a cada vez que você executar)

```
File Edit Window Help Explode
%java GameLauncher
Estou pensando em um número entre 0 e 9...
O número a adivinhar é 7
Estou pensando em 1
Estou pensando em 9
Estou pensando em 9
O jogador um forneceu o palpite 1
O jogador dois forneceu o palpite 9
O jogador três forneceu o palpite 9
Os jogadores terão que tentar novamente.
O número a adivinhar é 7
Estou pensando em 3
Estou pensando em 0
Estou pensando em 9
O jogador um forneceu o palpite 3
O jogador dois forneceu o palpite 0
O jogador três forneceu o palpite 9
Os jogadores terão que tentar novamente.
O número a adivinhar é 7
Estou pensando em 7
Estou pensando em 5
Estou pensando em 0
O jogador um forneceu o palpite 7
O jogador dois forneceu o palpite 5
O jogador três forneceu o palpite 0
Temos um vencedor!
O jogador um acertou? verdadeiro
O jogador dois acertou? falso
O jogador três acertou? falso
Fim do jogo.
```

O Java coleta o lixo

Sempre que um objeto é criado em Java, ele vai para uma área da memória conhecida como **Heap**. Todos os objetos - independentemente de quando, onde ou como sejam criados - residem no heap. Mas não se trata simplesmente de qualquer memória heap como as antigas; na verdade a memória heap Java se chama **Pilha de Lixo Coletável**. Quando você criar um objeto, a Java alocará espaço na memória heap de acordo com quanto esse objeto específico vai precisar. Um objeto com, digamos, 15 variáveis de instância, provavelmente precisará de mais espaço do que um objeto com apenas duas variáveis de instância. Mas o que acontecerá quando você precisar reclamar esse espaço? Como você tirará um objeto do heap quando não precisar mais dele? A Java gerenciará essa memória para você! Quando a JVM 'perceber' que um objeto pode nunca mais ser usado, ele se tornará *qualificado para a coleta de lixo*. E se você estiver ficando com pouco espaço na memória, o Coletor de Lixo será executado, eliminará os objetos inalcançáveis e liberará espaço, para que esse possa ser reutilizado. Em capítulos posteriores você aprenderá mais sobre como isso funciona.