Associação muitos para um, esse tipo de relação é indicado na própria classe, aquela que possui uma lista é a que aceita muitos valores da outra classe, já a classe que não tem lista só pode receber um valor da outra classe, caracterizando uma associação muitos para um (*..1). Como fazer isso? Veja...

Jogador.java

```java
public class Jogador {
    private String nome;
    private Time time;

    public Jogador(String nome) {
        this.setNome(nome);
    }

    public void imprime() {
        System.out.println(this.getNome());
        if (time != null) {
            System.out.println(time.getNome());
        }
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Time getTime() {
        return time;
    }

    public void setTime(Time time) {
        this.time = time;
    }
}
```

Time.java

```java
public class Time {
    private String nome;
    private Jogador[] jogadores;

    public Time(String nome) {
        this.setNome(nome);
    }

    public Time(String nome, Jogador[] jogadores) {
        this.setNome(nome);
        this.setJogadores(jogadores);
    }

    public void imprime() {
        System.out.println(this.getNome());
        if (jogadores == null) return;
        for (Jogador jogador : jogadores) {
            System.out.println(jogador.getNome());
        }
    }

    public String getNome() {return nome;}

    public void setNome(String nome) {this.nome = nome;}

    public Jogador[] getJogadores() {return jogadores;}

    public void setJogadores(Jogador[] jogadores) {
        this.jogadores = jogadores;
    }
}
```

JogadorTeste.java

```java
public class JogadorTeste {
    public static void main(String[] args) {
        // cria instancia de jogador e time
        Jogador jogador1 = new Jogador("Cafu");
        Jogador jogador2 = new Jogador("Pelé");
        Time time = new Time("Brasil");

        // cria lista de objetos do tipo Jogador
        Jogador[] jogadores = {jogador1, jogador2};

        // associa time ao jogador
        jogador1.setTime(time);
        jogador2.setTime(time);

        // associa lista de jogadores ao time
        time.setJogadores(jogadores);

        System.out.println("--- Jogador ---");
        jogador1.imprime();
        jogador2.imprime();

        System.out.println("--- Time ---");
        time.imprime();
    }
}
```

Como os passos dessa classe JogadorTeste.java funciona na memória do computador?

Isso pode ajudar a compreender o código como um todo, além de sua associação.

Vamos analisar?

1° PASSO

Cria uma instância do tipo Jogador chamado **jogador1**. Esse ato funciona da seguinte forma na memória do computador:

- Uma referência chamada jogador1 é criada na memória Stack, o primeiro da pilha.
- Essa referência aponta para o objeto criado na memória Heap, através do operador **new** e define valor para o atributo nome.
- O objeto referenciado por jogador1 tem um nome definido através do construtor, chama-se “Cafu”.

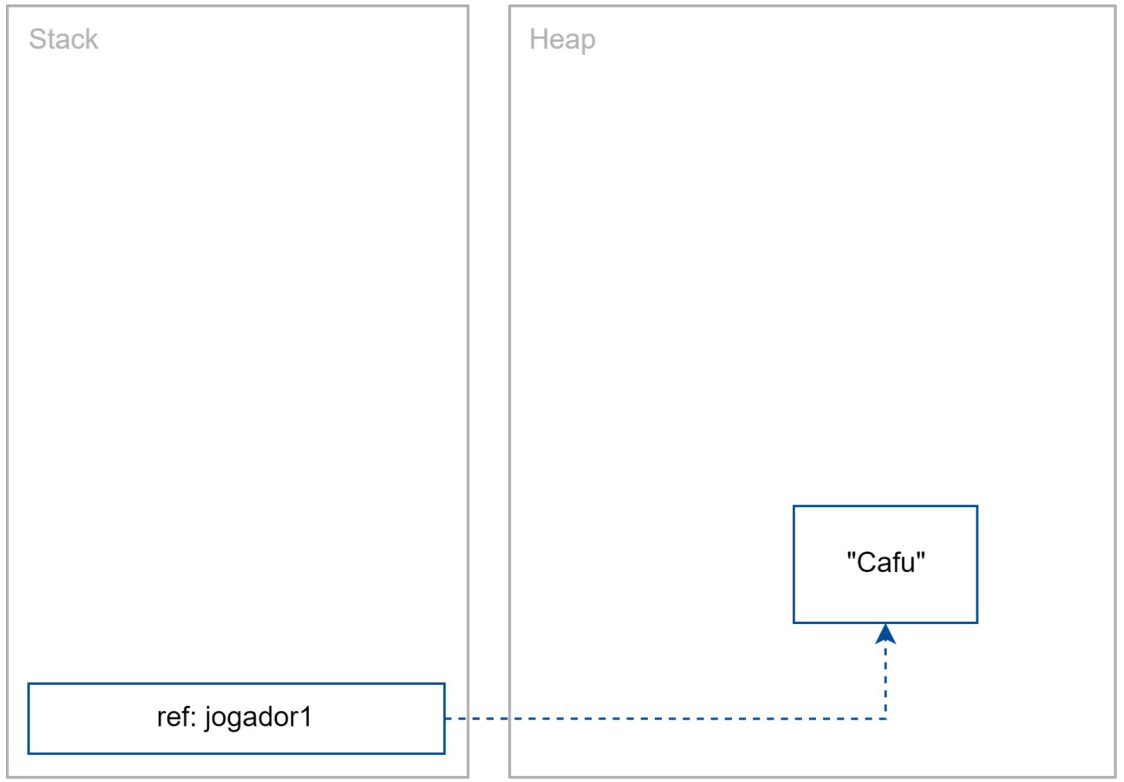
```
public class JogadorTeste03 {
   public static void main(String[] args) {

      Jogador jogador1 = new Jogador("Cafu");
      Jogador jogador2 = new Jogador("Pelé");
      Time time = new Time("Brasil");

      Jogador[] jogadores = {jogador1, jogador2};

      jogador1.setTime(time);
      jogador2.setTime(time);

      time.setJogadores(jogadores);
   }
}
```



2° PASSO

Cria uma instância do tipo Jogador chamado **jogador2**. Esse ato funciona praticamente igual ao da instância anterior:

- Uma referência para o objeto jogador2 é criada na memória Stack, o segundo da pilha.
- Essa referência aponta para o objeto criado na memória Heap, através do operador **new** e define valor para o atributo nome.
- O objeto referenciado por jogador2 tem um nome definido através do construtor, chama-se “Pelé”.

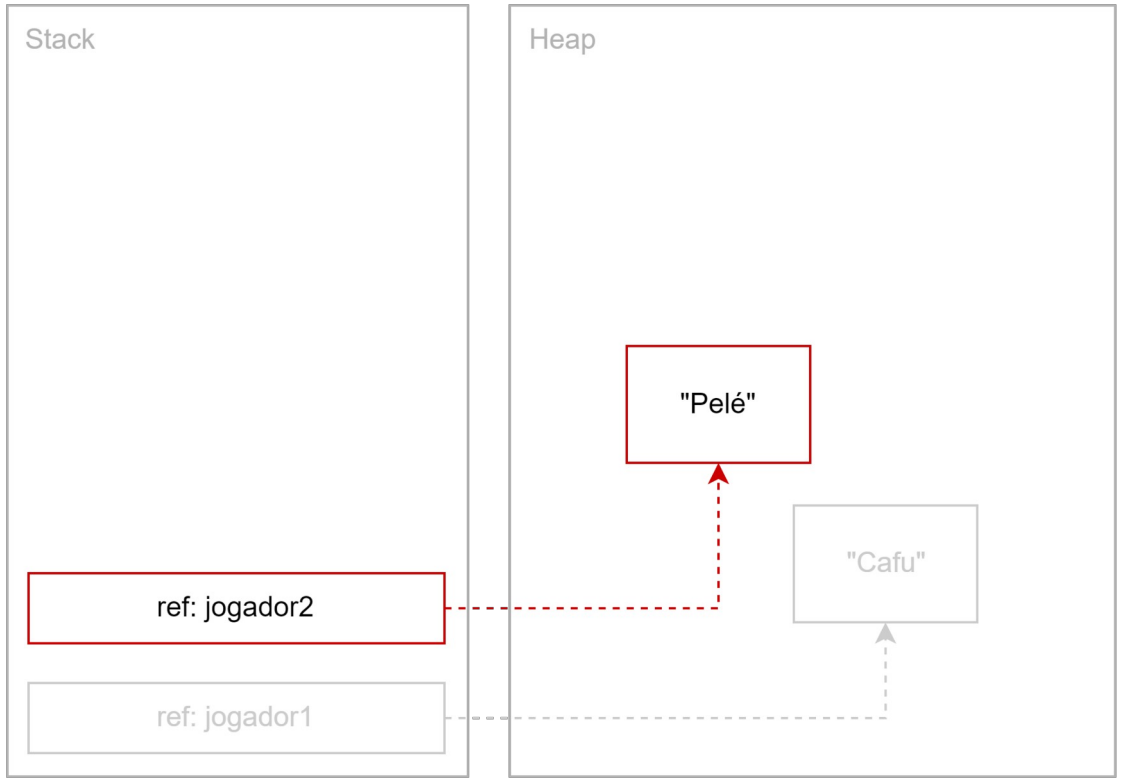
```
public class JogadorTeste03 {
    public static void main(String[] args) {

        Jogador jogador1 = new Jogador("Cafu");
        Jogador jogador2 = new Jogador("Pelé");
        Time time = new Time("Brasil");

        Jogador[] jogadores = {jogador1, jogador2};

        jogador1.setTime(time);
        jogador2.setTime(time);

        time.setJogadores(jogadores);
    }
}
```



3° PASSO

Cria uma instância do tipo Time chamado **time**. Esse ato funciona praticamente igual as instâncias anteriores:

- Uma referência para o objeto time é criada na memória Stack, o terceiro da pilha.
- Essa referência aponta para o objeto criado na memória Heap, através do operador **new** e atribui valor para o atributo nome.
- O objeto referenciado por time tem o atributo nome definido através do construtor, chama-se “Brasil”.

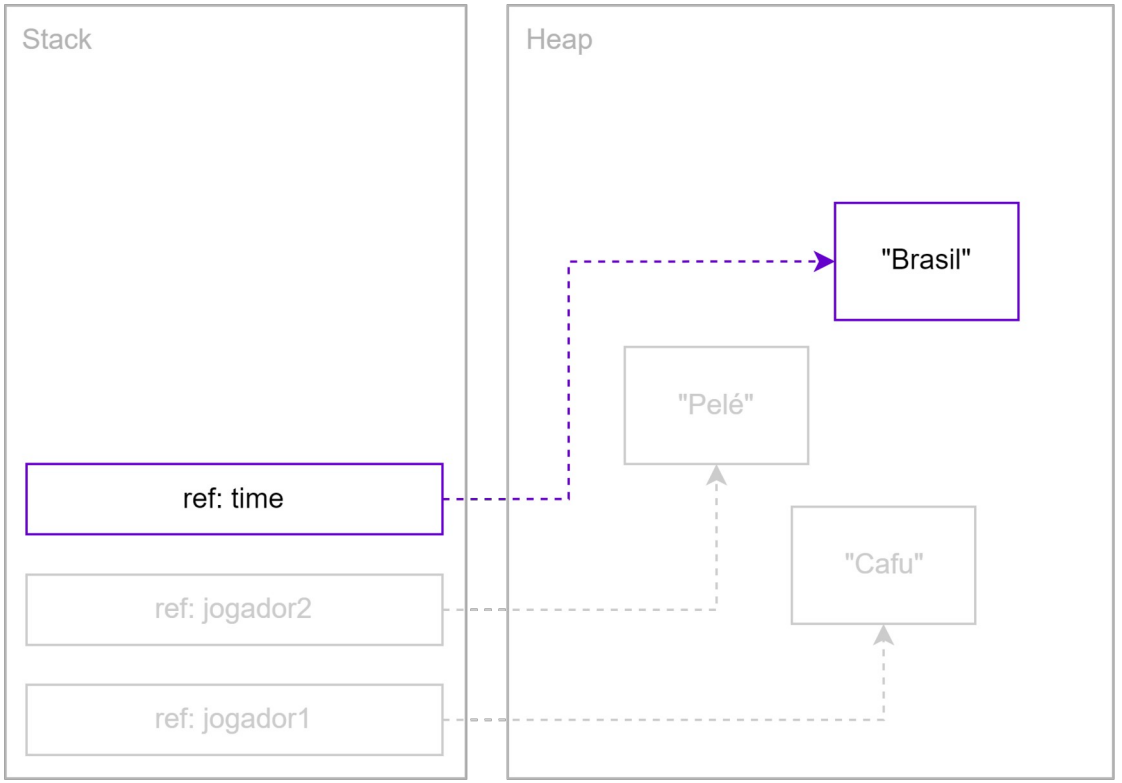
```
public class JogadorTeste03 {
    public static void main(String[] args) {

        Jogador jogador1 = new Jogador("Cafu");
        Jogador jogador2 = new Jogador("Pelé");
        Time time = new Time("Brasil");

        Jogador[] jogadores = {jogador1, jogador2};

        jogador1.setTime(time);
        jogador2.setTime(time);

        time.setJogadores(jogadores);
    }
}
```



4° PASSO

Cria uma lista de objetos do tipo Jogador chamado **jogadores**. Essa lista tem duas posições e funciona da seguinte forma:

- Uma lista de duas posições é criada na memória Stack, este é o quarto espaço ocupado na pilha.
- Cada posição possui um objeto que, por sua vez, possui referências que apontam para os objetos criados na memória Heap.
- As referências que apontam para os objetos estão nas posições da lista, o nome da lista não tem referência para nenhum objeto.

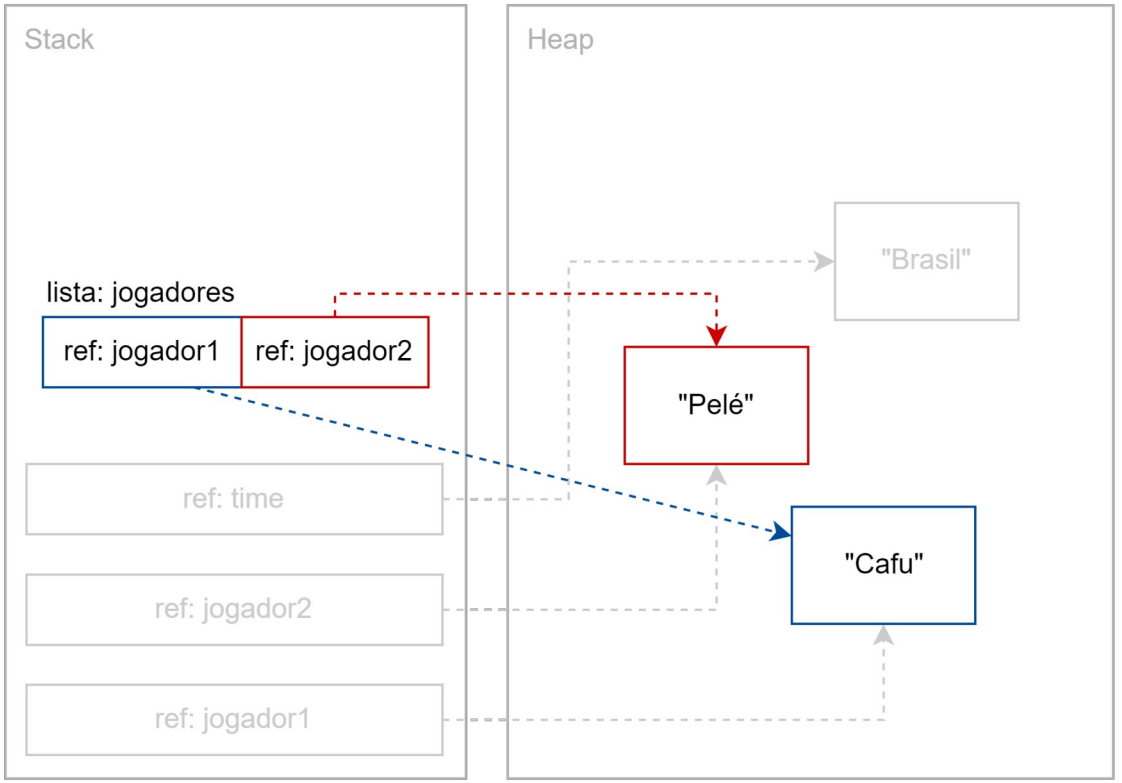
```
public class JogadorTeste03 {
    public static void main(String[] args) {

        Jogador jogador1 = new Jogador("Cafu");
        Jogador jogador2 = new Jogador("Pelé");
        Time time = new Time("Brasil");

        Jogador[] jogadores = {jogador1, jogador2};

        jogador1.setTime(time);
        jogador2.setTime(time);

        time.setJogadores(jogadores);
    }
}
```



5° PASSO

Altera o valor do atributo **time** através da referência **jogador1**. Essa alteração funciona da seguinte forma:

- Através da referência **jogador1** na Stack o método **setTime** é acessado para alterar o atributo time, definindo valor para ele.
- Esse método recebe um objeto, esse objeto possui dois atributos, mas apenas o nome “Brasil” foi definido no ato da criação.
- O objeto referenciado pelo jogador1 tem o atributo time definido com as informações da instância referenciada por time.

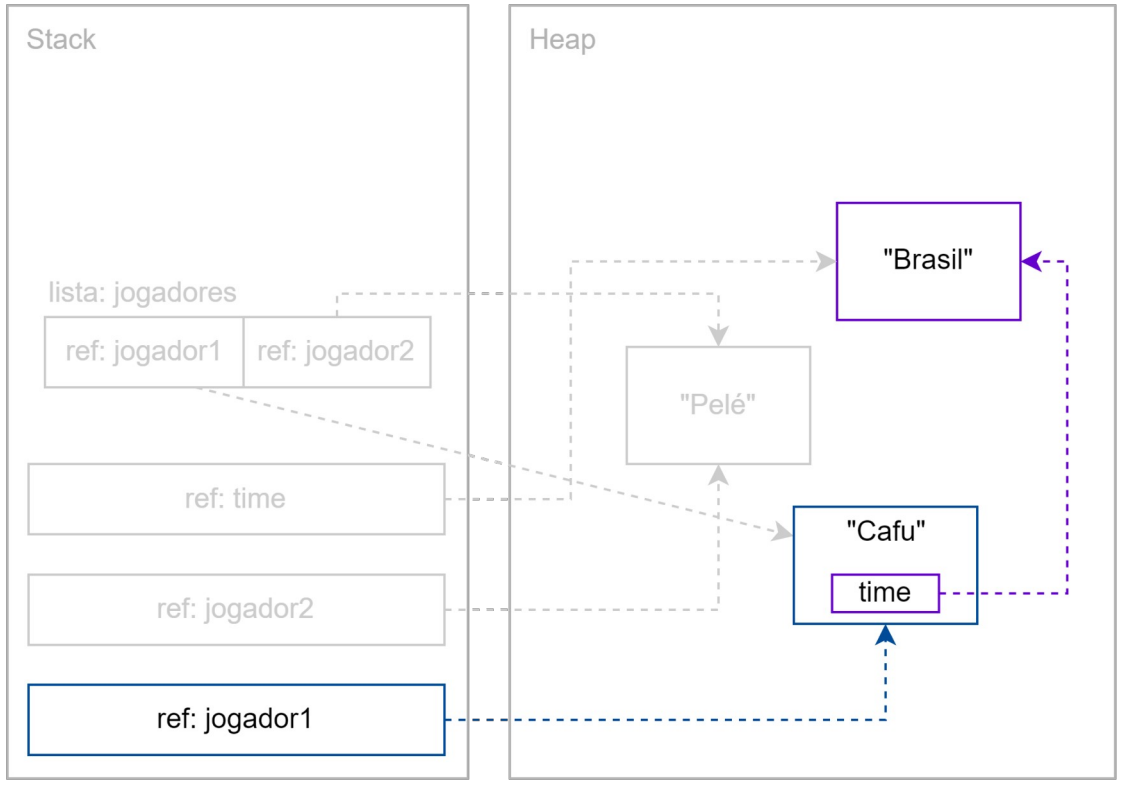
```
public class JogadorTeste03 {
   public static void main(String[] args) {

      Jogador jogador1 = new Jogador("Cafu");
      Jogador jogador2 = new Jogador("Pelé");
      Time time = new Time("Brasil");

      Jogador[] jogadores = {jogador1, jogador2};

      jogador1.setTime(time);
      jogador2.setTime(time);

      time.setJogadores(jogadores);
   }
}
```



6° PASSO

Altera o valor do atributo **time** através da referência **jogador2**. Essa alteração funciona igual ao passo 5, da seguinte forma:

- Através da referência **jogador2** na Stack o método **setTime** é acessado para alterar o atributo time, atribuindo valor a ele.
- Esse método recebe um objeto, sesse objeto possui dois atributos, mas apenas o nome “Brasil” foi definido no ato da criação.
- O objeto referenciado pelo jogador2 terá o atributo time definido com as informações da instância referenciada por time.

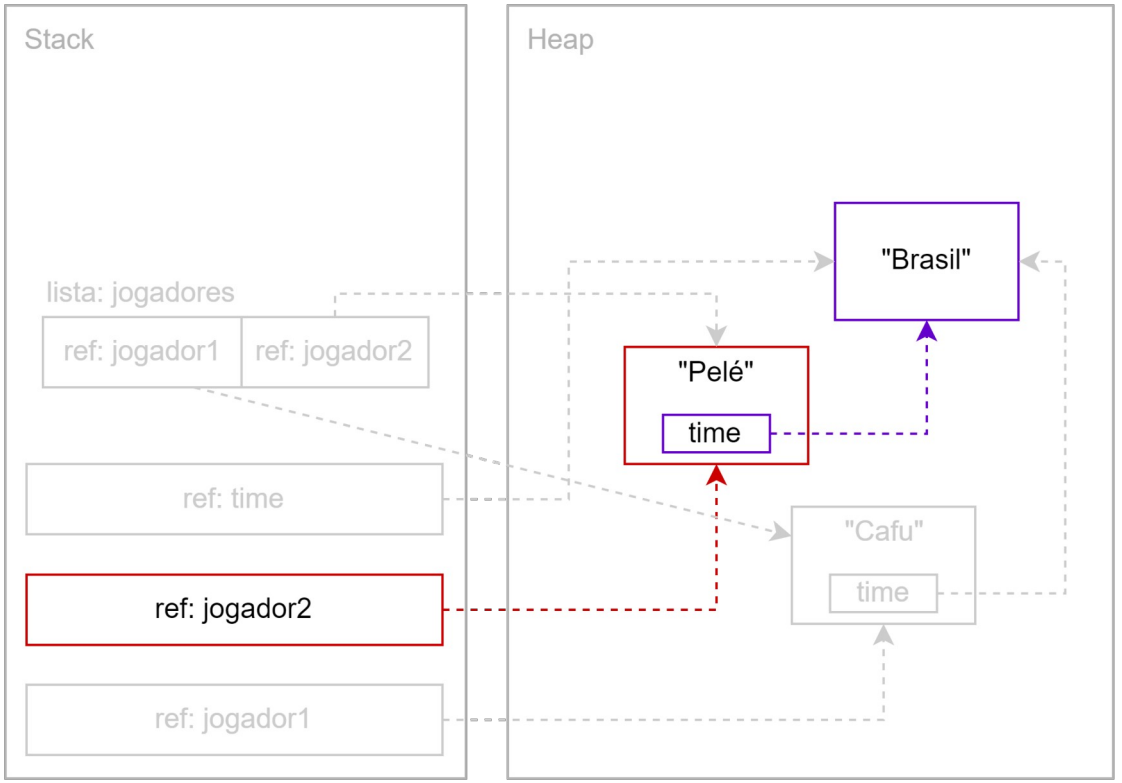
```
public class JogadorTeste03 {
   public static void main(String[] args) {

      Jogador jogador1 = new Jogador("Cafu");
      Jogador jogador2 = new Jogador("Pelé");
      Time time = new Time("Brasil");

      Jogador[] jogadores = {jogador1, jogador2};

      jogador1.setTime(time);
      jogador2.setTime(time);

      time.setJogadores(jogadores);
   }
}
```



7° PASSO

Adiciona valores ao atributo chamado **jogadores**, definido para receber uma lista, isso através da referência **time**, assim:

- Através da referência **time** na Stack o método **setJogadores** é acessado para adicionar jogadores ao time.
- Esse método recebe uma lista de objetos, essa lista está na memória Stack e agora além do nome o time tem 2 jogadores: Cafu e Pelé.
- Por meio da referência time o atributo jogadores foi definido com as informações contidas na lista presente na Stack.

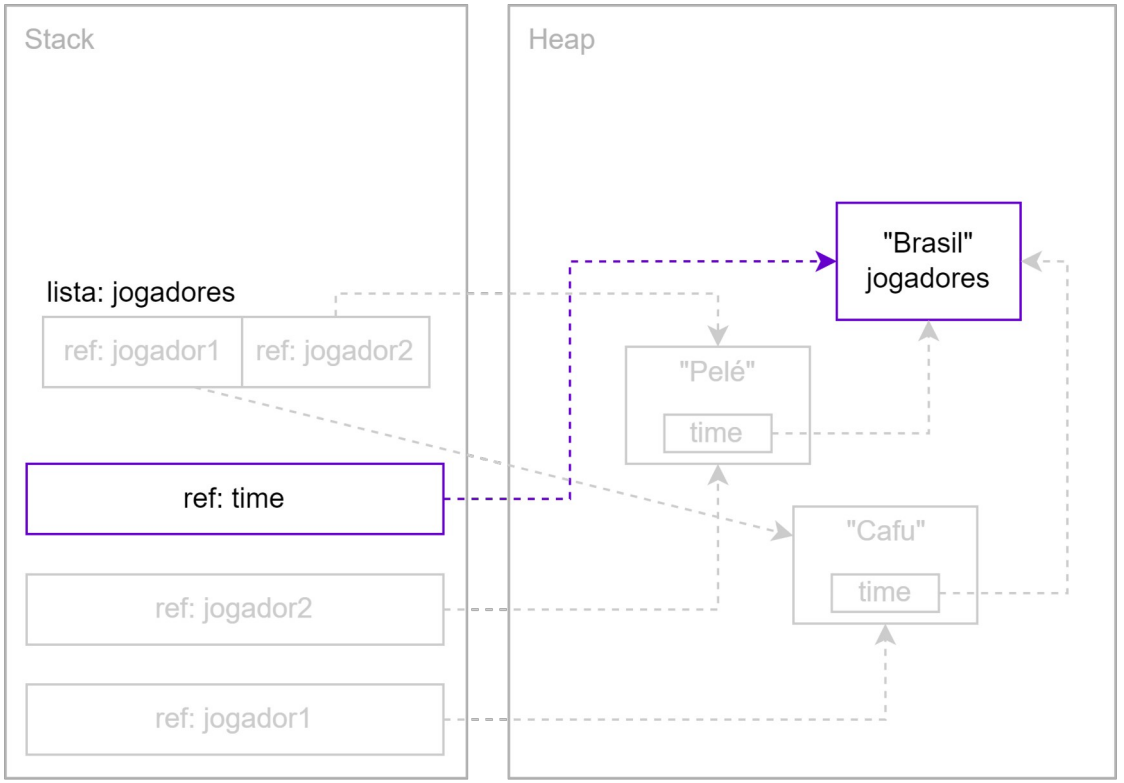
```
public class JogadorTeste03 {
   public static void main(String[] args) {

      Jogador jogador1 = new Jogador("Cafu");
      Jogador jogador2 = new Jogador("Pelé");
      Time time = new Time("Brasil");

      Jogador[] jogadores = {jogador1, jogador2};

      jogador1.setTime(time);
      jogador2.setTime(time);

      time.setJogadores(jogadores);
   }
}
```



AO FINAL DAS INSTRUÇÕES

Teremos um código funcional e totalmente compreensível até em seu comportamento na memória, sabemos:

- Criar objetos.
- Definir lista de objetos.
- Associar uma classe a outra, Time aceita vários Jogadores, mas Jogador só aceita um Time.

```
public class JogadorTeste03 {
    public static void main(String[] args) {

        Jogador jogador1 = new Jogador("Cafu");
        Jogador jogador2 = new Jogador("Pelé");
        Time time = new Time("Brasil");

        Jogador[] jogadores = {jogador1, jogador2};

        jogador1.setTime(time);
        jogador2.setTime(time);

        time.setJogadores(jogadores);

        // Instruções para impressão no console ocultas
    }
}
```