Alunos:

Eduardo Mendes
Isis Yasmim Almeida de Sousa
Leticia Moro
Luciemen Piazer

Roteiro de Teste Manual

Cenário: Sistema de Gerenciamento de Veículos
Objetivo: Validar as associações 1:1 (Placa obrigatória) e 0..1 (Rastreador opcional)

Passo 1 - Criar veículo com placa válida

Ação: Instanciar LicensePlate e Vehicle com dados válidos

```
1 reference
static void TestVehicleCreationWithValidLicensePlate()
{
    Console.WriteLine("1. Criar veículo com placa válida (1:1)");

    try
    {
        var licensePlate = new LicensePlate("ABC-1234");
        var vehicle = new Vehicle("Sedan", licensePlate);

        Console.WriteLine($"✅ SUCESSO: Veículo {vehicle.Model} criado com placa {vehicle.LicensePlate.Number}");
        Console.WriteLine($"   Estado do rastreador: {(vehicle.Tracker == null ? "Não possui" : "Possui")}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ FALHA: {ex.Message}");
    }
    Console.WriteLine();
}
```

Resultado Esperado: Veículo criado com sucesso, placa atribuída
Invariante Validado: Multiplicidade 1:1 - dependente obrigatório na criação

```
1. Criar veículo com placa válida (1:1)
✅ SUCESSO: Veículo Sedan criado com placa ABC-1234
   Estado do rastreador: Não possui
```

Passo 2: Tentar criar veículo sem placa

Ação: Passar null como LicensePlate no construtor

```
1 reference
static void TestVehicleCreationWithoutLicensePlate()
{
    Console.WriteLine("2. Tentar criar veículo sem placa (1:1)");

    try
    {
        var vehicle = new Vehicle("SUV", null!);
        Console.WriteLine("❌ FALHA: Permitiu criar veículo sem placa!");
    }
    catch (ArgumentNullException ex)
    {
        Console.WriteLine($"✅ SUCESSO: Impediu criação - {ex.ParamName} não pode ser nulo");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"✅ SUCESSO: Impediu criação - {ex.Message}");
    }
    Console.WriteLine();
}
```

Resultado Esperado: Exceção lançada, veículo não criado
Invariante Validado: Validação na fronteira - não permite estado inválido

```
2. Tentar criar veículo sem placa (1:1)
✅ SUCESSO: Impediu criação - licensePlate não pode ser nulo
```

Passo 3: Atribuir primeiro rastreador

Ação: Chamar AttachTracker() com rastreador válido

```
1 reference
static void TestAttachTrackerFirstTime()
{
    Console.WriteLine("3. Atribuir rastreador pela primeira vez (0..1)");

    try
    {
        var licensePlate = new LicensePlate("XYZ-9999");
        var vehicle = new Vehicle("Hatch", licensePlate);
        var tracker = new Tracker("R-001");

        bool success = vehicle.AttachTracker(tracker);

        if (success)
        {
            Console.WriteLine($"✅ SUCESSO: Rastreador {vehicle.Tracker!.SerialNumber} atribuído");
            Console.WriteLine($"   Veículo agora possui rastreador: {vehicle.Tracker != null}");
        }
        else
        {
            Console.WriteLine("❌ FALHA: Não conseguiu atribuir rastreador válido");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ FALHA: {ex.Message}");
    }
    Console.WriteLine();
}
```

Resultado Esperado: Rastreador atribuído com sucesso
Invariante Validado: Multiplicidade 0..1 - atribuição controlada

```
3. Atribuir rastreador pela primeira vez (0..1)
✅ SUCESSO: Rastreador R-001 atribuído
   Veículo agora possui rastreador: True
```

Passo 4: Tentar atribuir segundo rastreador

Ação: Chamar AttachTracker() novamente com outro rastreador

```
1 reference
static void TestAttachTrackerSecondTime()
{
    Console.WriteLine("4. Tentar atribuir segundo rastreador (0..1)");

    try
    {
        var licensePlate = new LicensePlate("LMN-4567");
        var vehicle = new Vehicle("Pickup", licensePlate);

        var tracker1 = new Tracker("R-101");
        var tracker2 = new Tracker("R-102");

        // Primeira atribuição deve funcionar
        vehicle.AttachTracker(tracker1);

        // Segunda atribuição deve falhar
        bool secondAttempt = vehicle.AttachTracker(tracker2);

        if (!secondAttempt)
        {
            Console.WriteLine("✅ SUCESSO: Impediu segunda atribuição de rastreador");
            Console.WriteLine($"   Rastreador atual mantido: {vehicle.Tracker!.SerialNumber}");
        }
        else
        {
            Console.WriteLine("❌ FALHA: Permitiu sobrescrever rastreador existente");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ FALHA: {ex.Message}");
    }
    Console.WriteLine();
}
```

Resultado Esperado: Método retorna false, rastreador original mantido
Invariante Validado: Não permite duplicação/sobrescrita silenciosa

```
4. Tentar atribuir segundo rastreador (0..1)
✅ SUCESSO: Impediu segunda atribuição de rastreador
   Rastreador atual mantido: R-101
```

Passo 5: Remover rastreador existente

Ação: Chamar DeleteTracker() quando rastreador está presente

```
static void TestRemoveTracker()
{
    Console.WriteLine("5. Remover rastreador existente (0..1)");

    try
    {
        var licensePlate = new LicensePlate("QWE-2025");
        var vehicle = new Vehicle("SUV", licensePlate);
        var tracker = new Tracker("R-500");

        vehicle.AttachTracker(tracker);
        Console.WriteLine($"   Rastreador antes da remoção: {vehicle.Tracker!.SerialNumber}");

        vehicle.DeleteTracker();

        if (vehicle.Tracker == null)
        {
            Console.WriteLine("✅ SUCESSO: Rastreador removido corretamente");
            Console.WriteLine($"   Estado do rastreador: {(vehicle.Tracker == null ? "Não possui" : "Ainda possui")}");
        }
        else
        {
            Console.WriteLine("❌ FALHA: Rastreador não foi removido");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"❌ FALHA: {ex.Message}");
    }
    Console.WriteLine();
}
```

Resultado Esperado: ✅ Rastreador removido, propriedade volta para null
Invariante Validado: Estado consistente - 0 ou 1, nunca >1

```
5. Remover rastreador existente (0..1)
   Rastreador antes da remoção: R-500
✅ SUCESSO: Rastreador removido corretamente
   Estado do rastreador: Não possui
```