Ambientes Virtuais de Execução

(1.º S 2010/2011)

Lista de Exercícios de Preparação para a 1ª Ficha

I Parte

1. Introdução à infra-estrutura .NET

1.1. Identifique e explique sucintamente os elementos da infra-estrutura .Net que suportam a interoperabilidade entre linguagens e a portabilidade dos componentes.

1.2. Indique uma utilização da metadata presente em módulos *managed* efectuada por cada uma das seguintes entidades: compilador da linguagem C#; ambiente de desenvolvimento Visual Studio; *loader* da máquina virtual; compilador JIT da máquina virtual.

1.3. Qual a finalidade do manifesto na definição de um *assembly*?

1.4. Para que servem as entradas `MemberRef` na metadata dos módulos managed? Porque razão a `MemberRef` de um método contém todos os parâmetros e valor de retorno do método?

1.5. O CLR suporta compilação "just-in-time". Descreva sumariamente o que acontece na primeira vez que um método é invocado e o que acontece nas invocações seguintes (Baseie a sua explicação num tipo e em métodos por si definidos).

1.6. Comente a seguinte afirmação: "A existência de um sistema de tipos comum (CTS) é condição suficiente para garantir a interoperabilidade entre linguagens .NET".

1.7. O que motivou a definição da "Common Language Specification" (CLS)?

2. Conceitos fundamentais do sistema de tipos

2.1. Na infra-estrutura Java também existem tipos valor com as mesmas características da infra-estrutura .NET? Justifique.

2.2. Explique a diferença entre coerção (8.3.2. *coercion*) e conversão (8.2.1 *casting*), tal como especificado na norma *Common Language Infrastructure* (CLI).

2.3. Seja uma instância `v1` do tipo valor `V`, que redefine `ToString`. É indiferente usar as seguintes versões de código para afixar uma instância na consola: `Console.WriteLine(v1)` e `Console.WriteLine(v1.ToString())`?

2.4. Por que motivo é necessária uma operação de "boxing" quando se invoca um método definido pelo tipo base de um tipo valor?

2.5. Qual o tempo de vida de uma instância de um tipo valor?

2.6. O método estático `bool System.Object.ReferenceEquals(object, object)` determina se duas referências referem o mesmo objecto. Não é essa a função do operador == quando aplicado a referências? Justifique a resposta.

2.7. Considere a seguinte definição do tipo `Ponto`. Se o tipo `Ponto` fosse definido como uma classe o código IL do construtor sofreria alguma alteração? E o código nativo? Justifique.

```
public struct Ponto {
    public int x, y;
    public Ponto(int x, int y) { this.x = x; this.y = y; }
    …
}
```

2.8. Considere o código em C, à frente, definidos em duas unidades de compilação distintas, A e B. Diga, justificando, se é necessário recompilar o módulo B se à estrutura T for retirado o comentário (1) e o módulo A for entretanto recompilado. O comportamento seria idêntico caso se tratasse de código equivalente escrito em C#?

Excerto de types.h

```
typedef struct t {
            /* char *name; */  (1)
            double val;
            /* … */
} T;
```

Excerto do Módulo A

```
#include "types.h"
…
T* createT() { return (T *)  malloc(sizeof(T));  }
```

Excerto do Módulo B

```
#include "types.h"
…
T *t1 = createT();
…
double d = t1->val;
```

2.9. Quais as vantagens de utilização de *value types* em relação a *reference types*?

2.10. Uma instância de ValueType pode estar localizada em *Managed Heap*? Justifique.

3. Estrutura de tipos (construtores e membros, métodos virtuais, atributo new e tabelas de métodos)

3.1. Qual a motivação do construtor de tipo? Explique as duas políticas que determinam o momento da chamada a este construtor?

3.2. Para a instrução `base.ToString()` o compilador de C# gera em IL um `call` ou `callvirt`? Justifique?

3.3. Porque é que a chamada ao método `GetType` de um tipo valor gera pelo compilador de C# um `call` enquanto que para um tipo referência gera um `callvirt`?

3.4. Considere uma hierarquia de três classes (A <- B <- C). Em C#, a chamada ao construtor sem parâmetros de C tem como consequência a chamada ao construtor sem parâmetros de B e depois de A, antes da execução de qualquer código específico do contrutor de C. Dada esta característica da linguagem C#, é seguro invocar métodos virtuais de um tipo na execução dos seus construtores?

3.5. Explique, usando um exemplo concreto, a relação entre o atributo `new` da linguagem C# e o atributo `newslot` da linguagem IL.

II Parte

1. Analise o seguinte tipo:

```
public struct A {
   public override String ToString() { return "Type 'A'"; }
   public void m() {
      A x = new A();
      A y = x;
      Object o1 = x;
      Object o2 = y;
      if (o1.Equals(o2)) Console.WriteLine("os objectos são iguais");
      else Console.WriteLine("os objectos são diferentes");
   }
}
```

   a) Diga, justificando, qual a mensagem apresentada na consola após a execução do método m() de A.
   b) Considerando as seguintes iniciações A a = new A(); A aa = a; diga, justificando, se em alguma das expressões a.ToString() e a.Equals(aa) é necessário efectuar a operação de boxing.

2. Considere o seguinte troço de código:

```
interface ICounter { void Increment(); }

struct Counter: ICounter {
  int value;
  public override string ToString() {
    return value.ToString();
  }
  public void Increment() { value++;  }
}
```

```
class Program  {
   static void Main() {
      Counter x = new Counter();
      Console.WriteLine(x);
      x.Increment();
      Console.WriteLine(x);
      ((ICounter)x).Increment();
      Console.WriteLine(x);
   }
}
```

   c) Indique, justificando, qual a saída resultante da execução.
   d) Acrescente no final do método Main a instrução "`Console.WriteLine(x.ToString())`".
      (1) Analise o código IL produzido, em particular o prefixo "`constrained. Counter`" (Norma CLI, parte III, secção 2.1).
      (2) Altere o código IL de forma a que seja produzido o mesmo resultado mas sem a utilização do prefixo `constrained`.
   e) Altere a categoria do tipo Counter para referência e explique, sucintamente, as diferenças no código IL produzido comparando com a alínea a).

3. Considere os excertos da implementação de Equals nas class B1, B2 e B3. Qual o único que se encontra correcto? Justifique. (Sugestão: tenha em conta a existência de classes derivadas destas.)

```
class B1 {
      public override bool Equals(object o) {
            B1 b1 = o as B1;
            if (b1 == null) return false;
            return true;
      }
}
class B2  {
      public override bool Equals(object o){
            if (o == null || o.GetType() != typeof(B2)) return false;
            return true;
      }
}

class B3 {
      public override bool Equals(object o) {
           if (o == null || o.GetType() != GetType()) return false;
           return true;
      }
}
```

4. Justifique a existência dos dois overloads do método `Equals` na estrutura `DateTime` do framework .Net 2 apresentada a seguir:

```
struct DateTime {
  //... outros campos e métodos
  public override bool Equals(object value){
      if (value is DateTime){
            DateTime time1 = (DateTime) value;
            return (this.InternalTicks == time1.InternalTicks);
      }
      return false;
  }

  public bool Equals(DateTime value) {
      return (this.InternalTicks == value.InternalTicks);
  }
}
```

5. Considere a definição da classe `Program` em CIL.
   a. Indique e justifique (descrevendo o que faz cada um dos métodos), o *output* resultante da execução do programa.
   b. Escreva um programa esquivalente em C#.

```
.class public auto Alinea4 extends [mscorlib]System.Object {

.method public hidebysig static int32 m(int32 a, int32 b) cil managed {

    .maxstack  3

    .locals init (int32 V_0)
              ldarg.1
              ldc.i4.1
              ble.s  IL_0011
              ldarg.0
              ldarg.1
              ldc.i4.1
              sub
              call       int32 Alinea4::m(int32, int32)
              ldarg.0
              add
              br.s       IL_0012
    IL_0011:  ldarg.0
    IL_0012:
              ret
  } // end of method Alinea4::m

  .method public hidebysig static void  Main() cil managed {
    .entrypoint
    .maxstack  8
          ldc.i4.2
          ldc.i4.5
          call       int32 Alinea4::m(int32, int32)
          call       void [mscorlib]System.Console::WriteLine(int32)
          ret
  } // end of method Alinea4::Main
} // end of class Program
```

`ble`	***Stack Transition:*** `…, value1, value2 → …` The `ble` instruction transfers control to *target* if `value1` is less than or equal to `value2`.
`br`	***Stack Transition:*** `… → …` Unconditional jump to target
`ldarg.0` `ldarg.1`	***Stack Transition:*** `… → …, value` The `ldarg.0` instruction pushes onto the evaluation stack, first incoming argument
`ldc.i4.s N` `ldc.i4.N`	***Stack Transition:*** `… → …, num` Pushes the integer value of N onto the evaluation stack as an int32.