

Ambientes Virtuais de Execução
(1.º S 2010/2011)
Lista de Exercícios de Preparação para a 3ª Ficha

1ª Parte

1) Reflexão

- a) Considere a classe `FieldInfo`, representante de um campo de um tipo. Qual é a informação que é necessário passar a `SetValue` para modificar um campo de instância?
- b) Será possível obter, através de `EventInfo`, o representante do campo de tipo *delegate* associado ao evento representado?
- c) Qual a necessidade do método `GetMethod(String methodName, Type[] types)` da classe `Type` receber um *array* de representantes de tipos.
- d) Tendo em conta aspectos de desempenho, qual das seguintes utilizações optaria para determinar se um tipo `T` tem aplicado o custom attribute `A`. Justifique.

<code>if (Attribute.GetCustomAttribute(typeof(T), typeof(A)) != null)...</code>
<code>if (Attribute.IsDefined(typeof(T), typeof(A)))...</code>

- e) Quais as restrições aos valores passados como argumentos a um *custom attribute*?
- f) A classe `System.Activator` contém o método estático `object CreateInstance(Type t, params object[] args)` que facilita a criação de instâncias via reflexão. O método retorna uma nova instância do tipo representado por `t` iniciada pelo construtor de parâmetros compatíveis com os argumentos presentes em `args`. O construtor usado tem de ser público.

Implemente o método `object CreateInstance(Type t, params object[] args)` da classe `AVEUtils`, com comportamento semelhante ao método da classe `System.Activator`, mas que permita a utilização de construtores não públicos.
- g) O método estático `CreateDelegate(Type t, object o, MethodInfo mi)`, da classe `Delegate`, permite criar uma instância do tipo *delegate* representado por `t`, com `target o` e método representado por `mi`. Poderá criar uma instância de um tipo *delegate* sem tirar partido deste método auxiliar ? (sugestão: analise a propriedade `MethodHandle` da classe `MethodInfo`).
- h) Comente a seguinte afirmação: “Se a um campo da classe `T` for aplicado o *custom attribute CA*, as instâncias de `T` ocuparão mais espaço no *heap*”.

1) Controlo de versões e partilha de componentes

- a) Para além de um nome único no contexto de uma organização, apresente outra motivação para que um *assembly* tenha *strong name*?
- b) Considere dois *assemblies*, `A` e `B`, sendo que o *assembly B* tem *strong name*. Justifique se, para cada um dos seguintes cenários, é necessário que o *assembly A* tenha *strong name*:
 - i) *Assembly A* usa o *assembly B*;
 - ii) *Assembly B* usa o *assembly A*.
- c) Considere o componente `C`, que usa o componente `D`, ambos com *strong name*. Onde está guardada a versão do componente `D` que `C` quer usar?
- d) Explique sucintamente de que forma é possível a determinada aplicação utilizar a nova versão de um componente com *strong name*, sem que a aplicação tenha de ser modificada.

2ª Parte

1. Implemente os seguintes métodos estáticos da classe `ReflectionUtils`:

- `bool ImplementsInterface(Type t, Type tIntf)` o qual verifica se o tipo representado por `t` implementa a interface representada por `tIntf`.
- `bool IsDisposablePattern(Type t)` o qual verifica se o tipo representado por `t` implementa o padrão *Disposable*, isto é, se contém todos os métodos necessários à concretização deste padrão.
- `bool isBoxedInstance(object o)` que determina se o objecto `o` corresponde à versão *boxed* de uma instância de tipo valor.
- `bool isCompatibleWithDelegate(Type t, MethodInfo mi)` que retorna *true* se o método `mi` tiver uma assinatura compatível com o tipo *delegate* representado por `t`.

2. Pretende-se desenvolver uma infra-estrutura para medir o tempo de execução de métodos de instância, sem parâmetros, de qualquer tipo.

- Implemente o *delegate* genérico `Action`, que retorna `void` e recebe 3 parâmetros de tipos genéricos distintos.
- Implemente o *custom attribute* `ProfileAttribute`, o qual só poderá ser aplicado uma vez a métodos e que recebe no construtor um valor do tipo `long`, correspondente ao número máximo de *ticks* que um método poderá demorar a executar.
- Implemente o método estático da classe `ProfileExecutor`, `void Measure<T>(T obj, Action<String, long, bool> result)`, o qual executa todos os métodos sem parâmetros de `obj` anotados com o atributo `ProfileAttribute`. Para cada execução é medido o número de *ticks* e chamado o *callback* `result`, passando-lhe o nome do método executado, o número de *ticks* correspondente à duração da execução e se este ultrapassou o máximo definido no atributo. O número de *ticks* corrente é obtido por `DateTime.Now.Ticks`.
- Implemente o método genérico `void ShowMeasure(T obj)` da classe `ProfileExecutor` que mostra no *standard output* o resultado da chamada ao método `Measure`, usando o seguinte formato para cada método de `obj` invocado:

```
Method=<method name>, ExecutionTime=<number of miliseconds>, TimeExceeded={true|false}
```

3. Na declaração de um evento apenas é especificado o tipo de *delegate* suportado, não havendo qualquer limitação quanto ao tipo que define o método referido pelo *delegate*. Pretende-se implementar uma infra-estrutura que dê suporte a este tipo de limitação, tal como demonstra o código seguinte:

```
class Group7Example {
    [ForbiddenType(typeof(Cx))] [ForbiddenType(typeof(Cy))] // proíbe os Cx e Cy
    public event Action<int> SomeEvent {
        add {
            if (EventUtils.IsDelegateAllowedByEvent(this, "SomeEvent", value)) someEvent += value;
        }
        remove {
            if (EventUtils.IsDelegateAllowedByEvent(this, "SomeEvent", value)) someEvent -= value;
        }
    }
    private Action<int> someEvent;
}
```

- Implemente o atributo `ForbiddenType`, o qual só poderá ser aplicado a eventos.
- Implemente o método estático `bool EventAllowsDelegate(EventInfo ei, Type type)` da classe `EventUtils`, o qual retorna *true* se o evento representado por `ei` não proibir o tipo representado por `type`, ou seja, se não tiver sido aplicado ao evento o atributo `ForbiddenType` especificando o representante `type`.
- Usando o método anterior, implemente o método estático `IsAllowedTypeForEvent` da classe `EventUtils` o qual retorna *true* se o primeiro parâmetro tiver associado um evento com o nome indicado no segundo parâmetro e esse evento permitir o registo do *delegate*.

4. Considere os assemblies apresentados em anexo ([prog.zip](#)).
 - a) Execute a aplicação `client`. Justifique a excepção apresentada.
 - b) Apresente uma solução em que passa a ser usada apenas a versão 2.0.0.0 do `assembly math.dll`.
 - c) Apresente uma solução em que continuam a ser usadas as versões referidas nos manifestos dos assemblies `client.exe` e `statisticutils.dll`.