1º Ficha de avaliação de Modelação e Padrões de Desenho
Licenciatura em Informática e Computadores
Semestre de Verão 2013/2014 Verão - Turmas LI41D e LI61D
(14-04-2014)

1. [4] Considere o sistema de tipos da linguagem Java.
   1.1. [1] Quais as categorias de tipos que existem em Java, que suportam a definição novos tipos.
   1.2. [1] Para cada uma das categorias identificados na alínea anterior, indique os diferentes tipos de membros que estes podem conter.
   1.3. [1] Qual a diferença entre membro de tipo e membro de instância?
   1.4. [1] Qual o espaço ocupado por um objecto em memória.

2. [4] Considere a *framework* de reflexão do Java implementada nos *packages* `java.lang` e `java.lang.reflect`.
   2.1. [1,5] O que representa a classe `Class` e cada instância desta classe? Quantas instâncias de classe existem num programa em execução?
   2.2. [2,5] Desenhe um diagrama de classes que relacione os tipos `Class`, `Field`, `Method`, `Constructor` e `Member`.

3. [12] Considere a *framework* de *binders* desenvolvida na aula. A interface `IBinderStrategy` e a classe `Binder` têm as seguintes definições:

```
public interface IBinderStrategy {
    <T> boolean bindMember(T newT, String key, Object value);
}

public class Binder {
    private final IBinderStrategy[] binderStrats;

    public Binder(IBinderStrategy... binderStrat) { ... }
    public static Map<String, Object> getFieldsValues(Object o) { ... }
    public <T> T bindTo(Class<T> klass, Map<String, Object> vals){ ... }
}
```

   3.1. [5] Pretende-se desenvolver um novo `IBinderStrategy`, `PropertiesWithBackingField`, que apenas afectam propriedades de instância (de acordo com a definição de *Java Beans*) que têm um campo associado. O Campo deve ter o mesmo nome da propriedade, ignorando maiúsculas ou minúsculas.
   Em seguida apresenta-se um exemplo de código de uma classe que contém duas propriedades. A propriedade `Prop1` que será afectada por este novo *Binder Strategy* e propriedade `Prop2` não.
   Desenvolva pelo menos um teste unitário em JUnit que verifique a correcção da solução.

```
public class ClassWithProperties {
        private String prop1;
        private String foo;
        public void setProp1(String prop1) { this.prop1 = prop1; }
        public void setProp2(String prop2) { this.foo = prop2; }
}
```

   NOTA: Na resolução deste exercício, apresente também o diagrama UML onde localiza a classe desenvolvida na hierarquia de *Binder Strategies*.

   3.2. [7] De modo a não ser obrigatório que o campo tenha o mesmo nome da propriedade, crie a anotação `BackingField` que indica o nome do campo que armazena o valor da propriedade, conforme apresentado na listagem seguinte.

```
public class ClassWithProperties {
      private String foo;
      @BackingField("foo")
      public void setProp2(String prop2) { this.foo = prop2; }
}
```

   Implemente a anotação `BackingField` e apresente uma solução para ter um *Binder Strategy* que suporta esta anotação. Deve ter em consideração o reaproveitamento de código relativamente ao já desenvolvido. Na solução final indique o/os padrões de desenho que utilizou.
   Desenvolva pelo menos um teste unitário em JUnit que verifique a correcção da solução.