2º Ficha de avaliação de Modelação e Padrões de Desenho
Licenciatura em Informática e Computadores
Semestre de Verão 2013/2014 Verão - Turmas LI41D e LI61D
(18-06-2014)

1. [3] Implemente o método estático `Iterable<String> iterFields(Object o)` que retorna um `Iterable` de *Strings* com o nome de todos os campos não públicos do objecto `o`.

2. [7] Considere o método estático: `<T> Predicate<T> and(Predicate<T> p1, Predicate<T> … pArray)` da classe `Utils`, que retorna um novo predicado representando o operador lógico AND entre os predicados recebidos por parâmetro (`p1` e `pArray`).
   2.1. Realize três testes unitários que demonstram a correcção do método implementado. Um em que este método recebe um predicado, outro com dois e outro com três.
   2.2. Implemente o método `and`.

3. [10] Considere a classe `IterUtils` e a interface `Queryable` apresentados em seguida.
   O método zip da interface `Queryable`, retorna um novo `Queryable` com a combinação de cada elemento do `Queryable` sobre o qual o método é chamado com o elemento do mesmo índice no `Iterable secondIter`, passado como argumento. A combinação de cada um dos elementos é realizada pela função `resultSelector` passada como argumento. Note-se que ambos os `Iterable` envolvidos nesta operação podem ter número de elementos diferente e o `Iterable` resultante tem o número de elementos correspondente ao menor dos dois.
   Abaixo tem um exemplo de utilização Ilustrativo da utilização e o respectivo output.

```
public final class IterUtils {
   public static <T> Queryable<T> query(Collection<T> elems);
}

public interface Queryable<T1> extends Iterable<T1> {
   <T2, R> Queryable<R> zip(Iterable<T2> secondIter, BiFunction<T1, T2, R> resultSelector);
}
```

```
// Exemplo de utilização
List<Integer> numbers = Arrays.asList(1,2,3,4,5);
List<String> words = Arrays.asList("O","Benfica","é","o","campeão","nacional", "2013/2014");
IterUtils.query(numbers).zip(words, (n, w) -> n + " - " + w).forEach(System.out::println);

// Output apresentado
1 - O
2 - Benfica
3 - é
4 - o
5 - campeão
```

   3.1. Realize uma classe que implementa a interface `Queryable` de forma *Eager*.
   3.2. Realize uma classe que implementa a interface `Queryable` de forma *Lazy*.