2º Ficha de avaliação de Modelação e Padrões de Desenho - Repetição
Licenciatura em Informática e Computadores
Semestre de Verão 2013/2014 Verão - Turmas LI41D e LI61D
(23-06-2014)

1. [3] Implemente o método estático `Iterable<String> iterProperties(Object o)` que retorna um `Iterable` de *Strings* com o nome de todos as propriedades públicas, segundo a convenção *Java Beans* com pelo menos o método de acesso, do objecto `o`.

2. [8] Considere o método estático: `<T1,T2> Function<T2, R> partial(BiFunction<T1, T2, R> biFunction, T1 t1)` da classe `Utils`, que recebe como argumento a função `biFunction` (com dois argumentos, T1 e T2, e retorna R), fixando o primeiro parâmetro com valor de t1 e retorna uma função que recebe apenas T2 e que, quando chamada, utiliza o T1 fixado, o T2 passado como parâmetro e retorna R.

```
// Exemplo de utilização
Function<Integer, Float> invMul = partial(
      (Integer x, Integer y) -> (float)x.intValue() / y.intValue(), 1
   );
   float f = invMul.apply(2);
   System.out.println(f);

// Output apresentado
0.5
```

2.1. Realize dois testes unitários que demonstram a correcção do método implementado.
2.2. Implemente o método `partial`.
2.3. No exemplo anterior, da declaração da função lambda passada à função `partial`, seria necessário definir explicitamente o tipo dos argumentos `x` e `y`? Justifique.

3. [9] Considere a classe `IterUtils` e a interface `Queryable` apresentados em seguida.
O método `concat` da interface `Queryable`, retorna um novo `Queryable` com a concatenação do `Iterable` sobre o qual o método é chamado com o `Iterable secondIter`, passado como argumento. Como optimização, se um dos *Iterables* não tiver elementos, é retornado o outro.

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

public interface Queryable<T> extends Iterable<T> {
   Queryable<T> concat(Iterable<T> secondIter);
}
```

```
// Exemplo de utilização
List<String> words1 = Arrays.asList("O","Benfica","é");
List<String> words2 = Arrays.asList("o","campeão","nacional", "2013/2014");
IterUtils.query(words1).concat(words2).forEach(System.out::println);

// Output apresentado
O
Benfica
é
o
campeão
nacional
2013/2014
```

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