Aufgabenblatt 01

Bearbeitungsende: 10.04.2016

Aufgabe 1 [Programmierung]

Ein Paar sind zwei Objekte, die als Einheit betrachtet werden.

Implementieren Sie eine generische Klasse `Paar<E,Z>` zur Repräsentation von Paaren von Objekten mit beliebigen Typen `E` und `Z`.

Die beiden Objekte sollen dem `Paar`-Konstruktor als Parameter übergeben werden. Darüberhinaus soll die generische Klasse folgende Methoden anbieten:

- `erstes()` und `zweites()`

  geben die beiden Komponenten des Paares zurück

- `toString()`

  erzeugt für das Paar eine Darstellung als `String` in der Form `"(`*erstes*`,`*zweites*`)"`, worin die beiden Platzhalter für die Stringdarstellungen der beiden Objekte stehen.

Aufgabe 2 [Programmierung]

In den Codebeispielen zur Vorlesung finden Sie die generische Klasse `FeldFix<T>`. Deren Implementierung haben wir noch nicht vollständig besprochen, was für diese Aufgabe aber auch nicht notwendig ist. Außerdem finden Sie die generische Schnittstelle `Feld<T>`, die von `FeldFix<T>` implementiert wird.

Implementieren Sie eine generische Klasse `PaarFeld<E,Z>`. Eine Instanz einer Klasseninstanz referenziert zwei gegebene `Feld`-Instanzen mit beliebigen Typparametern `E` und `Z`.

Die Klasse soll einen Konstruktor bieten, der Referenzen auf die zwei Felder annimmt. Eine `PaarFeld`-Instanz soll die beiden zugrundeliegenden Felder nur referenzieren, nicht als eigene übernehmen.

Die Methodensignaturen der Klasse sollen so aussehen und sich so verhalten, als wäre sie die Instanz `Feld<Paar<E,Z>>`. Bei Lesezugriff sind also `Paar<E,Z>`-Instanzen (siehe Aufgabe 1) zu erzeugen und zurückzugeben.

Falls die beiden zugrundeliegenden Felder nicht gleich lang sind, bestimmt das kürzere die effektive Länge des Paar-Feldes.

Aufgabe 3 [Theorie]

Betrachten Sie folgenden Quelltext:

```
class A { }
class B extends A { }
class C extends A { }
```

```
class D extends C { }
class E { }

public class Foo {
  public static <T> T f(T x, T y) {
    return x;
  }
  public static <T> T g(T[] x, T[] y) {
    return x[0];
  }
  public static <T> T h(T x, T[] y) {
    return x;
  }
  public static <S,T> S k(S[] x, T y) {
    return x[0];
  }
  public static void main(String[] args) {
    Object o;
    A a = new A();
    B b = new B();
    C c = new C();
    D d = new D();
    E e = new E();

    Object[] of;
    A[] af = new A[1];
    B[] bf = new B[2];
    C[] cf = new C[3];
    D[] df = new D[4];
    E[] ef = new E[5];

    Object[][] off;
    A[][] aff = new A[1][1];
    B[][] bff = new B[2][2];
    C[][] cff = new C[3][3];
    D[][] dff = new D[4][4];
    E[][] eff = new E[5][5];

    // Anweisung
  }
}
```

Akzeptiert der Compiler jeweils den obigen Code, wenn die folgenden Anweisungen für "`// Anweisung`" eingefügt werden? Begründen Sie kurz jede Antwort.

(01) `a = f(a,c);`

(02) `a = f(c,a);`

(03) `a = f(b,d);`

(04) `a = f(c,d);`

(05) `b = f(a,c);`

(06) `b = f(c,d);`

(07) `e = f(e,a);`

(08) `o = f(b,e);`

(09) `c = f(b,d);`

(10) `af = f(bf, cf);`

(11) `bf = f(bf, cf);`

(12) `of = f(bf, c);`

(13) `of = f(bf, df);`

(14) `o = f(df, ef);`

(15) `af = g(bf,df );`

(16) `a = g(cf, df);`

(17) `a = g(bf, ef);`

(18) `a = g(aff, cf);`

(19) `af = g(bff, dff);`

(20) `a = h(b, bf);`

(21) `a = h(bf, b);`

(22) `c = h(d, bf);`

(23) `a = h(b, df);`

(24) `af = h(bf, cff);`

(25) `o = h(cff, bf);`

(26) `cf = h(cf, cf);`

(27) `e = k(ef, e);`

(28) `a = k(df, ef);`

(29) `b = k(cf, b);`

(30) `of = k(bff, eff);`

Lösungen zu mit [**Programmierung**] markierten Aufgaben sind im **Praktomat** einzureichen!
Allgemeine **Fragen** zu den Aufgaben können Sie im **LEA-Forum** ***Übungsaufgaben*** stellen.
Hilfe bei der Lösung der Aufgaben erhalten Sie in den **Übungen**.