Parcial CAP

Curs 2017-18 (22/XI/2017) Duració: 2 hores

1.- (1.5 punts) Fes petits bocinets de Smalltalk (per ser executats al *Workspace*) per respondre les següents qüestions:

- Trobar tots els mètodes que envien el missatge **`#callcc:`**
- Trobar totes les classes que tenen un mètode anomenat **`#callcc:`**
- Trobar tots els mètodes anomenats **`#callcc:`**
- Trobar quantes classes implementen un mètode anomenat **`#callcc:`**

```
a) SystemNavigation default allCallsOn: #callcc:
an OrderedCollection(AMB>>#try: AMB2>>#try: BlockClosure>>#repeatUntil:
BlockClosure>>#mentreCert: ContinuationTest>>#testBlockTemps
ContinuationTest>>#testBlockVars ContinuationTest>>#testReentrant
ContinuationTest>>#testBlockEscape ContinuationTest>>#testSimplestCallCC
ContinuationTest>>#testSimpleCallCC ContinuationTest>>#testMethodTemps
Coroutine>>#initialize HeavenAndHell>>#devil: HeavenAndHell>>#milestone:
TND>>#extreu: Tardis class>>#angelPloraner: Tardis class>>#fita:).

b) SystemNavigation default allImplementorsOf: #callcc:.
an OrderedCollection(Continuation class>>#callcc: ContinuationTest>>#callcc:)

c) SystemNavigation default allMethodsSelect: [ :m | m selector == #callcc: ].
an OrderedCollection(Continuation class>>#callcc: ContinuationTest>>#callcc:)

d) SystemNavigation default numberOfImplementorsOf: #callcc:
2
```

2.- (2 punts) Defineix *introspecció* i *intercessió*. Dóna exemples d'introspecció i intercessió en Smalltalk i Java (un de cada a cada llenguatge).

Les definicions són a les transparències.
- Introspecció:
  Smalltalk: Poder *veure* la pila d'execució amb **`thisContext`**.
  Java: Poder saber quins mètodes i atributs té una classe a Java (amb **`getDeclaredMethods`** i **`getDeclaredFields`** de **`Class`**).
- Intercessió:
  Smalltalk: Poder *canviar* la pila d'execució, gràcies a **`thisContext`**.
  Java: Poder interceptar les invocacions a mètode amb l'ús de la classe **`java.lang.reflect.Proxy`**.

3.- (1.5 punt) Imaginem una expressió:

`Continuation callcc: [ :k | ... <cos del bloc>... ]`

on a **`<cos del bloc>`** s'ignora el paràmetre **`k`** del bloc (és a dir, no fem *absolutament res* amb la continuació). Sustitueix l'expressió anterior per una expressió *equivalent* que no utilitzi la classe **`Continuation`**.

`[ :k | ... <cos del bloc>... ] value: nil` (aquest valor és irrellevant)

4.- (2.5 punts) Implementeu els bucles de tipus

```
repeat
     <body>
until <boolean expression>
```

en Smalltalk: **`BlockClosure >> #repeatUntil: aBlock`**, utilitzant continuacions. El paràmetre **`aBlock`** ha de ser un bloc sense paràmetres que, en ser avaluat, retorni un booleà.

```
BlockClosure >> repeatUntil: aBlock
     | cont |
      self value.
      cont := Continuation callcc: [ :cc | cc ].
      aBlock value ifFalse: [self value. cont value: cont]
                   ifTrue:  [^ nil].
```

5.- (2.5 punts) Escriviu un mètode que, passant un *class object* com a paràmetre, retorni **`true`** si aquesta classe té algun mètode propi que retorni un *class object*, és a dir, un objecte instància de **`Class`**. Què passa amb el mètode demanat si treiem *propi* de la pregunta anterior?

```
    public boolean retorna_class (Class c) {
      if (c != null) {
          try {
            Method m[] = c.getDeclaredMethods();
            for (int i = 0; i < m.length; i++)
                if ((m[i].getReturnType()).equals(Class.class)) return true;
          }
          catch (Throwable e) {
            System.err.println(e);
          }
      }
      return false;
    }
```

El mètode és trivial si treiem el *propi* de l'enunciat ja que Object té un mètode (getClass()) que retorna un *class object*. Així doncs, el mètode demanat sempre retornarà true.