

PRÀCTICA CAP Q1 CURS 2016-2017

# CORUTINES

Robert Almar Graupera

Daniel García Romero

## ÍNDIX

---

1. Implementació de la classe Coroutine .....	02
2. Testing de la classe Coroutine .....	06
3. Solució al problema dels matrimonis estables .....	10
4. Testing de la solució .....	12
5. Bibliografia .....	16

# 1. Implementació de la classe Coroutine

Per tal d'implementar la classe Coroutine, primer de tot hem obert el System Browser i hem afegit un Package al qual hem anomenat Practica.

Tot seguit, hem definit la classe Coroutine amb les següents variables d'instància: savedBlock, resumeBlock i continuacio, la utilitat de les quals l'anirem explicant a continuació.

```
Object subclass: #Coroutine
```

```
instanceVariableNames: 'savedBlock resumeBlock continuacio'
```

```
classVariableNames: ''
```

```
category: 'Practica'
```

Aquestes variables d'instància tindran els seus *getter* i *setter* corresponents, els quals podem crear automàticament amb la opció de *refactoring* que ens ofereix Smalltalk.

**continuacio**

```
^ continuacio
```

**continuacio: anObject**

```
continuacio := anObject
```

**resumeBlock**

```
^ resumeBlock
```

**resumeBlock: anObject**

```
resumeBlock := anObject
```

**savedBlock**

```
^ savedBlock
```

**savedBlock:** *anObject*

```
savedBlock := anObject
```

La classe Coroutine podrà construir objectes que es comportin com a corutines mitjançant el mètode *maker* que definirem al Class side i mitjançant el mètode d'instància *initialize* amb protocol *initialize-release*. Aquest últim es cridarà quan fem un *new* al *maker*.

Caldrà rebre com a paràmetre un bloc que contindrà el codi de la corutina i guardar-lo a *savedBlock*.

**maker:** *block*

```
^ self new savedBlock: block
```

Aquest bloc tindrà dos paràmetres: *resume* i *value*. Una corutina podrà invocar-ne d'altres via el paràmetre *resume*, que serà un bloc amb dos paràmetres:

*resume value: <nom corutina> value: <valor passat a la corutina invocada>*

A *initialize* definirem aquest *resume*, el qual serà el mateix per cada corutina independentment de qualsevol codi que hi posem a dins de cadascuna d'elles. El contingut d'aquest bloc que guardarem a *resumeBlock* s'entén millor si expliquem primer què fa el mètode d'instància *value*.

Invoquem corutines mitjançant *value : valor*. En un programa fet amb corutines només cal arrencar la primera corutina, ja que a partir d'aquest moment, les corutines es comencen a cridar entre elles.

Quan s'invoca per primer cop, una corutina ha de començar a executar el codi especificat en el bloc que es passa com a paràmetre a *maker*, és a dir, *savedBlock*, amb els paràmetres *resumeBlock* i *valor*.

Si durant l'execució troba una expressió del tipus resume value: b value: 'A', conceptualment vol dir que ha d'invocar la corutina b amb valor 'A' i, programàticament, que s'ha d'executar resumeBlock amb paràmetres b i 'A'.

Tenint en compte tot això, ara ja podem parlar del contingut de resumeBlock. Aquest invocarà a la corutina nomCor amb valorCor mitjançant el mètode value que acabem de comentar, però no només haurà de fer això. També caldrà que guardi la pila d'execució a continuacio per tal de restaurar-la quan em tornin a cridar i poder continuar amb l'execució del savedBlock de la corutina.

**initialize**

```
super initialize.  
  
resumeBlock := [ :nomCor :valorCor |  
    continuacio := Continuation fromContext: thisContext sender.  
    nomCor value: valorCor.  
].
```

**value: valor**

```
continuacio notNil  
  
ifTrue: [continuacio value: valor]  
  
ifFalse: [savedBlock value: resumeBlock value: valor.]
```

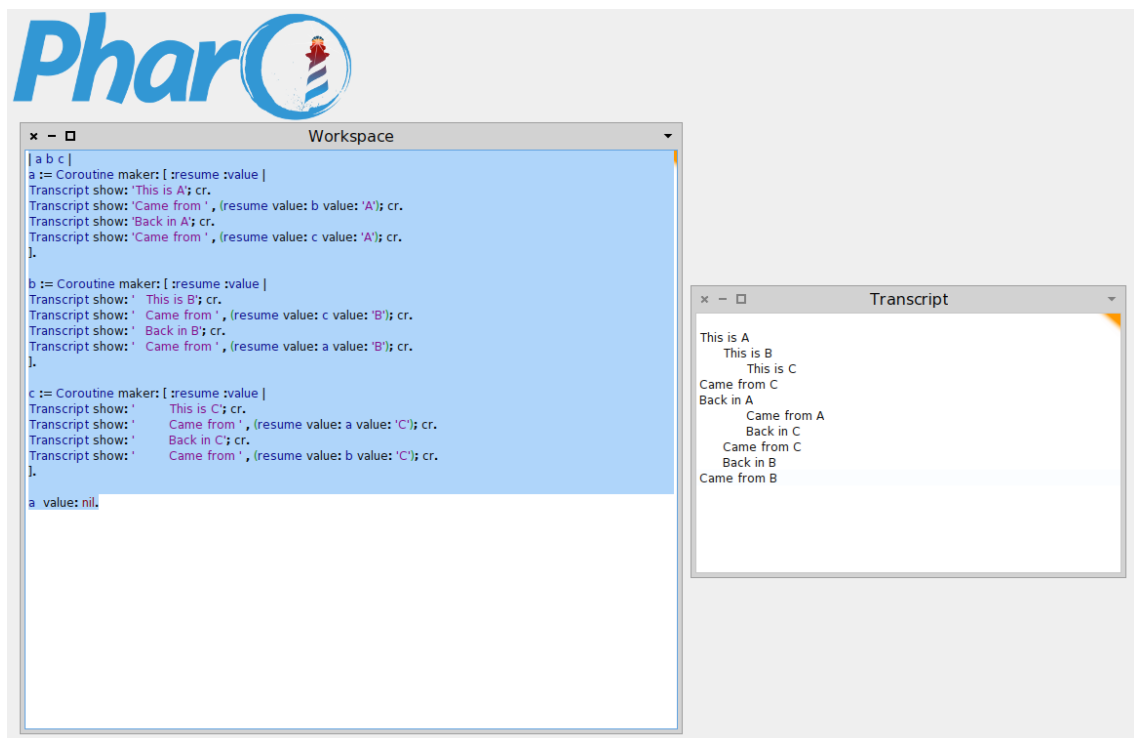
Inicialment, teníem una variable d'instància més, firstTime, la qual posàvem a true a initialize i a false just abans d'avaluar savedBlock, però ens vam adonar que no calia perquè amb continuacio en teníem suficient per comprovar si la corutina s'invocava per primer cop (ifFalse del mètode value) o no (ifTrue).

Pel que fa al procés de guardar i restaurar la pila d'execució:

- `fromContext` crea una instància de `Continuation` i la inicialitza amb el mètode `initializeFromContext`, que el que farà és guardar, a una variable d'instància `values`, la pila associada al context que es passa com a paràmetre, que en aquest cas serà el context que representa l'execució del bloc, és a dir, `thisContext sender`.
- El mètode `value` de la classe `Continuation` s'encarregarà de recuperar el context que teníem guardat, convertir-lo en el context actual i retornar l'objecte que rep com a paràmetre per poder continuar amb l'execució de la corutina.

## 2. Testing de la classe Coroutine

• **Jocs de proves.** Hem verificat que el resultat de l'exemple de l'enunciat es correspon amb la sortida que hem obtingut al Transcript. A més a més, n'hem pensat i provat un altre en el qual, a diferència del primer, el valor que li passem a la corutina a no és irrellevant (l'utilitza per mostrar un *This is A* al Transcript), invoquem a b per tornar després a a (b, cridada per primera vegada, aquí sí que mostra qui l'ha cridat), la qual invoca a c amb valor irrellevant, c invoca a b (esperant una resposta que no rebrà perquè no tornarà a ser invocada) i, finalment, b invoca a a amb valor irrellevant (demostrant que podem cridar més de dos cops a una mateixa corutina i a vegades fer ús del valor de retorn i d'altres no) i aquesta finalitza la seva execució amb un *Back in A again* al Transcript.



The screenshot shows the Phar IDE interface. The 'Workspace' window contains the following code:

```
[ a b c ]
a := Coroutine maker: [ :resume :value |
Transcript show: 'This is A'; cr.
Transcript show: 'Came from ', (resume value: b value: 'A'); cr.
Transcript show: 'Back in A'; cr.
Transcript show: 'Came from ', (resume value: c value: 'A'); cr.
].

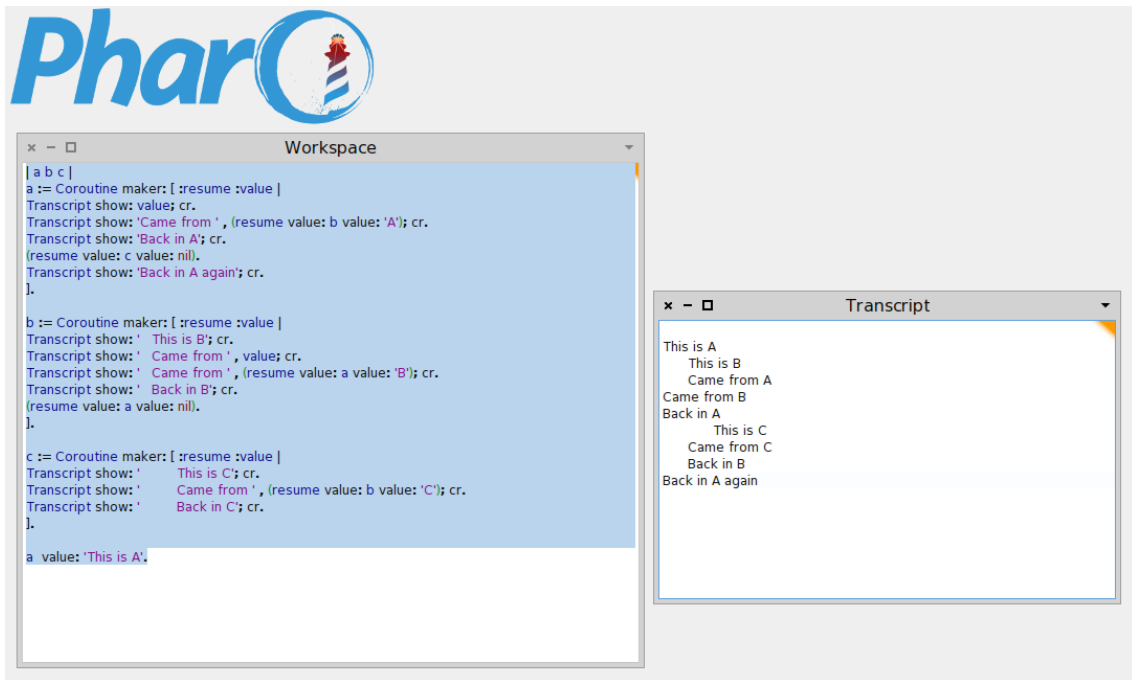
b := Coroutine maker: [ :resume :value |
Transcript show: ' This is B'; cr.
Transcript show: ' Came from ', (resume value: c value: 'B'); cr.
Transcript show: ' Back in B'; cr.
Transcript show: ' Came from ', (resume value: a value: 'B'); cr.
].

c := Coroutine maker: [ :resume :value |
Transcript show: ' This is C'; cr.
Transcript show: ' Came from ', (resume value: a value: 'C'); cr.
Transcript show: ' Back in C'; cr.
Transcript show: ' Came from ', (resume value: b value: 'C'); cr.
].

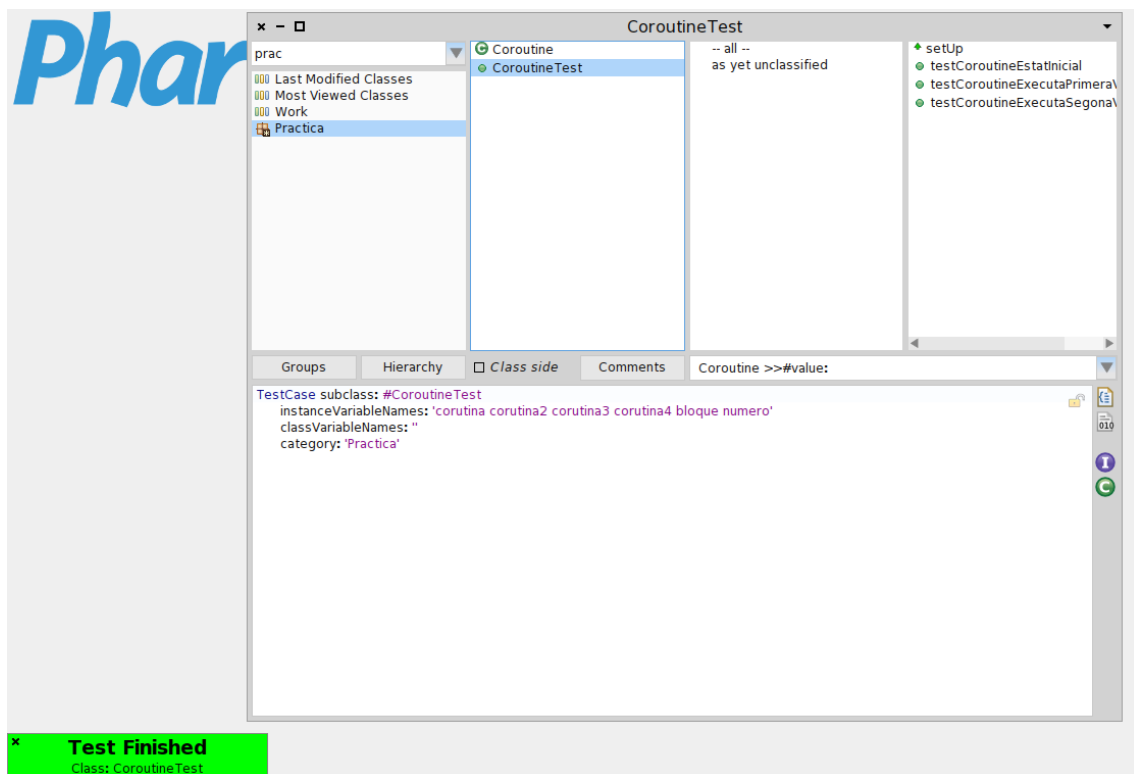
a value: nil.
```

The 'Transcript' window shows the following output:

```
This is A
  This is B
    This is C
      Came from C
    Back in A
      Came from A
    Back in C
      Came from C
    Back in B
  Came from B
```



- **TDD.** Basant-nos en l'exemple de PastaTest que vam veure a teoria, hem implementat la classe CoroutineTest amb tres tests i un mètode setUp per inicialitzar les variables d'instància.





```

TestCase subclass: #CoroutineTest
instanceVariableNames: 'corutina corutina2 corutina3
corutina4 bloque numero'
classVariableNames: ''
category: 'Practica'

setUp
corutina := Coroutine maker: [:resume :value |
Transcript show: 'This is A'; cr.].
bloque := corutina savedBlock.
numero := 0.
corutina2 := Coroutine maker: [:resume :value |
numero := 2+3].
corutina3 := Coroutine maker: [ :resume :value |
numero := numero+1.
numero := numero + (resume value: corutina4 value:'A').
numero := numero+3].
corutina4 := Coroutine maker: [ :resume :value |
(resume value: corutina3 value: 2)].

testCoroutineEstatInicial
|aux|
aux := [:resume :value | Transcript show: 'This is A';
cr.] asString.
self assert: bloque asString = aux.
self assert: corutina continuacio isNil.

```

**testCoroutineExecutaPrimeraVegada**

`corutina2` value: `nil`.

`self` assert: `numero` = `5`.

`self` assert: `corutina2` continuacio isNil.

**testCoroutineExecutaSegonaVegada**

`corutina3` value: `nil`.

`self` assert: `numero` = `6`.

`self` assert: `corutina3` continuacio notNil.

### 3. Solució al problema dels matrimonis estables

```
|n w m mchoice wrank wengagement corutinaMain|
n := 3.
w := Array new: n.
m := Array new: n.
mchoice := Matrix rows: 3 columns: 3.
wrank := Matrix rows: 3 columns: 3.
wengagement := Array new: n.
corutinaMain := Coroutine maker: [ :resume :value |
    "Transcript show: 'This is corutinaMain'; cr."
    "for para que las corutinas tengan su indice en value"
    1 to: n do: [:each |
        resume value: (m at: each) value: each.
        resume value: (w at: each) value: each.
    ].
    "set up choices and rankings"
    mchoice at: 1 at: 1 put: 1. mchoice at: 1 at: 2 put: 2. mchoice at: 1 at: 3 put: 3.
    mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 3. mchoice at: 2 at: 3 put: 2.
    mchoice at: 3 at: 1 put: 2. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 1.
    wrank at: 1 at: 1 put: 2. wrank at: 1 at: 2 put: 1. wrank at: 1 at: 3 put: 3.
    wrank at: 2 at: 1 put: 2. wrank at: 2 at: 2 put: 3. wrank at: 2 at: 3 put: 1.
    wrank at: 3 at: 1 put: 1. wrank at: 3 at: 2 put: 2. wrank at: 3 at: 3 put: 3.
    "mchoice at: 1 at: 1 put: 2. mchoice at: 1 at: 2 put: 1. mchoice at: 1 at: 3 put: 3.
    mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 2. mchoice at: 2 at: 3 put: 3.
    mchoice at: 3 at: 1 put: 1. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 2.
    wrank at: 1 at: 1 put: 1. wrank at: 1 at: 2 put: 2. wrank at: 1 at: 3 put: 3.
    wrank at: 2 at: 1 put: 3. wrank at: 2 at: 2 put: 2. wrank at: 2 at: 3 put: 1.
    wrank at: 3 at: 1 put: 3. wrank at: 3 at: 2 put: 1. wrank at: 3 at: 3 put: 2."
    1 to: n do: [:i |
        resume value: (m at: i) value: nil.
    ].
    "here output the solution held in wengagement"
    Transcript show: wengagement.
].
1 to: n do: [:each |
    m at: each put:
    (
        Coroutine maker: [ :resume :value |
```

```

    |choice|
    "value = me/manindex"
    "Transcript show: 'This is a man coroutine and value = ' ; show: value; cr."
    resume value: corutinaMain value: nil.
    "propose to preferences in order"
    1 to: n do: [:j |
        choice := mchoice at: value at: j.
        resume value: (w at: choice) value: value.
    ].
]
).
w at: each put:
(
    Coroutine maker: [:resume :value |
        |suitor jilt|
        "value = me/womanindex"
        "Transcript show: 'This is a woman coroutine and value = ' ; show: value; cr."
        suitor := resume value: corutinaMain value: nil.
        "accept first offer"
        wengagement at: value put: suitor.
        suitor := resume value: corutinaMain value: nil.
        "loop all the while accepting better offers"
        1 to: n-1 do: [:loop |
            (wrank at: value at: suitor) < (wrank at: value at: (wengagement at: value))
            ifTrue: [
                jilt := wengagement at: value.
                wengagement at: value put: suitor.
                suitor := resume value: (m at: jilt) value: nil.
            ]
            ifFalse: [
                suitor := resume value: (m at: suitor) value: nil.
            ]
        ].
    ].
).
].
corutinaMain value: nil.

```

## 4. Testing de la solució

El primer que hem fet ha estat provar l'exemple de l'article i hem obtingut el resultat esperat: wengagement = (2 3 1), és a dir, la dona 1 (A) amb l'home 2, la dona 2 (B) amb l'home 3 i la dona 3 (C) amb l'home 1.

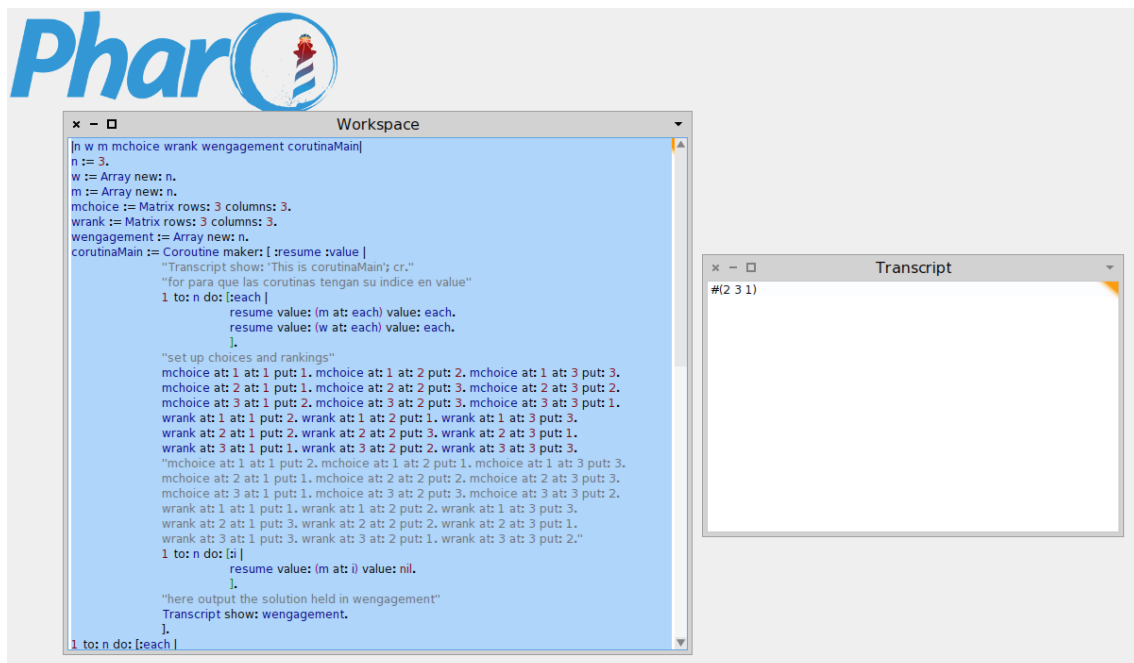
As an example suppose there are three men 1, 2 and 3 and three women A, B and C with preferences

1	A	B	C
2	A	C	B
3	B	C	A

A	2	1	3
B	2	3	1
C	1	2	3

The stable marriage is then

(1, C)      (2, A)      (3, B)



The screenshot shows the PharC workspace with a code editor and a transcript window. The code implements a Stable Marriage algorithm using a coroutine-based approach. The transcript shows the final wengagement result as #2 3 1.

```

x - □ Workspace
in w m mchoice wrank wengagement corutinaMain
n := 3.
w := Array new: n.
m := Array new: n.
mchoice := Matrix rows: 3 columns: 3.
wrank := Matrix rows: 3 columns: 3.
wengagement := Array new: n.
corutinaMain := Coroutine maker: [ :resume :value |
  "Transcript show: 'This is corutinaMain'; cr."
  "for para que las corutinas tengan su indice en value"
  1 to: n do: [each |
    resume value: (m at: each) value: each.
    resume value: (w at: each) value: each.
  ].
  "set up choices and rankings"
  mchoice at: 1 at: 1 put: 1. mchoice at: 1 at: 2 put: 2. mchoice at: 1 at: 3 put: 3.
  mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 3. mchoice at: 2 at: 3 put: 2.
  mchoice at: 3 at: 1 put: 2. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 1.
  wrank at: 1 at: 1 put: 2. wrank at: 1 at: 2 put: 1. wrank at: 1 at: 3 put: 3.
  wrank at: 2 at: 1 put: 2. wrank at: 2 at: 2 put: 3. wrank at: 2 at: 3 put: 1.
  wrank at: 3 at: 1 put: 1. wrank at: 3 at: 2 put: 2. wrank at: 3 at: 3 put: 3.
  "mchoice at: 1 at: 1 put: 2. mchoice at: 1 at: 2 put: 1. mchoice at: 1 at: 3 put: 3.
  mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 2. mchoice at: 2 at: 3 put: 3.
  mchoice at: 3 at: 1 put: 1. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 2.
  wrank at: 1 at: 1 put: 1. wrank at: 1 at: 2 put: 2. wrank at: 1 at: 3 put: 3.
  wrank at: 2 at: 1 put: 3. wrank at: 2 at: 2 put: 2. wrank at: 2 at: 3 put: 1.
  wrank at: 3 at: 1 put: 3. wrank at: 3 at: 2 put: 1. wrank at: 3 at: 3 put: 2."
  1 to: n do: [i |
    resume value: (m at: i) value: nil.
  ].
  "here output the solution held in wengagement"
  Transcript show: wengagement.
  1.
  1 to: n do: [each |

```

Transcript

```

#2 3 1

```

També hem provat un altre exemple que hem trobat a Internet i hem obtingut novament un aparellament estable pel grup:  $wengagement = (2\ 1\ 3)$ , és a dir, la dona 1 (María) amb l'home 2 (Marco), la dona 2 (Ana) amb l'home 1 (Carlos) i la dona 3 (Lucía) amb l'home 3 (Juan).

**Ejemplo 1.2.1.** Sean María, Ana, Lucía el conjunto de mujeres y Carlos, Marco, Juan el conjunto de hombres y sean las matrices de preferencias las que se muestran a continuación. Obtener un emparejamiento estable  $M$ .

María	Carlos	Marco	Juan
Ana	Juan	Marco	Carlos
Lucía	Juan	Carlos	Marco

*Matriz de preferencias de las mujeres*

Carlos	Ana	María	Lucía
Marco	María	Ana	Lucía
Juan	María	Lucía	Ana

*Matriz de preferencias de los hombres*

The screenshot shows the PharC workspace with a script defining preference matrices and finding a stable matching. The transcript window shows the output of the script, which is the stable matching  $\#(2\ 1\ 3)$ .

```

x - □ Workspace
In w m mchoice wrank wengagement corutinaMain
n := 3.
w := Array new: n.
m := Array new: n.
mchoice := Matrix rows: 3 columns: 3.
wrnk := Matrix rows: 3 columns: 3.
wengagement := Array new: n.
corutinaMain := Coroutine maker: [ resume :value ]
    "Transcript show: 'This is corutinaMain'; cr."
    "for para que las corutinas tengan su indice en value"
    1 to: n do: [each |
        resume value: (m at: each) value: each.
        resume value: (w at: each) value: each.
        l.
        "set up choices and rankings"
        "mchoice at: 1 at: 1 put: 1. mchoice at: 1 at: 2 put: 2. mchoice at: 1 at: 3 put: 3.
        mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 3. mchoice at: 2 at: 3 put: 2.
        mchoice at: 3 at: 1 put: 2. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 1.
        wrnk at: 1 at: 1 put: 2. wrnk at: 1 at: 2 put: 1. wrnk at: 1 at: 3 put: 3.
        wrnk at: 2 at: 1 put: 2. wrnk at: 2 at: 2 put: 3. wrnk at: 2 at: 3 put: 1.
        wrnk at: 3 at: 1 put: 1. wrnk at: 3 at: 2 put: 2. wrnk at: 3 at: 3 put: 3."
        mchoice at: 1 at: 1 put: 2. mchoice at: 1 at: 2 put: 1. mchoice at: 1 at: 3 put: 3.
        mchoice at: 2 at: 1 put: 1. mchoice at: 2 at: 2 put: 2. mchoice at: 2 at: 3 put: 3.
        mchoice at: 3 at: 1 put: 1. mchoice at: 3 at: 2 put: 3. mchoice at: 3 at: 3 put: 2.
        wrnk at: 1 at: 1 put: 1. wrnk at: 1 at: 2 put: 2. wrnk at: 1 at: 3 put: 3.
        wrnk at: 2 at: 1 put: 3. wrnk at: 2 at: 2 put: 2. wrnk at: 2 at: 3 put: 1.
        wrnk at: 3 at: 1 put: 3. wrnk at: 3 at: 2 put: 1. wrnk at: 3 at: 3 put: 2.
        1 to: n do: [i |
            resume value: (m at: i) value: nil.
            l.
            "here output the solution held in wengagement"
            Transcript show: wengagement.
            l.
        ]
    ]
1 to: n do: [each |

```

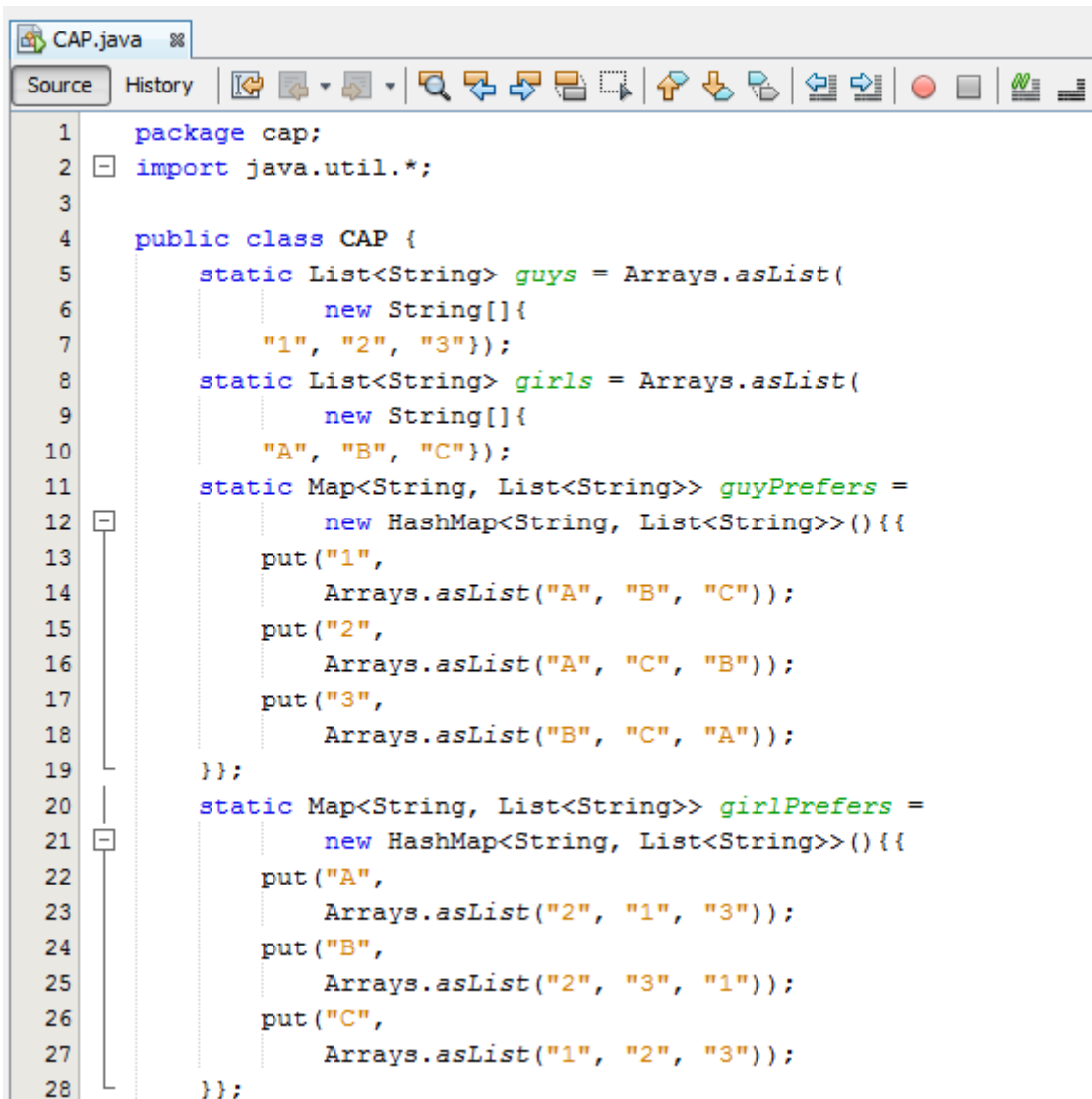
Transcript

```

#(2 1 3)

```

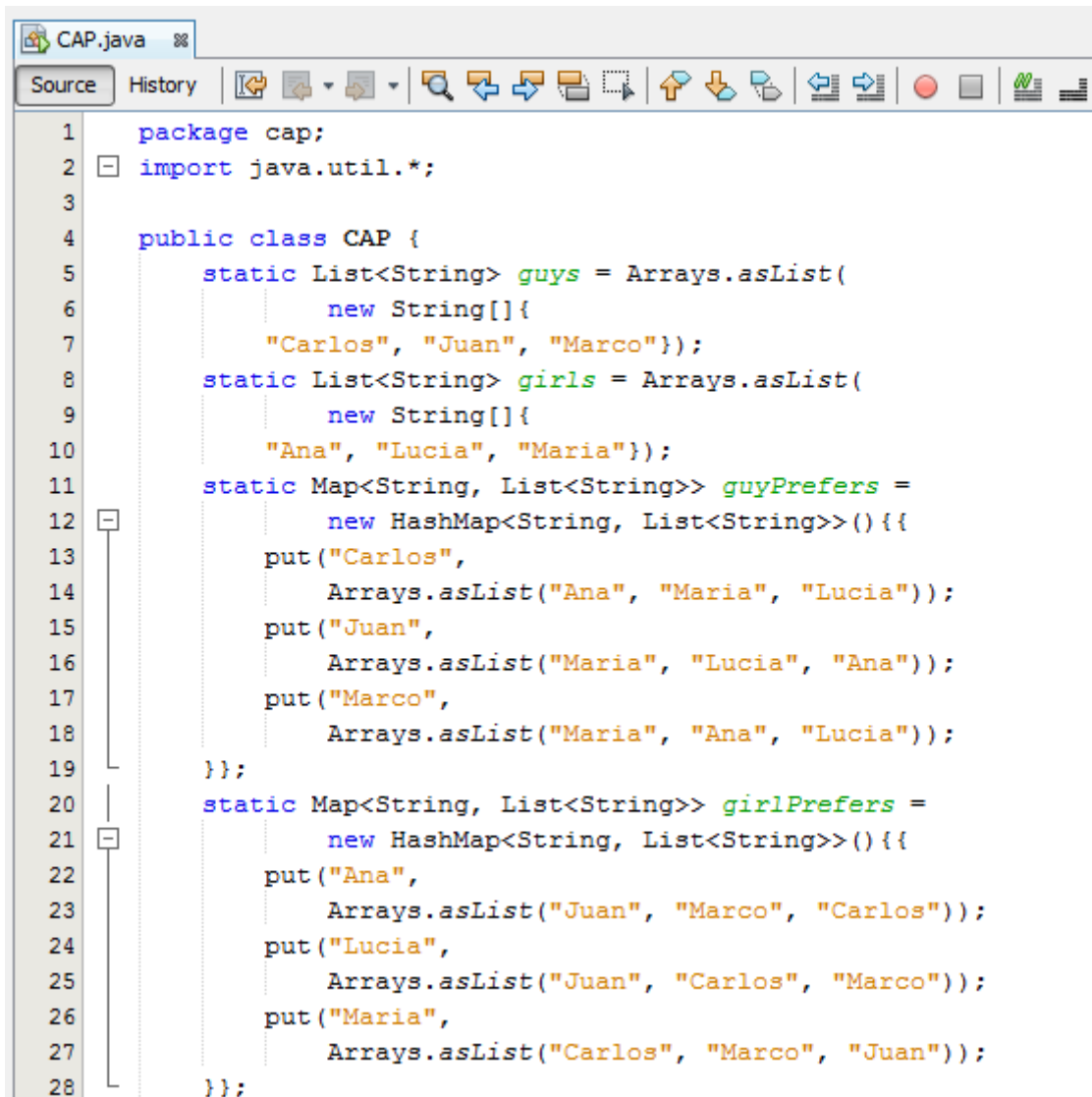
Per estar encara més convençuts que el codi és correcte, podem intentar demostrar que els aparellaments anteriors són inestables (mirant de trobar un home i una dona que no estiguin aparellats i que es prefereixen l'un a l'altre més que a la seva parella) i veurem que no ho són pas, o també podem agafar una implementació externa en Java, executar-la primer amb un *input* i després amb l'altre i comprovar que ambdues sortides es corresponen amb les que hem obtingut al Transcript.



```
1 package cap;
2 import java.util.*;
3
4 public class CAP {
5     static List<String> guys = Arrays.asList(
6         new String[]{
7             "1", "2", "3"});
8     static List<String> girls = Arrays.asList(
9         new String[]{
10            "A", "B", "C"});
11     static Map<String, List<String>> guyPrefers =
12         new HashMap<String, List<String>>() {{
13             put("1",
14                 Arrays.asList("A", "B", "C"));
15             put("2",
16                 Arrays.asList("A", "C", "B"));
17             put("3",
18                 Arrays.asList("B", "C", "A"));
19         }};
20     static Map<String, List<String>> girlPrefers =
21         new HashMap<String, List<String>>() {{
22             put("A",
23                 Arrays.asList("2", "1", "3"));
24             put("B",
25                 Arrays.asList("2", "3", "1"));
26             put("C",
27                 Arrays.asList("1", "2", "3"));
28         }};
```

Output - CAP (run)

```
run:
A is engaged to 2
B is engaged to 3
C is engaged to 1
Marriages are stable
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
1 package cap;
2 import java.util.*;
3
4 public class CAP {
5     static List<String> guys = Arrays.asList(
6         new String[]{
7             "Carlos", "Juan", "Marco"});
8     static List<String> girls = Arrays.asList(
9         new String[]{
10             "Ana", "Lucia", "Maria"});
11     static Map<String, List<String>> guyPrefers =
12         new HashMap<String, List<String>>() {{
13             put("Carlos",
14                 Arrays.asList("Ana", "Maria", "Lucia"));
15             put("Juan",
16                 Arrays.asList("Maria", "Lucia", "Ana"));
17             put("Marco",
18                 Arrays.asList("Maria", "Ana", "Lucia"));
19         }};
20     static Map<String, List<String>> girlPrefers =
21         new HashMap<String, List<String>>() {{
22             put("Ana",
23                 Arrays.asList("Juan", "Marco", "Carlos"));
24             put("Lucia",
25                 Arrays.asList("Juan", "Carlos", "Marco"));
26             put("Maria",
27                 Arrays.asList("Carlos", "Marco", "Juan"));
28         }};
```

Output - CAP (run)

```
run:
Ana is engaged to Carlos
Lucia is engaged to Juan
Maria is engaged to Marco
Marriages are stable
BUILD SUCCESSFUL (total time: 0 seconds)
```



## 5. Bibliografia

- ALLISON, Lloyd. *Stable marriages by coroutines* [en línia]. Article. Austràlia: University of Western Australia, 1983.

<[http://www.academia.edu/26551802/Stable\\_marriages\\_by\\_coroutines](http://www.academia.edu/26551802/Stable_marriages_by_coroutines)>

[Consulta: 24 gener 2017]

- BERGEL, A.; CASSOU, D.; DUCASSE, S.; LAVAL, J. *Deep into Pharo* [en línia]. Primera edició. Suïssa: Square Bracket Associates, 2013. ISBN 978-3-9523341-6-4.

<<http://files.pharo.org/books-pdfs/deep-into-pharo/2013-DeepIntoPharo-EN.pdf>>

[Consulta: 24 gener 2017]

- BLANCO PEÑA, Marina. *El problema de los matrimonios estables y otros problemas de emparejamiento* [en línia]. Treball de fi de grau. La Rioja: Universidad de La Rioja, 2015.

<[http://biblioteca.unirioja.es/tfe\\_e/TFE001024.pdf](http://biblioteca.unirioja.es/tfe_e/TFE001024.pdf)>

[Consulta: 24 gener 2017]

- DELGADO, Jordi. *Introducció a les continuacions a Smalltalk*. Material de suport de l'assignatura CAP. Barcelona: Facultat d'Informàtica de Barcelona, 2016.

[Consulta: 24 gener 2017]

- DELGADO, Jordi. *Introducció a Smalltalk*. Material de suport de l'assignatura CAP. Barcelona: Facultat d'Informàtica de Barcelona, 2016.

[Consulta: 24 gener 2017]

- *Stable marriage problem* [en línia].

<[https://rosettacode.org/wiki/Stable\\_marriage\\_problem](https://rosettacode.org/wiki/Stable_marriage_problem)>

[Consulta: 24 gener 2017]