

Unitat 1: Activitat Pràctica d'Avaluació Contínua

L'activitat pràctica d'avaluació contínua de la primera unitat consisteix en un parell d'exercicis, que aniran englobats dins el mateix paquet Java `com.ieseljust.psp.apac1`.

Creeu una estructura de carpetes com la següent, seguint els estàndards de java:

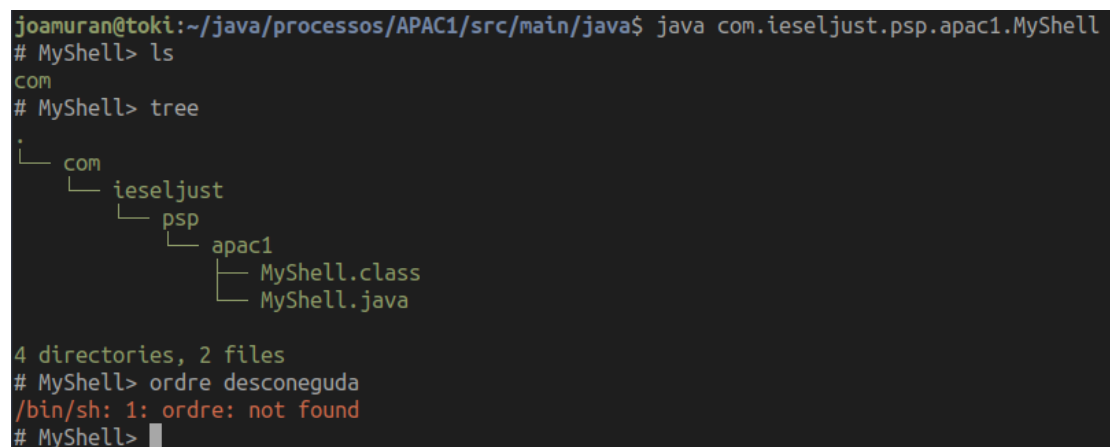
```
1 .
2 +-- src
3     +-- main
4         +-- java
5             +-+ com
6                 +-- ieseljust
7                     +-- psp
8                         +-- apac1
```

Tots els fonts (.java) aniran ubicats dins la carpeta *apac1*, i correspondran al paquet `com.ieseljust.psp.apac1`. És a dir:

- La primera línia de cada font serà `package com.ieseljust.psp.apac1`.
- A més, per executar cada aplicació, des del directori `src/main/java`, farem `java com.ieseljust.psp.NomDeLaClasse`.

Exercici 1. MyShell.

Creeu una classe anomenada `MyShell`, que siga un intèrpret d'ordres de Bash. El resultat serà el següent:



```
joamuran@toki:~/java/processos/APAC1/src/main/java$ java com.ieseljust.psp.apac1.MyShell
# MyShell> ls
com
# MyShell> tree
.
├── com
│   └── ieseljust
│       └── psp
│           └── apac1
│               ├── MyShell.class
│               └── MyShell.java
4 directories, 2 files
# MyShell> ordre desconeguda
/bin/sh: 1: ordre: not found
# MyShell> █
```

Figura 1: MyShell

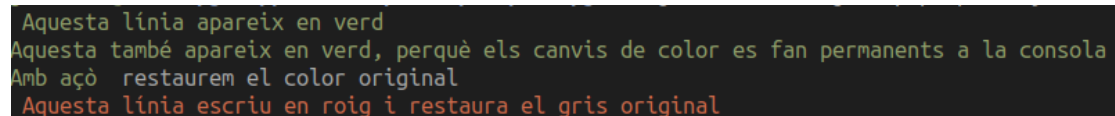
El funcionament de l'aplicació és el següent:

- Escriu un prompt # `MyShell>`, i està a l'espera que introduïm alguna ordre.
- Quan s'introdueix alguna ordre, llança un procés que executa aquesta, capturant el resultat, i mostrant-lo per pantalla, tenint en compte el següent:
 - Si el resultat és correcte (eixida estàndard del procés), l'eixida la mostra en verd.
 - Si el resultat no és correcte (eixida d'error del procés), l'eixida es mostra en roig.
 - Cal tindre en compte que algunes ordres de tipus builtin (aquelles interpretades directament per bash) ¹, no donaran error, però no s'executaran. No és l'objectiu d'aquesta pràctica la implementació d'aquestes funcionalitats, sinó la gestió de processos, pel que aquest comportament es dona per correcte.
 - El programa eixirà quan detecte l'ordre `quit` com a entrada.

Per tal de *pintar* l'eixida per pantalla, no cal més que especificar el color als mateixos `println` en forma de cadena de text. Per exemple:

```
1 System.out.println("\u001B[32m Aquesta línia apareix en roig");
2 System.out.println("Aquesta també apareix en roig, perquè els
   canvis de color es fan permanents a la consola");
3 System.out.println("Amb açò \u001B[0m restaurem el color original
   ");
4 System.out.println("\u001B[31m Aquests línies escvriu en verd i
   restaura el gris original \u001B[0m");
```

Obté el següent resultat:



```
Aquesta línia apareix en verd
Aquesta també apareix en verd, perquè els canvis de color es fan permanents a la consola
Amb açò restaurem el color original
Aquesta línia escriu en roig i restaura el gris original
```

Figura 2: MyShell

Exercici 2. Programació multiprocés.

Creeu un programa multiprocés que obtinga el valor màxim dels elements d'un vector generat aleatòriament, de longitud indicada per l'usuari com a únic argument, i aprofitant totes les capacitats multiprocés de la màquina (tots els processadors disponibles).

Per exemple, la següent crida:

¹<http://manpages.ubuntu.com/manpages/bionic/man7/bash-builtins.7.html>

```
1 java com.ieseljust.psp.ObteMaximMultiproces 1000
```

Obtindrà el màxim número d'un vector del 1.000 elements generats aleatòriament.

Consideracions i ajudes:

- Caldrà desenvolupar un programa a banda que calcule el màxim d'una sèrie de valors especificats com a arguments. (`java com.ieseljust.psp.calculaMaxim 10 3123 123 99929 213 432 123 123`). Aquestes quantitats seràn enters llargs, entre 0 i el valor màxim per a aquest tipus, representat per `Long.MAX_VALUE`. Tingueu en compte que els arguments d'un programa en Java es representen com a un vector d'Strings, pel que caldrà fer les corresponents conversions quan siga necessari.
- Caldrà dividir la tasca **en tants processos com processadors tinga l'ordinador** (haurem d'utilitzar la classe `Runtime` per a això). Si utilitzeu una màquina virtual, tingueu en compte que aquesta només té un processador assignat. Podeu afegir-li'n en la configuració d'aquesta.
- Aquesta divisió del problema, implicarà la divisió del vector original en tantes parts com processos anem a llançar. En aquest punt segurament necessitem crear nous vectors que siguin subvectors de l'original. Per a això, ens pot ser útil el mètode **System.arraycopy**: `System.arraycopy(arrayOriginal, posició_inicial_en_array_original, arrayOnVaLaCopia, posicio_inicial_al_vector_copia_generalment_0, numero_elements_a_copiar);`
- Tal i com hem fet a l'exercici del sumatori, caldrà esperar les respostes de cada procés llançat i calcular finalment el màxim de tots. Aquest últim càlcul pot fer-se a la mateixa classe principal o llençant un nou procés altre procés.
- Alguns mètodes que ens poden resultar útils són:

```
1 /* Donat un ArrayList d'strings */
2 ArrayList<String> el_meu_arraylist = new ArrayList<String>();
3 /* Podem afegir elements simples: */
4 params.add("java");
5 /* O concatenar-li el contingut d'un altre vector*/
6 params.addAll(Arrays.asList(args));
```

```
1 /* La següent expressió obté un enter llarg aleatori
2     entre 0 i el valor especificat amb MAX
3     i el converteix a String */
4 String.valueOf((long) (Math.random() * MAX) + 1);
```

Un possible exemple de l'eixida de l'aplicació seria el següent:

```
1 $ java com.ieseljust.psp.apac1.ObteMaximMultiproces 100000
2 Dividint la tasca en 4 processos
3 Llançant el procés 0 per fer càlculs de 0 fins a 25000
4 Llançant el procés 1 per fer càlculs de 25001 fins a 50001
5 Llançant el procés 2 per fer càlculs de 50002 fins a 75002
6 Llançant el procés 3 per fer càlculs de 75003 fins a 99999
7 S'han finalitzat els dos processos. Esperant els búffers.
8 Esperant al buffer 0
9 Esperant que es plene el buffer
10 Màxim parcial del procés 0=9221975700441651201
11 Esperant al buffer 1
12 Esperant que es plene el buffer
13 Màxim parcial del procés 1=9223198816394112001
14 Esperant al buffer 2
15 Esperant que es plene el buffer
16 Màxim parcial del procés 2=9222854824348106753
17 Esperant al buffer 3
18 Esperant que es plene el buffer
19 Màxim parcial del procés 3=9222605503125530625
20 Màxim valor del vector: 9223198816394112001
```

En aquest exemple, s'ha generat un vector de 100.000 enters llargs aleatoris, entre 0 i el màxim valor representable. En aquest cas, l'ordinador consta de quatre nuclis, pel que s'ha dividit la tasca en quatre processos: el primer busca el màxim entre les posicions 0 i 25000 del vector, el segon entre la 25001 i la 50001, el tercer entre la 50002 i la 75002, i el quart entre la 75003 i la 99999. Com veiem, s'obtenen quatre resultats, i finalment, es mostra el màxim de tots ells.