

ACTIVITAT
Objectius: <ul style="list-style-type: none">- Saber com definir objectes Singleton i com ignorar aquest patró
Instruccions: <ul style="list-style-type: none">- Es tracta d'un treball individual, no s'admet cap tipus de còpia.- Responen a l'espai de cada pregunta, si ho feu amb diapositives enganxeu la diapositiva en aquest mateix espai.- Es valorarà la cura en la presentació del document i que segueixi l'estructura indicada.
Criteris d'avaluació: <ul style="list-style-type: none">- Cada pregunta té el mateix pes sobre 90%- Les metodologies de treball, organització personal i participació contenen un 10%
Entrega: <ul style="list-style-type: none">- Aquest document amb les explicacions i captures necessàries i els arxius adjunts necessaris del codi que es demana- El nom dels arxius adjunts a entregar serà: nomicognom-nomicognom.zip

Noms i Cognoms: [Joel Berzal Álamo](#)

Materials:

Necessiteu un entorn de desenvolupament en JAVA

Feu servir Google per buscar els tutorials que us serveixin millor

Creeu els arxius a la carpeta 'src' del projecte i executeu amb els scripts './build.sh' i './build.ps1'

Tasques:

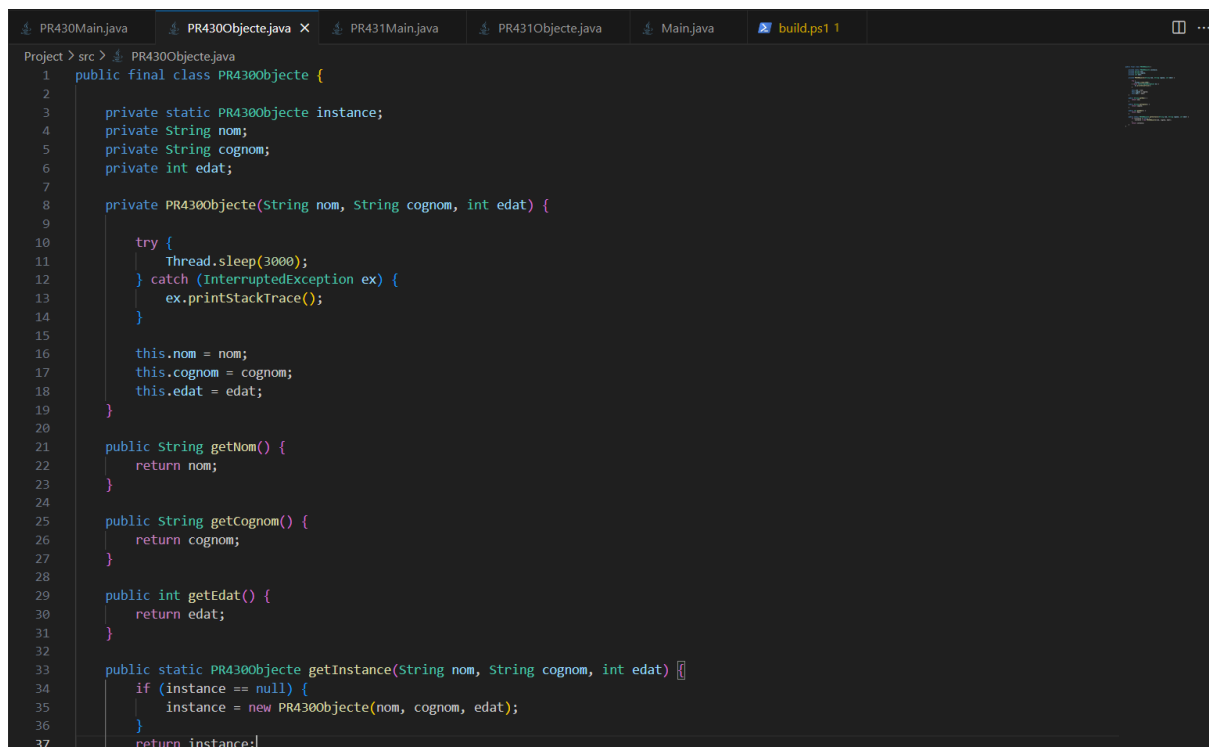
- **Exercici 0** - Crea un programa “PR430Main.java” que instanciï 3 objectes amb dades diferents de la classe “PR430Objecte.java” amb 3 segons de diferència.

L’objecte “PR430Objecte” ha de tenir les variables privades ‘nom’, ‘cognom’, ‘edat’ com a privades NO estàtiques i ha de seguir el model Singleton.

Mostra les dades de cada instància al final (caldrà sobreescriure toString).

Per poder dur a terme aquest exercici, primer he hagut de definir la classe “PR430Objecte.java”, la qual està conformada per diversos elements:

- Tres variables privades i no estàtiques anomenades ‘nom’, ‘cognom’ i ‘edat’.
- Un constructor privat per “PR430Objecte.java”, el qual instancia les tres variables anteriors i conté un Thread que fa que, a l’hora d’instanciar un objecte d’aquesta mateixa classe, hi hagi tres segons de diferència.
- Tres getters públics per les tres variables anteriors.
- Un mètode públic i estàtic de la classe “PR430Objecte.java” anomenat ‘getInstance’, el qual té dues variables de tipus String (‘nom’ i ‘cognom’) i una de tipus Integer (‘edat’). Aquest mètode s’encarrega d’instanciar un nou objecte de la classe “PR430Objecte.java” en cas que el valor de la instància sigui NULL.



```
1 public final class PR430Objecte {
2
3     private static PR430Objecte instance;
4     private String nom;
5     private String cognom;
6     private int edat;
7
8     private PR430Objecte(String nom, String cognom, int edat) {
9
10        try {
11            Thread.sleep(3000);
12        } catch (InterruptedException ex) {
13            ex.printStackTrace();
14        }
15
16        this.nom = nom;
17        this.cognom = cognom;
18        this.edat = edat;
19    }
20
21    public String getNom() {
22        return nom;
23    }
24
25    public String getCognom() {
26        return cognom;
27    }
28
29    public int getEdat() {
30        return edat;
31    }
32
33    public static PR430Objecte getInstance(String nom, String cognom, int edat) {
34        if (instance == null) {
35            instance = new PR430Objecte(nom, cognom, edat);
36        }
37        return instance;
38    }
39 }
```

Finalment, he hagut de definir la classe “PR430Main.java”, la qual té un únic mètode públic i estàtic anomenat ‘main’, el qual s’encarrega de crear tres instàncies idèntiques i, un cop fet això imprimir-les per pantalla.

```

PR430Main.java X PR430Objecte.java PR431Main.java PR431Objecte.java Main.java build.ps1 1
Project > src > PR430Main.java
1 import java.lang.reflect.Constructor;
2
3 public class PR430Main {
4
5     public static void main(String[] args) {
6
7         System.out.println("\nIniciant 0");
8         PR430Objecte instance_0 = PR430Objecte.getInstance("Manel", "Polar", 18);
9
10        System.out.println("Iniciant 1");
11        PR430Objecte instance_1 = PR430Objecte.getInstance("Manel", "Polar", 18);
12
13        System.out.println("Iniciant 2");
14        PR430Objecte instance_2 = PR430Objecte.getInstance("Manel", "Polar", 18);
15
16        System.out.println("Nom: " + instance_0.getNom() + ", Cognom: " + instance_0.getCognom() + ", Edat: " + instance_0.getEdat());
17        System.out.println("Nom: " + instance_1.getNom() + ", Cognom: " + instance_1.getCognom() + ", Edat: " + instance_1.getEdat());
18        System.out.println("Nom: " + instance_2.getNom() + ", Cognom: " + instance_2.getCognom() + ", Edat: " + instance_2.getEdat());
19    }
20 }
21

```

- **Exercici 1** - Crea un programa “PR431Main.java” que instanciï 3 objectes amb dades diferents de la classe “PR431Objecte.java” amb 3 segons de diferència i **aconseguint 3 instàncies diferents**, ignorant el fet que es tracta d’un objecte que implementa el model ‘Singleton’.

L’objecte “PR431Objecte” ha de tenir les variables privades ‘nom’, ‘cognom’, ‘edat’ com a privades NO estàtiques i ha de seguir el model Singleton (com l’exercici anterior).

Mostra les dades de cada instància al final (caldrà sobreesciure toString).

Pots crear una funció ‘getNewDestroyedInstance’ que retorni una instància ‘hackejada’ de Singleton, per no anar repetint codi.

Per poder dur a terme aquest exercici, primer he hagut de definir la classe “PR431Objecte.java”, la qual està conformada per diversos elements:

- Tres variables privades i no estàtiques anomenades ‘nom’, ‘cognom’ i ‘edat’.
- Un constructor privat per “PR431Objecte.java”, el qual instancia les tres variables anteriors i conté un Thread que fa que, a l’hora d’instanciar un objecte d’aquesta mateixa classe, hi hagi tres segons de diferència.
- Tres getters públics per les tres variables anteriors.
- Un mètode públic i estàtic de la classe “PR431Objecte.java” anomenat ‘getInstance’, el qual té dues variables de tipus String (‘nom’ i ‘cognom’) i una de tipus Integer (‘edat’). Aquest mètode s’encarrega d’instanciar un nou objecte de la classe “PR431Objecte.java” en cas que el valor de la instància sigui NULL.

```

1 public final class PR431Objecte {
2
3     private static PR431Objecte instance;
4     private String nom;
5     private String cognom;
6     private int edat;
7
8     private PR431Objecte(String nom, String cognom, int edat) {
9
10        try {
11            Thread.sleep(3000);
12        } catch (InterruptedException ex) {
13            ex.printStackTrace();
14        }
15
16        this.nom = nom;
17        this.cognom = cognom;
18        this.edat = edat;
19    }
20
21    public String getNom() {
22        return nom;
23    }
24
25    public String getCognom() {
26        return cognom;
27    }
28
29    public int getEdat() {
30        return edat;
31    }
32
33    public static PR431Objecte getInstance(String nom, String cognom, int edat) {
34        if (instance == null) {
35            instance = new PR431Objecte(nom, cognom, edat);
36        }
37        return instance;
38    }
39

```

Finalment, he hagut de definir la classe “PR431Main.java”, la qual té dos mètodes:

- Un mètode públic i estàtic anomenat ‘main’, el qual s’encarrega de crear tres instàncies diferents i, un cop fet això imprimir-les per pantalla.
- Un mètode estàtic anomenat ‘getNewDestroyedInstance’, el qual s’encarrega d’ignorar el patró Singleton per tal de poder obtenir instàncies diferents sense haver d’anar repetint codi.

```

1 import java.lang.reflect.Constructor;
2
3 public class PR431Main {
4
5     public static void main(String[] args) {
6
7         System.out.println("\nIniciant 0");
8         PR431Objecte instance_0 = PR431Objecte.getInstance("Manel", "Polar", 18);
9
10        System.out.println("Iniciant 1");
11        PR431Objecte instance_1 = getNewDestroyedInstance("Laura", "Gelada", 19);
12
13        System.out.println("Iniciant 2");
14        PR431Objecte instance_2 = getNewDestroyedInstance("Pingu", "Ice", 22);
15
16        System.out.println("Nom: " + instance_0.getNom() + ", Cognom: " + instance_0.getCognom() + ", Edat: " + instance_0.getEdat());
17        System.out.println("Nom: " + instance_1.getNom() + ", Cognom: " + instance_1.getCognom() + ", Edat: " + instance_1.getEdat());
18        System.out.println("Nom: " + instance_2.getNom() + ", Cognom: " + instance_2.getCognom() + ", Edat: " + instance_2.getEdat());
19    }
20
21    static PR431Objecte getNewDestroyedInstance (String nom, String cognom, int edat) {
22
23        PR431Objecte result = null;
24        try {
25            Constructor<>[] constructors = PR431Objecte.class.getDeclaredConstructors();
26            for (Constructor<> constructor : constructors) {
27                constructor.setAccessible(true);
28                result = (PR431Objecte) constructor.newInstance(nom, cognom, edat);
29                break;
30            }
31        } catch (Exception e) {
32            e.printStackTrace();
33        }
34        return result;
35    }
36

```