ACTIVITAT
Objectius: - Saber com definir objectes Singleton i com ignorar aquest patró
Instruccions: - Es tracta d'un treball individual, no s'admet cap tipus de còpia. - Responeu 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: - Cada pregunta té el mateix pes sobre 90% - Les metodologies de treball, organització personal i participació conten un 10%
Entrega: - 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 seràn: nomicognom-nomicognom.zip

Noms i Cognoms:

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:

- Preparació - Crea un arxiu 'Main.java' amb un menú per cridar cada un dels altres arxius amb funció 'main' d'aquesta activitat. Aquí tens un exemple que hauras d'adaptar al què demana l'enunciat de cada exercici:

```
import java.io.IOException;
import java.util.*;

public class Main {

  static Scanner in = new Scanner(System.in); // System.in és global

  // Main
  public static void main(String[] args) throws InterruptedException, IOException {
    boolean running = true;
    while (running) {
      String menu = "Escull una opció:";
      menu = menu + "\n 0) PR430Main";
      menu = menu + "\n 1) PR431Main";
      // Adapta aquí les altres classes de l'exercici (PR432Main…)
      menu = menu + "\n 100) Sortir";
      System.out.println(menu);

      int opcio = Integer.valueOf(llegirLinia("Opció:"));
      try {
        switch (opcio) {
          case 0: PR430Main.main(args); break;
          case 1: PR431Main.main(args); break;
          // Adapta aquí les altres classes de l'exercici (PR432Main…)
          case 100: running = false; break;
          default: break;
        }
      } catch (Exception e) {
          System.out.println(e);
      }
    }
    in.close();
  }

  static public String llegirLinia (String text) {
    System.out.print(text);
    return in.nextLine();
  }
}
```

- Exercici 0 - Crea un programa "PR430Main.java" que instancii 3 objectes amb dades diferents de la classe "PR430Objecte.java" amb 3 segons de diferència.

```java
public class PR430Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Iniciant 0");
        PR430Objecte instance1 = PR430Objecte.getInstance(nom:"Manel", cognom:"Polar", edat:"18");
        PR430Objecte instance2 = PR430Objecte.getInstance(nom:"Laura", cognom:"Gelada", edat:"19");
        PR430Objecte instance3 = PR430Objecte.getInstance(nom:"Pingu", cognom:"Ice", edat:"22");

    }

}
```

Aqui esperaremos los 3 segundos.dentro del metodo que obtenemos el singelton

```java
public static PR430Objecte getInstance(String nom,String cognom, String edat){
    if (instance == null) {
        instance = new PR430Objecte(nom,cognom, edat);
    }
    try {
        Thread.sleep(millis:3000);
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
    return instance;
}
```

L'objecte "PR430Objecte" ha de tenir les variables privades 'nom', 'cognom', 'edat' com a privades NO estàtiques i ha de seguir el model Singleton.

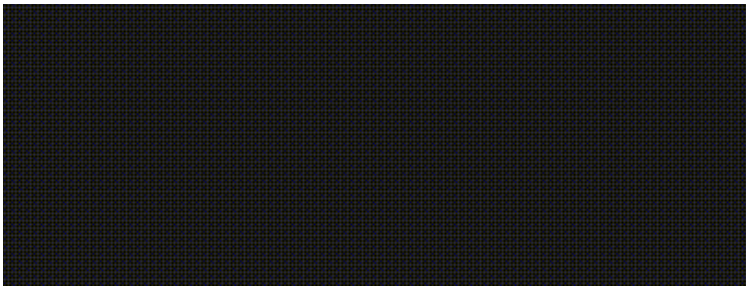
```java
public final class PR430Objecte{

    private static PR430Objecte instance;
    private String nom, cognom, edat;


    private PR430Objecte(String nom,String cognom, String edat){
            // Simulem una inicialització lenta
            this.nom = nom;
            this.cognom = cognom;
            this.edat = edat;

    }
```

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



```
-Singleton_cc9581fe\bin' 'PR430Main'
Iniciant 0
Iniciant 1
Iniciant 2
Nom: Manel    cognom: Polar  edat:18
Nom: Manel    cognom: Polar  edat:18
Nom: Manel    cognom: Polar  edat:18
PS D:\rzajr\Documents\DAM2-MPDUAL-PR41-Singleton>
```

- Exercici 1 - Crea un programa "PR431Main.java" que instancii 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'.

```java
public class PR431Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Iniciant 0");
        PR431Objecte instance1 = PR431Objecte.getInstance(nom:"Roberto", cognom:"Zambrano", edat:"22");
        System.out.println(x:"Iniciant 1");
        PR431Objecte instance2 = PR431Objecte.getNewDestroyedInstance(nom:"Miguel", cognom:"Carrasco", edat:"19");
        System.out.println(x:"Iniciant 2");
        PR431Objecte instance3 = PR431Objecte.getNewDestroyedInstance(nom:"Ahmed", cognom:"Jalil", edat:"11");
        System.out.println(instance1.toString());
        System.out.println(instance2.toString());
        System.out.println(instance3.toString());
    }

}
```

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).

```java
public final class PR431Objecte{

    private static PR431Objecte instance;
    private String nom, cognom, edat;


    private PR431Objecte(String nom,String cognom, String edat){
            // Simulem una inicialització lenta
            this.nom = nom;
            this.cognom = cognom;
            this.edat = edat;

    }
```

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

```java
@Override
public String toString() {
    return "Nom: "+nom+"    cognom: "+cognom+"  edat:"+edat;
}
```

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

```
    static PR431Objecte getNewDestroyedInstance (String nom,String cognom, String edat) {

        PR431Objecte result = null;
        try {
            Constructor<?>[] constructors = PR431Objecte.class.getDeclaredConstructors();
            for (Constructor<?> constructor : constructors) {
                //Below code will destroy the singleton pattern
                constructor.setAccessible(flag:true);
                result = (PR431Objecte) constructor.newInstance(nom,cognom,edat);
                break;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            Thread.sleep(millis:3000);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
        return result;
    }

}
```

```
UAL-PR41-Singleton_cc9581fe\bin' 'PR431Main'
Iniciant 0
Iniciant 1
Iniciant 2
Nom: Roberto    cognom: Zambrano  edat:22
Nom: Miguel    cognom: Carrasco  edat:19
Nom: Ahmed    cognom: Jalil  edat:11
PS D:\rzajr\Documents\DAM2-MPDUAL-PR41-Singleton>
```