Crear una **Aplicación Java** llamada **GestionEmpleados** que:

- **Defina** el interfaz **`IEmpleado`** que declare el método:

  ```
  public void mostrarInfo();
  ```

- **Defina** la **clase** JavaBean **`Empleado`** que:
  - **Defina y encapsule** de forma que **no pueda heredarse**, un atributo llamado **`cargo`** que contenga el **cargo del empleado** que representa el objeto actual. Los **únicos valores posibles** para este atributo son **"Operario"** o **"Encargado"**, aunque en general su valor será **"Operario"**.
  - **Defina** sendos constructores predeterminado y parametrizado.
  - Utilice un método de la librería **UtilesEmpleado** llamado **validarCargo** para validar el **cargo** de cada empleado.
  - **Implemente** el interfaz **`IEmpleado`** de forma que el método **`mostrarInfo`** muestre por pantalla, accediendo al **reloj del sistema**, información relativa al usuario actual en un formato similar a:

    ```
    "Informe .....: 08:20h"
    "Cargo .......: Encargado"
    ```

- **Defina** la clase JavaBean **`Operario`** derivada de la clase **`Empleado`** que:
  - No pueda tener **clases derivadas**.
  - **Establezca** su cargo como **`"Operario"`**.
  - **Defina y encapsule** un atributo llamado **`seccion`** que contiene el nombre de la sección asignada. Los valores posibles son **`"Producción"`**, **`"Almacén"`** y **`"Mantenimiento"`**. La sección predeterminada será **`"Producción"`**.
  - Utilice un método de la librería **UtilesEmpleado** llamado **validarSeccion** para validar la **sección** de cada operario.
  - **Redefina** el método **mostrarInfo,** incluyendo la llamada al método heredado, para mostrar por pantalla una información similar a:

    ```
    "Informe .....: 09:15h"
    "Cargo .......: Operario"
    "Sección .....: Producción"
    ```

- **Defina** la clase JavaBean **`Encargado`** derivada de la clase **`Empleado`** que:
  - No Pueda tener **clases derivadas.**
  - **Establezca** su cargo como **`"Encargado"`**.
  - **Defina** una constante llamada **`OBJETIVO`** que representa el número de unidades que se deben de fabricar diariamente en la sección a su cargo y que está establecido en **`100`**.
  - **Defina y encapsule** un atributo llamado **`produccion`** que contiene el número de unidades que se llevan fabricadas. Cada jornada de producción empieza con un **stock remanente de la jornada anterior** que **CADA DÍA** es diferente, aleatorio e inferior a 10 unidades.
  - **Redefina** el método **mostrarInfo** para:
    - En caso de que la producción **SI consiga el objetivo** se mostrará por pantalla una información similar a:

      ```
      "Informe .....: 09:15h"
      "Cargo .......: Encargado"
      "Producción ..: 57/100 - Pendiente"
      ```

        - En caso de que la producción **NO consiga el objetivo** se mostrará por pantalla una información similar a:

```
"Informe .....: 09:15h"
"Cargo .......: Encargado"
"Producción ..: 103/100 - Conseguido"
```

- **Defina** la clase de librería llamada **`UtilesEmpleado`** que contenga aquellos recursos de programación que se utilizarán en las clases anteriores, tales como:
    - El método **`validarCargo`** que devuelve **`true`** si el **`cargo`** es **`"Operario"`** o **`"Encargado"`** y **`false`** en caso contrario.
    - El método **`validarSeccion`** que devuelve **`true`** si la **`seccion`** es **`"Producción"`**, **`"Almacén"`** o **`"Mantenimiento"`** y **`false`** en caso contrario.
    - El método **`validarProducción`** que devuelve **`true`** si el valor de la **`producción`** es **`mayor o igual a 0`** y **`false`** en caso contrario.
- **Cree** un objeto de la clase **`Operario`** y otro de la clase **Encargado** utilizando sus respectivos **constructores parametrizados** utilizando valores arbitrarios a discreción del alumno y por cada uno de ellos:
    - Mostrar la información relativa a cada uno de ellos llamando a sus respectivos métodos **`mostrarInfo`**.
    - Modificar arbitrariamente la **sección** en la que trabaja el **`Operario`** y la **producción** controlada por el **Encargado** llamando al respectivo **método mutador** de cada objeto.
    - Mostrar de nuevo la información relativa a cada uno de ellos llamando a sus respectivos métodos **`mostrarInfo`**.
    - Comparar ambos objetos mediante el método heredado **equals** considerando que ambos objetos son **empleados**, que se redefinirá para tener en cuenta el estado de los objetos a comparar, mostrando por consola un **mensaje informativo** adecuado.