Informática II - Prepa Tec Campus Eugenio Garza Lagüera
Laboratorio 2ndo Parcial

## Sección 1: Métodos Estáticos

**Problema 1: `Merlin`**

En ocasiones, queremos que una clase sólo permita tener una instancia activa en cualquier momento. Normalmente, estas clases se determinan como *Singletons*. Crea una clase `Merlin` que tenga un solo atributo, `theWizard` que sea de tipo `Merlin`. La clase tiene un constructor, y los siguientes dos métodos:

- `Merlin`: un constructor privado. Sólo el código dentro de la clase `Merlin` podrá invocar el constructor.
- `summon()`: un método estático que retorna el valor de la variable `theWizard`, siempre y cuando éste no sea `null`. Si `theWizard` es `null`, este método deberá crear una instancia de `Merlin` utilizando el constructor privado, así como asignarlo a la variable estática `theWizard`.
- `consult()`: este método de instancia deberá imprimir en la consola aleatoriamente (random) cualquiera de las siguientes frases:
    - I am Merlin
    - No one's ever needed to do it, as much as you do. This is your moment. Believe in yourself.
    - What exactly do you think I'm capable of?
    - Camelot isn't built on magic, but on people, on their faith.
    - I have walked my way since the beginning of time. Sometimes I give, sometimes I take, it is mine to know which and when!
    - Anál nathrach, orth' bháis's bethad, do chél dénmha.
    - Behold! The Sword of Power! Excalibur!
    - Remember, there's always something cleverer than yourself.

Utiliza el siguiente bloque de código para probar tu clase.

```
Merlin wizard = Merlin.summon();
wizard.consult();
wizard.consult();
wizard.consult();
Merlin.summon.consult();
```

**Problema Reto! Calculadora de Impuestos**

Diseña una clase que contendrá múltiples métodos estáticos para calcular operaciones de impuestos. La clase no deberá tener un constructor. Los atributos serán los siguientes:

- `basicRate`: una variable estática de tipo double que comience con un valor de 4%.
- `luxuryRate`: una variable estática de tipo double que comience con un valor de 10%

Los métodos serán:

- `computeCostBasic(price)`: un método estático que retorne el precio mas el impuesto básico, redondeado a dos decimales.
- `computeCostLuxury(price)`: un método estático que retorne el precio mas el impuesto a productos de lujo, redondeado a dos decimales.
- `changeBasicRateTo(newRate)`: un método estático que modifique el valor del impuesto básico.
- `changeLuxuryRateTo(newRate)`: un método estático que modifique el valor del impuesto a productos de lujo.
- `roundToNearestPenny(price)`: un método estático privado que retorne el precio redondeado a dos decimales. Por ejemplo, si el precio es 12.567, el método retornará 12.57.

## Sección 2: Herencia y Polimorfimo

**1. Genera un proyecto con un paquete llamado store. Descarga las clases Store.java y Product.java y StoreDemo.java de Canvas, e impórtalas a un proyecto nuevo con un paquete llamado "store".**
**2. Diseña una subclase llamada CoffeeShop que herede la clase Store. La nueva clase CoffeeShop debe tener los siguientes atributos y métodos:**

- `maxOccupancy` como una variable de tipo entero para representar el aforo máximo del restaurante.
- `menuItems` como un arreglo objetos de la clase Product que servirá para almacenar el nombre y los precios de cada uno de los productos en el menu.
- `menuItemsCount` como un entero que servirá para almacenar la cantidad de elementos que existen en el arreglo `menuItems`.
- `Método constructor` que reciba como parámetro de entrada los siguientes elementos:
  - `address`
  - `storeName`
  - `employeeCount`
  - `maxOccupancy`

  El arreglo `menuItems` lo instanciarás con espacio para almacenar 100 ítems. `menuItemsCount` comenzará con un valor de 0.
  Nota: recuerda que el constructor de la clase **CoffeeShop** debe hacer siempre una llamada al constructor de la clase base.
- Un segundo `método constructor` que no reciba parámetros de entrada, pero que utilice el método constructor 1 para instanciar el objeto con valores default.
  Nota: recuerda utilizar el método `this( )` para esto.
- `Método addMenuItem(Product item)` que agregará el producto recibido como parámetro al arreglo `menuItems`, e incrementará en 1 la variable `menuItemsCount`.
- Método `toString()` que retorne la información de todas las variables de instancia de la clase, incluidas aquellas variables que han sido heredadas.
  - `address`
  - `storeName`
  - `employeeCount`
  - `maxOccupancy`
  - ***cada elemento del menu y su respectivo precio***

  Recuerda que puedes reutilizar la implementación de `toString()` de la clase base llamando `super.toString()`.
- **Método** `openShop()` que imprima en pantalla tres líneas con la siguiente información:
  - Mensaje de bievenida a la tienda, que incluya el nombre de ésta.
  - El menú de los productos que vende.
- **Método** `closeShop()` que imprima en pantalla:
  - Mensaje de despedida de la tienda, que incluya el nombre de ésta.

**3. Utiliza la clase StoreDemo.java, adjunta en Canvas, para probar tu clase.**

- Declara un arreglo de 2 elementos con variables de tipo Store.
- En el primer elemento, instancia un objeto de la clase Store.
- En el segundo elemento, instancia un objeto de tipo CoffeeShop. Agrega 3 productos al menú.
- Para cada elemento del arreglo, prueba la impresión de la información del objeto, así como los métodos `openShop() y closeShop().`

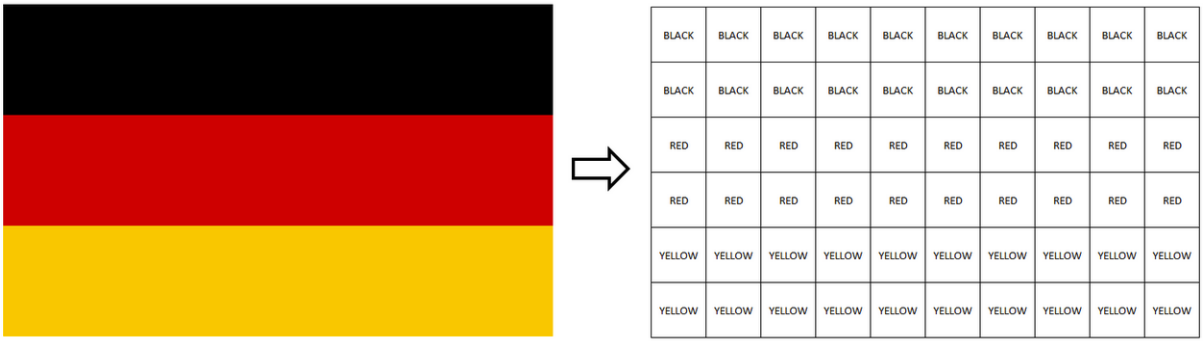
## Sección 3: Arreglos multidimensionales

Diseñe un método llamado `createGermanyFlag` que instancíe y retorne una matriz de Strings que sirvan para representar la bandera de Alemania. Cada elemento de la matriz almacenará el nombre del color con el que se debe colorear para darle vida a dicha bandera.

La bandera deberá generarse con una proporción de 3:5 (3 filas por cada 5 columnas). Adicionalmente, el parámetro de entrada size se multiplicará por las proporciones de la matriz para determinar su tamaño.

Ejemplo:

```
createGermanyFlag(2)
```




```
public static String[][] createGermanyFlag(int size) {

}
```