Taller de Lenguajes II
Práctica nº 5 – parte B

Temas

- Encapsulamiento
- Especificadores de acceso: private, public, protected, package (default)
- Palabra clave static y final
- UML

1. **Patrón Singleton.** Hay un número de casos donde se necesita asegurar que **NUNCA exista más de una instancia de una determinada clase** en la aplicación. A modo de ejemplo, cuando un sistema arranca su ejecución, los parámetros generales de configuración podrían levantarse en una ÚNICA instancia de la clase. Aquí aplica lo que se conoce como el **patrón Singleton**.

   - Implemente una clase llamada **CharlyGarcia** que cumpla con el patrón Singleton. La clase **CharlyGarcia** debe proveer una manera para acceder a esa única instancia.

     a. Escriba la clase **CharlyGarcia** (piense en los modificadores de acceso del constructor y en los calificadores java que tiene disponibles, para escribir la solución).

     b. Realice el Diagrama UML de su solución.

2. **Sobrescritura de métodos y especificadores de acceso.** Considerando el siguiente código:

```java
package unlp.info.reinoanimal;
abstract public class Animal {
        abstract protected void saludo();
}
```

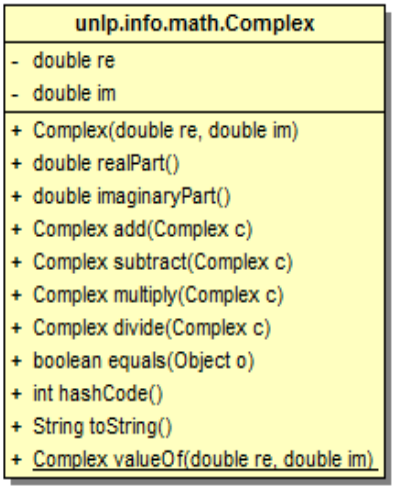
```java
package unlp.info.reinoanimal;
public class Gato extends Animal {
   @Override
   private void saludo() {
      System.out.println("Miau!");
   }
}
```

```java
package unlp.info.reinoanimal;
public class Perro extends Animal {
   @Override
   void saludo() {
      System.out.println("Guau!");
   }
}
```

I. Indique el motivo del error en compilación.
II. Indique qué valores de especificadores de acceso son válidos de acuerdo a las reglas del lenguaje

3. Clase Número Complejo

a) Cree una clase Número Complejo de acuerdo al diagrama UML especificado y teniendo en cuenta las siguientes consideraciones:



I. Considere que esta clase debe pertenecer al paquete "unlp.info.math"
II. Considere que por cuestiones de seguridad nadie debería sobreescribir su comportamiento
III. Sobreescriba los métodos **equals** y **toString**
IV. Implemente sólo las operaciones de suma y resta de números complejos.
V. Tenga en cuenta que el método **valueOf** es un método de clase que devuelve un número complejo

b) Cree una aplicación llamada **TestNumeroComplejo** que lea **desde la consola** 2 números complejos, y muestre el resultado de sumarlos y restarlos. **Esta operación se realizará indefinidamente.**

I. Desde el eclipse, exporte las clases TestNumeroComplejo y Complex como un archivo Test.jar
II. Ejecute la clase Test.jar y pruebe que funciona correctamente.