Taller de Lenguajes II
Práctica nº 5 – parte A

Temas

- Encapsulamiento
- Paquetes
- Especificadores de acceso: private, public, protected, default
- Palabra clave static y final

1. **Especificadores de acceso.** Analice las siguientes clases, pruebe en eclipse y responda cada uno de los incisos que figuran a continuación.

a) Considere la siguiente clase **Alpha**. ¿Es válido el acceso de la clase Gamma? **JUSTIFIQUE**

```
package  griego;
class Alpha {
        protected int x;
        protected void otroMetodoA(){
              System.out.println("Un método protegido");
        }
}

package  griego;
class Gamma {
         void unMétodoG(){
           Alpha a = new Alpha();
           a.x=10;
           a.otroMetodoA();
        }
 }
```

b) Modifique la clase Alpha como se indica debajo. ¿Es válido el método de la clase Beta? **JUSTIFIQUE**

```
package  griego;
public class Alpha {
 int x;
 void unMetodoA(){
  System.out.println("Un mét. paquete");
 }
}

package romano;
import griego.*;
class Beta {
       void unMetodoB(){
              Alpha a = new Alpha();
              a.x=10;
              a.unMetodoA();
        }
   }
```

c) Analice el método de la clase **Delta**. ¿Es válido? **JUSTIFIQUE** analizando cómo influye el control de acceso **protected** en la herencia de clases.

```
package  griego;
public class Alpha {
        protected int x;
        protected void otroMetodoA(){
              System.out.println("Un método protegido");
        }
}

package romano;
import griego.*;
public class Delta extends Alpha {
       void unMetodoD(Alpha a, Delta d){
              a.x=10;
              d.x=10;
              a.otroMetodoA();
              d.otroMetodoA();
        }
}
```

2. **Números Random.** Para hacer este ejercicio puede necesitar la clase Math: http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

   a. Escriba una clase GeneraRandom que contiene:
      i. Una **constante de clase** llamada MAXIMO_VALOR de tipo entero que almacena un valor específico (será el máximo valor por default a generar).
      ii. Un **método de clase** llamado obtenerRandom que devuelve un valor entero, considerando que sea mayor o igual que cero y menor o igual que el valor máximo especificado en el punto i.
      iii. Escriba una clase llamada TestRandom. Su método main verificará el correcto funcionamiento de la clase GeneraRandom invocando 100 veces a obtenerRandom().
      iv. ¿Es posible crear una instancia de GeneraRandom y enviarle mensajes de clase? Si su respuesta es afirmativa, indique si esto es correcto. Si su respuesta es negativa, indique por qué. En ambos casos **JUSTIFIQUE**.

   b. La clase Math está definida como ***public final class Math***.
      i. ¿Qué nos está indicando con esta definición?
      ii. ¿Conoce alguna otra clase de la API con esta misma definición?

   c. Modifique la clase GeneraRandom de modo que no permita subclases ni sobreescritura de sus métodos.

   d. Modifique la clase GeneraRandom de modo que no permita la generación de instancias.

   e. En este caso, tenemos un proyecto de sólo 2 clases, pero podríamos tener una aplicación mucho más compleja y una forma de distribuirla es creando un archivo con extensión ".jar". Desde el eclipse, exporte las clases GeneraRandom y TestRandom en un archivo JAR que ejecute el método main de TestRandom (preste atención a las opciones que aparecen durante el Wizard, en particular cuando deba indicar cuál es la clase con el método **"main"**).
      i. ¿Qué archivo nuevo se generó dentro del JAR?
      ii. Ejecute el archivo JAR generado.

3. Constructores y especificadores de acceso.

a) Pruebe éste código e indique si compila o no. De ser necesario modifique el código. **JUSTIFIQUE.**

```
package unlp.info.test;
public class SuperClase {
  private SuperClase() {
  }
}
```

```
package unlp.info.test;
public class SubClase extends SuperClase {
  public SubClase() {
  }
}
```