Diseño y Paradigmas de Lenguajes - Año 2015
Práctico Nro. 5
Tipos de Datos Abstractos y Programación Orientada a Objetos
Corresponde al **Apunte de TDA y POO**

Nota: Los ejercicios marcados con ► son para pasarlos y ejecutarlos en máquina.

Ejercicio 1.

Dado el siguiente pseudo-código que permite almacenar información de un cliente:

```
type_register Cliente {
    id;
    nbre_y_apell;
    cuit;
    empresa;
}
```

a. Codifique el tipo de datos `Cliente` en lenguaje C, considerando que `id` es un valor entero único que será provisto automáticamente por el sistema almacenando un valor secuencial luego de la creación de un nuevo cliente (variable estática). Los campos `nbre_y_apell` y `empresa` serán codificados como strings y el campo `cuit` como un entero largo. Especifique qué sentencias utilizaría para crear un nuevo cliente en un programa que posee implementado el tipo `Cliente`.

b. Codifique el tipo de datos `Cliente` en lenguaje Java, considerando las mismas especificaciones de tipos para los campos. Especifique qué sentencias utilizaría para crear un nuevo cliente en un programa que posee implementado el tipo `Cliente`.

c. Suponiendo que se desea proveer valores por defecto para todos los campos de `Cliente` (en cero los valores enteros y NULL los strings), ¿cómo lo resolvería en C y cómo en Java? Explique luego la utilidad de los *constructores* y la ventaja de contar con ellos.

► **Ejercicio 2.**

Completar la definición de la clase `Fifo`, que implementa una estructura cola

```
class Fifo{
  static int COLAVACIA = -1;
  private Object[] elementos;
  private int pri = COLAVACIA;
  private int ult = COLAVACIA;
}
```

con los siguientes métodos:

- Un constructor para crear una cola con capacidad determinada (parámetro).
- Un método `insert(Object x)` que recibe por parámetro el objeto `x` que se debe insertar.
- Un método `delete()` que retorna y elimina el primer objeto de la cola.
- Un método `isEmpty()` que retorna `true` si la cola está vacía.

Explique si la clase `Fifo` provee un adecuado encapsulamiento y ocultamiento de información y pruebe la clase `Fifo` con la siguiente clase `TestFifo`:

```
class TestFifo{
 public static void main (String[] args){
   Fifo p1=new Fifo(10);
   p1.insert(new Float(4.5));
   p1.insert("Hola");
   p1.insert(new Integer(68));
   System.out.println("\nCapacidad: "+p1.capacidad()+"\n");
   while(!p1.isEmpty())
        System.out.println("elemento: "+p1.delete());
   }
}
```

Ejercicio 3.

Escriba un trozo de programa en lenguaje Java en donde se ejemplifique el uso de una *variable de clase* y una *variable de instancia*. Luego, explique cuál es la diferencia entre ambas variables.

Ejercicio 4.

Dada las siguientes definiciones de clases, especifique cuál es la salida del método `main()` en la clase `Test1234`.

```
class Uno{
  public int test(){
    return 1;
  }
  public int resultado1(){
    return this.test();
  }
}
class Dos extends Uno{
  public int test(){
    return 2;
  }
}
class Tres extends Dos{
  public int resultado3a(){
    return this.resultado1();
  }
  public int resultado3b(){
    return super.test();
  }
}
class Cuatro extends Tres{
  public int test(){
    return 4;
  }
}
```

```
class Test1234{
  public static void main(String[] args){

    Uno o1 = new Uno();
    Dos o2 = new Dos();
    Tres o3 = new Tres();
    Cuatro o4 = new Cuatro();

    System.out.println("o1.text() =           "+ o1.test());
    System.out.println("o1.resultado1() =     "+ o1.resultado1());
    System.out.println("o2.text() =           "+ o2.test());
    System.out.println("o2.resultado1() =     "+ o2.resultado1());
    System.out.println("o3.text() =           "+ o3.test());
    System.out.println("o4.resultado1() =     "+ o4.resultado1());
    System.out.println("o3.resultado3a() =    "+ o3.resultado3a());
    System.out.println("o4.resultado3a() =    "+ o4.resultado3a());
    System.out.println("o3.resultado3b() =    "+ o3.resultado3b());
    System.out.println("o4.resultado3b() =    "+ o4.resultado3b());
  }
}
```

Ejercicio 5.

Suponga que se desea almacenar información sobre las materias que se dictan en una universidad. Defina los tipos de datos abstractos que considere necesarios para la resolución de lo requerido y luego implemente la solución en lenguaje Java. Además implemente:

1. Una clase con información general de `materias`. Se debe guardar información relativa a: nombre de la materia, código (número de 6 dígitos), vigencia (valor booleano), carreras que la dictan (arreglo de strings) y cantidad de instancias creadas de esta clase. Implemente los métodos necesarios para modificar todos los campos y también los correspondientes para obtener el valor de ellos.
2. Crear otra clase para las materias `optativas`. Esta deberá heredar el comportamiento de la clase `materias`, aunque también se debe guardar el año de implementación de la misma. Defina los métodos apropiados para modificar y obtener los datos de esta clase. Reuse código si es posible.
3. En una clase diferente defina el método principal, para que entre otras cosas, invoque a los constructores para guardar la siguiente información:
   Materia: Análisis Comp de Leng, código: 223344, vigente: Si, carreras: Prof. en Comp. y Lic. en Comp.
   Materia: Introducción a la Comp, código: 101122
   Optativa: Metaheurísticas, código: 115566
   Optativa: Minería de datos, código: 212121, año: 2005
4. Luego llene los campos incompletos de todas las materias, con información determinada por Ud.
5. Imprima toda la información de todas las materias guardadas, separando la misma por materia, también muestre el número de instancias creadas. Use carteles para diferenciar la información.