# TRABAJO PRÁCTICO N ° 1

## Pasaje de parámetros, interfaces y genericidad paramétrica

*Departamento de Ciencias e Ingeniería de la Computación - U.N.S.*

**Ejercicio 1:** Suponga que cuenta con una clase persona que tiene como atributos de instancia un nombre de tipo String y un DNI de tipo entero. Indique cuál sería la salida por consola.

```
public void cambiarDNI1(int dni){
  dni=000;}

public static void cambiarDNI2(Persona p){
       p.establecerDNI(000);}

public static void cambiarPersona1(Persona p){
       p=new Persona(“Favio”,55555555);}

public  static void cambiarPersona2(Persona p){
       p.establecerNombre(“Favio”);
       p.establecerDNI(55555555); }

public static void main (String [] args){
 Persona p= new Persona(“Alejandra”,11111111);
 System.out.println(p.toString());
 cambiarDNI1(p.obtenerDNI());
 System.out.println(p.toString());
 cambiarDNI2(p);
 System.out.println(p.toString());
 cambiarPersona1(p);
 System.out.println(p.toString());
 cambiarPersona2(p);
 System.out.println(p.toString());
}
```

**Ejercicio 2:**
Dado el siguiente listado de operaciones y sus especificaciones, implemente en Java una interface genérica llamada *“Conjunto<E>”.*

- *size(): Retorna la cantidad de elementos del conjunto.*
- *capacity(): Retorna la capacidad máxima del conjunto.*
- *isEmpty(): Retorna verdadero si y sólo si el conjunto está vacío.*
- *get(i):retorna el i-ésimo elemento del conjunto. Requiere que la posición sea válida.*
- *put(elem): Agrega un elemento al conjunto. Requiere que el conjunto no esté lleno y que el elemento no se encuentre en el conjunto. La comparación se realiza por equivalencia.*
- *pertenece(elem): Retorna verdadero si y sólo si el elemento elem se encuentra en el conjunto. La comparación se realiza por equivalencia.*
- *intersección(c): Retorna un nuevo conjunto producto de intersectar los elementos del conjunto que recibe el mensaje y el pasado por parámetro.*

**Ejercicio 3:**
Implemente una clase llamada “*ConjuntoArreglo<E>*” que implemente la interface “*Conjunto<E>*”. La clase “*ConjuntoArreglo<E>*” encapsula un arreglo genérico de tipo E.

**Ejercicio 4:**
Implemente una clase llamada “*TesterConjunto*” que permita verificar la implementación de su conjunto.

**Ejercicio 5:**
¿Qué sucedería si cambiamos la representación interna de la clase ConjuntoArreglo y en lugar de utilizar un arreglo utilizamos un Vector (java.util)?

**Ejercicio 6:**
Proponga una implementación alternativa para el método pertenece de la clase "*ConjuntoArreglo<E>*". En este caso la implementación debe ser recursiva. Escriba un planteo que se corresponda con su implementación.

Tips:

- Se sugiere que el ejercicio 1 se resuelva en papel, puede usar el entorno para probar la correctitud. Tenga en cuenta que **en Java el pasaje de parámetros siempre es por valor**;
- Programe el resto de este TP en el entorno de programación. Para esto se sugiere crear un proyecto llamado, por ej, "*Practico_1*", y dentro del mismo un paquete llamado "*TDAConjunto*" donde se encontrarán la interface "*Conjunto<E>"* y las clases "*ConjuntoArreglo<E>*" y "*TesterConjunto*";
- Tenga en claro las diferencias entre proyecto y paquete;
- Recuerde que cuando una clase implementa una interface además de heredar su comportamiento se verifica la relación "es-un";
- En la clase tester deberá elegir un tipo de dato para instanciar la genericidad paramétrica. Recuerde que en el caso de querer probar con tipos elementales deberá utilizar las clases wrapper;
- Recomendamos que el tipo estático de las variables sea el de la interface. Por ejemplo: **Conjunto<E>** c= new ConjuntoArreglo<E>().