Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica 2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #4

Fecha de entrega: 22 de Febrero, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 4". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.

Iniciación

Crear una solución para este deber. La solución debe tener 2 projectos: *Genericos* y *GenericosTests*. El projecto *Genericos* debe ser de tipo *console* y el proyecto *GenericosTests* de tipo *xunit*. Finalmente definir una clase llamada *Genericos* en el proyecto *Genericos* y una clase llamada *GenericosTests* en el proyecto *GenericosTests*.

Ejercicio #1 (20%)

Escriba un metodo estatico generico en la clase *Genericos* llamado *Head*, con un parametro generico `T`, el cual tiene tipo "`Head : T[ ] → T`". Este metodo recibe un arreglo de elementos y retorna el primer elemento del arreglo. **Nota:** No es necesario revisar que el arreglo tenga al menos un elemento.

Escribir una prueba unitaria en la clase *GenericosTests* llamada *TestHead* que verifique el funcionamiento correcto de este metodo. Esto significa que verifica que el objeto retornado sea el primer objeto del arreglo dado.

Ejercio #2 (20%)

Escribir un metodo generico llamado *Tail*, con un parametro generico `T`, de tipo "`Tail : T[ ] → T[ ]`", que acepta un arreglo como parametro y retorna un arreglo con todos los elementos, en el mismo orden a excepción del primer elemento del arreglo original.

Escribir una prueba unitaria para este metodo en la clase *GenericosTests*, llamada *TestTail*, que verifique el funcionamiento correcto de esta función.

Ejercicio #3 (40%)

```
public class Tupla<T1,T2>{

    public T1 Primero {get;}

    public T2 Segundo {get;}

    public Tupla(T1 primero, T2 segundo){
        this.Primero = primero;
        this.Segundo = segundo;
    }
}
```

Utilizar la clase *Tupla*, que esta definida en esta seccion (debe incluirla en su codigo), para definir en la clase *Genericos* un metodo generico llamado *Zip*. Este metodo recibe dos parametros genericos (`T1` y `T2`) y tiene tipo "`Zip T1`$[\,] \otimes$ `T2`$[\,] \rightarrow$ `Tupla`$\langle$`T1,T2`$\rangle[\,]$". Este metodo recibe como parametro dos arreglos y retorna un arreglo donde el cada elemento del primer arreglo es emparejado con el elemento del segundo arreglo con el mismo indice. En caso que los arreglos pasados como parametro sean de longitud diferente, el resultado debe tener la misma longitud que el arreglo más corto.

Escribir una prueba unitaria llanada *TestZip* en la clase *GenericosTests* que verifique el funcionamiento correcto del metodo *Zip*. Queda a su criterio determinar como verificar dicho funcionamiento.

Ejercicio #4 (20% + 10%)

Escriba un programa ejemplo (no lo coloque en el proyecto, sino que en la misma carpeta que sus archivos latex de esta tarea) que muestre las ventajas que tiene el metodo *Head* al utilizar tipos genericos sobre una version equivalente de ese metodo que en vez de quenericos, reciba un arreglo de tipo `object` y retorne un `object`. Brevemetne, explicar dicho programa en un archivo latex. Se le otorgara un 10% extra en esta tarea si incluye su codigo fuente en el pdf final (asi como en este pdf).