Guía de estilo

Sobre los *commits*

- Cada Actividad Evaluable (AE) tendrá una rama llamada AE01, AE02, etc. Estas ramas también se subirán a remoto.
- Los *commits* se realizarán sobre la rama de cada actividad y cuando se termine, se hará un *merge* sobre master.
- El último *commit* sobre una rama será el que se utilice para corregir dicha AE.
- Cuando se hace un *merge* en master es porque el código ya está completo.

Sobre el código

El código que se ve en los repositorios y vídeos del profesor tiene propósito didáctico. El código que debes implementar deberá seguir unas reglas de estilo. Puedes refactorizarlo antes de cada entrega, aunque si te acostumbras desde el principio será menos laborioso.

Recomendaciones:

- La convención *PascalCasing* pone en mayúsculas el primer carácter de cada palabra (incluidos los acrónimos de dos letras de longitud: *Get*, *Post*, *Retrieve*, *RetrieveDiscos*, etc.).
- La convención *camelCasing* pone en minúsculas el primer carácter de cada palabra y en mayúsculas el primer carácter de las palabras siguientes: *customerName*, *employeeDetails*, *salary*, etc.
- Los nombres de clases, funciones y propiedades: *PascalCasing*.
- Los nombres de variables y parámetros: *camelCasing*.
- No utilices guiones bajos para diferenciar palabras o, en ese caso, en cualquier parte de los identificadores

Ejemplo:

```
01.  public class EmployeeDetails
02.    {
03.        private int totalSalary = 0;
04.        public void GetEmployeeSalary( int employeeId)
05.        {
06.            int amount = 30000;
```

- En cuanto a nombres de variables, elije nombres fácilmente legibles, favoreciendo la legibilidad en lugar de la brevedad.

Mal:

```
01.  public class EmployeeDetails
02.     {
03.
04.          public void GetSalary(int x, int p)
05.          {
06.              int z = x + p;
```

Bien:

```
01.  public class EmployeeDetails
02.      {
03.
04.          public void GetEmployeeSalary( int salary, int bonus)
05.          {
06.              int totalSalary = salary + bonus;
```

Esto no quiere decir que utilices palabras innecesarias como *theSalary* o *nameValue*. En este caso sería mejor *salary* y *name*.

- Utiliza nombres semánticamente interesantes en lugar de palabras clave específicas del lenguaje para los nombres de tipo. Por ejemplo, *GetLength* es mejor que *GetInt*.
- Evita código repetido o duplicado. Reutiliza todo lo que puedas.
- Evita variables de clase si no son necesarias.
- Evita métodos muy grandes. Si no tienes más remedio, utiliza secciones con comentarios para hacerlo más legible.
- Haz un código legible y no excesivamente anidado.
- Intenta no poner mensajes. Los nombres de métodos, variables, parámetros, etc. deben ser lo suficientemente claros como para entenderlo a la primera. Antes de insertar un comentario piensa si es realmente necesario.
- No entregues código final con métodos y trozos de código comentados. Utiliza algún *commit* anterior para dejarlos de manera que lo que entregues esté limpio.
- Identa el código y elimina espacios innecesarios.

Sobre la base de datos MySQL

- Los nombres de las tablas irán en mayúsculas y en plural.
- Los nombres de los campos irán en minúsculas y en singular.
- Las claves primarias que se incluyan a propósito tendrán el sufijo “_id” o el prefijo “id”.
- Las claves ajenas tendrán el sufijo “_ref” o el prefijo “ref”.