**Meaninfull Names**

* Qué significa usar nombres que “expresen su intención”? Por qué debemos usar este tipo de nombres?
* Por qué debemos usar nombres que se puedan pronunciar?
* Qué es un encoding y cuales son algunos de los encoding más comunes que existen? Qué problemas trae y por qué no tiene sentido utilizarlos?
* Qué regla deben seguir los nombres de las clases y métodos?
* Qué es “The Inverse Scope Rule”?

Referencias:

* Book: 18 – 19, 21 -22, 23 – 24, 25
* <http://www.informit.com/articles/article.aspx?p=1323426>

**Function Arguments**

* ¿Cuál es el problema de tener muchos argumentos?
* ¿Cuál es el rango de argumentos recomendado que tenga una función?
* ¿Qué cambios podemos realizar al código para evitar tener muchos argumentos?
* Argumentos Flags.
* Argumentos Output.

Referencias

* Book: 40 – 43, 45

**Function Structure**

* Organización dentro de una clase
  + Metáfora del periódico.
  + Cómo debería estar ordenado las instancias, constructores y métodos de una clase. (Utilizan un pequeño diagrama si es posible)

Referencias

* <http://programmers.stackexchange.com/questions/199311/most-human-friendly-way-to-order-class-method-definitions>
* Temporal Coupling
  + Qué es? Qué problema ocasiona?
  + La técnica del Passing Block.

Referencias:

* <http://mikeebert.tumblr.com/post/26415251071/clean-coders-episode-4-function-structure>
* <http://geekswithblogs.net/Optikal/archive/2013/01/13/151820.aspx>
* Command Query Separation
  + Qué es un Comando? Qué es un Query?
  + Que nos dice la regla y cuáles son sus ventajas?

Referencias:

* Book: 45 – 46
* http://martinfowler.com/bliki/CommandQuerySeparation.html
* Tell Don’t Ask
  + Qué significa el principio? Qué beneficios tiene?

**Handling Errors**

* Códigos de Errores o Excepciones
  + Cuál deberíamos preferir? Por qué?
  + Qué relación tiene esto con el principio de Command Query Separation?

Referencias

* Book: 45 – 46, 104 – 105
* <http://stackoverflow.com/questions/253314/exceptions-or-error-codes>
* <http://nedbatchelder.com/text/exceptions-vs-status.html>
* No Retornar Nulos
  + Por qué
  + En qué circunstancias sí debemos devolver nulos?

Referencias

* Book: 110
* http://leedumond.com/blog/should-we-return-null-from-our-methods/
* <http://programmers.stackexchange.com/questions/120355/is-it-better-to-return-null-or-empty-values-from-functions-methods-where-the-ret>
* Null Checks
  + Cuáles son los problemas de los Null Checks excesivos?
  + Cuando utilizar Null Checks y cuando no?

Referencias:

* Book: 111 – 112
* <http://danielroop.com/blog/2009/10/15/why-defensive-programming-is-rubbish/>
* <http://elegantcode.com/2010/05/01/say-no-to-null/>

# Posibles

**Function Arguments**

(Defensive Programming) Why no es bueno. Cuando se recomienda.

**Functions Structure**

Side Efects, Qué es? Cual es el problema?

Law of Demeter? problemas de no respetarla?

Switch and Cases, Why are bad? Why are not OO? alternativas?

**Handling Errors**

- Check vs Unchecked? Cuál usar? XQ?

- Try Catch Finally First

# Preguntas:

* Otro ejemplo práctico de la técnica de passing block
* Que refactorización nos permite reducir el número de parámetros?