Temas y Lecturas

Lecturas obligatorias

Primera parte:

1. "[**Design Principles Behind Smalltalk**](https://www.cs.virginia.edu/~evans/cs655/readings/smalltalk.html)", [Dan Ingalls](https://en.wikipedia.org/wiki/Dan_Ingalls)
2. "[**Unit Testing Guidelines**](https://petroware.no/unittesting.html)", Petroware SA
3. [**What Is the Point of Test-Driven Development?**](https://github.com/gg-daddy/ebooks/blob/master/Growing%20Object-Oriented%20Software%2C%20Guided%20by%20Tests.pdf), Freeman & Pryce (Capítulo 1 del libro "Growing object-oriented software, guided by tests", Addison-Wesley - 2011)
4. "[**8 Principles of Better Unit Testing**](http://esj.com/Articles/2012/09/24/Better-Unit-Testing.aspx?p=1)", Dror Helper
5. "[**The Art of Enbugging**](https://media.pragprog.com/articles/jan_03_enbug.pdf)", [Andy Hunt](https://en.wikipedia.org/wiki/Andy_Hunt_(author)) & [Dave Thomas](https://en.wikipedia.org/wiki/Dave_Thomas_(programmer))
6. "[**GetterEradicator**](https://martinfowler.com/bliki/GetterEradicator.html)", [Martin Fowler](https://en.wikipedia.org/wiki/Martin_Fowler)
7. "[**Replace Conditional With Polymorphism**](https://sourcemaking.com/refactoring/replace-conditional-with-polymorphism)", [SourceMaking](https://sourcemaking.com/)
8. "[**¿Para qué sirve un modelo?**](https://campus.fi.uba.ar/mod/page/view.php?id=85330)", [Martin Fowler](https://en.wikipedia.org/wiki/Martin_Fowler) (traducción no oficial)
9. "[**Pruebas de software**](https://drive.google.com/file/d/1XRLSngRNe_9XOLBPvPv5FK1qXO65mAk5/view?usp=sharing)", [Carlos Fontela](https://limetfiuba.github.io/carlosfontela/)

Segunda parte:

1. "[**Continuous Integration**](https://www.martinfowler.com/articles/continuousIntegration.html)", [Martin Fowler](https://en.wikipedia.org/wiki/Martin_Fowler)
2. ~~"~~[**~~Estado del arte y tendencias en Test-Driven Development~~**](http://sedici.unlp.edu.ar/bitstream/handle/10915/4216/Documento_completo.pdf?sequence=1&isAllowed=y)~~",~~[~~Carlos Fontela~~](https://limetfiuba.github.io/carlosfontela/)
3. "[**Using Java Reflection**](http://www.oracle.com/technetwork/articles/java/javareflection-1536171.html)", Glen McCluskey

Clases:

1. **"**[**Patrones**](https://www.martinfowler.com/articles/continuousIntegration.html) **de Diseño"**
2. **"**[**Catálogo de Patrones de Diseño**](https://www.martinfowler.com/articles/continuousIntegration.html)**"**
3. **“Persistencia”**
4. **“Concurrencia”**
5. **“Diseño POO”**
6. **“EIS – Mocking”**
7. **“Mocking”**
8. **“Mocking aren´t stubs”**
9. **“MVC”**
10. **“Polimorfismo”**
11. **“SOLID”**
12. **“UML si o si”**
13. **“¿Usabilidad?”**

Artículo de Ingalls -> leverage es apalancamiento y va!

Preguntas Final

Lecturas obligatorias

* ¿Qué es una Race Condition?

Se da cuando varios threads pueden acceder a recursos compartidos (código). El resultado del programa depende de cómo se intercalen los threads.

* ¿Qué es Inyección de Dependencias?

Tiene que ver con Mocking

* ¿Cuáles son las prácticas recomendadas para CI?

la idea es facilitar y automatizar al máximo las integraciones antes de liberar el producto que se está desarrollando. Consiste en realizar la compilación, construcción y pruebas del producto en forma sucesiva y automática como parte de la integración.

* ¿Qué son los Patrones de Diseño?

Los patrones: Ayudan a decir más con menos Hablar a nivel de patrón permite mantenernos “en el diseño” por más tiempo. Empodera al equipo de trabajo. Anima a los menos experimentados a ponerse al día. Un patrón es una solución a un problema en un contexto.

* ¿Qué nos permite hacer Java Reflexion?

La reflexión permite que un programe se examine a sí mismo y manipule sus propiedades internas. Un uso tangible de la reflexión es en JavaBeans, donde los componentes de software se pueden manipular visualmente a través de una herramienta de construcción.

* ¿Qué es usabilidad (UX)?
* Human center design

Involucra al usuario en todo momento. Es iterativo.

* ¿Cómo funciona (en líneas generales) la se realización en Java?
* ¿Qué define Craig como “Experto en información”?

Poner el comportamiento en la misma clase que la data es lo que Craig Larman llama “Information Expert”. Una buena regla es que las cosas que cambian juntas deben estar juntas.

* Utilidad de TDD
* ¿Qué es SOLID?

Conjunto de reglas que favorecen y/o fomentan la escritura de código extensible y mantenible, minimizando los posibles códigos afectados.

**Principio de responsabilidad única, principio de abierto/cerrado, principio de sustitución de Liskov, principio de segregación de la interfaz, principio de inversión de dependencias.**

* ¿Qué son los tests de calidad?
* ¿Qué es la Heurística de Nielsen?
* ¿Qué afirmación del Patrón Strategy es correcta?
* ¿Cuál afirmación del patrón MVC es correcta?
* ¿Cuál afirmación de los mochitos es correcta (fake, mock, stub)?
* ¿Qué es el polimorfismo?
* ¿Qué afirmaciones permite la integración continua según el autor?
* Según UML, ¿qué es la herencia?
* ¿Qué es la usabilidad en interfaces gráficas? (clase Maia)
* ¿Qué afirmaciones son correctas en concurrencia?
* ¿Qué afirmación es correcta según el autor en The Art of Enbugging?
* ¿Qué afirmaciones de Replace Ifs with Polymorphism son correctas?
* ¿Cúal NO va en desarrollo en torno al usuario? ¿~~Es una metodología ágil~~? ¿Hay distintos prototipos? ¿El usuario está presente en todo momento? ¿Es iterativo?
* ¿El patrón state es el strategy en distinto contexto? NO
* Persistencia
* UCD
* Continuous Int
* Replace Conditionals: se pueden agregar nuevas funcionalidades agregando subclases
* Actores en diagrama de secuencia: solo puede haber uno por diagrama.
* Refactor: cuando se hace un refactor es un buen momento para aplicar un patron de diseño.
* Afirmaciones en The Art of Enbugging

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamenteImagen de la pantalla de un celular

Descripción generada automáticamenteImagen de la pantalla de un celular

Descripción generada automáticamente con confianza mediaTexto, Carta

Descripción generada automáticamenteTexto, Carta

Descripción generada automáticamenteTexto, Carta

Descripción generada automáticamente