miércoles, 26 de febrero de 2020

Sesion 5

Dependencias externas

- Pruebas unitarias y Dependencias externas
  - Objetivo: Encontrar defectos en el código de las UNIDADES PROBADAS.
    - Driver invoca al SUT, se anota el resultado esperado, ejecuta la unidad a probar y obtiene el informe.
    - Si hay alguna unidad externa, debemos poder controlar dichas unidades.
- La regla de “oro” para realizar las pruebas.
  - El código de la unidad que estoy probando ha de ser idéntico, de forma que yo **no puedo alterar** o cambiar temporalmente y luego modificarlo de nuevo.
  - Entradas directas (entras directamente, por parámetros) entradas indirectas (no están en los parámetros)
- Código estable y control de dependencias
  - Código testable: que sea fácilmente probado de forma aislada.
  - C1, C2 -> Dependencias externas, DOCs. (métodos java que no queremos probar pero componen el código a probar).
  - Debemos cambiar las dependencias (C1…) por su dobles (D1…).
- Concepto de seam
  - Es un lugar en el código (sentencia) desde la cual yo puedo cambiar el comportamiento de esa línea sin tocarla.
  - Si el SUT no tiene seams, voy a tener que refactorizar.
  - Refactorizar: cambiar el SUT pero que quede permanentemente de esa manera.
  - Seam es un punto de inyección donde donde meteré mis dobles.
  - ¿Como identificar un Seam?
    - Cuando podemos cambiar el método sin modificar el SUT
    - Primer código:
      - No es testable porque no puedo cambiar el comportamiento del código sin haberlo tocado.

    - Segundo código:
        - Es testable porque le paso por parámetro un objeto que puedo modificar al invocarlo.
    - Tercer código
        - Es testable ya que sustituyo un método por un doble
    - Cuarto código
        - Es un seam porque no necesito tocar el código de la unidad a probar para tocar su comportamiento.

- Pasos a seguir para automatizar las pruebas
    - Identificar las dependencias externas
        - Identificar los colaboradores
    - Una vez identificados, saber si es TESTABLE.
        - Si es sí, me lo salto
        - Si es no, REFACTORIZAR.
    - Para cada colaborador realizar sus DOBLES.
    - Implementamos los DRIVERS
        - Verificación basada en el ESTADO
            - Comparando el resultado que da y comparándolo con el real.
        - Verificación basada en el COMPORTAMIENTO. (Pa la siguiente semana xD)
- Formas para que sea testable nuestra SUT
    - Parametro
        -
    - Constructor
        -
    - Método setter
        -
    - Factoría
        - .

- Implementación del doble.
  - Cada tipo de doble soluciona un problema diferente sobre esa dependencia externa.
    - Stub, es un doble que controla las dependencias con entrada indirecta al SUT.
  -