Programación Orientada a Objetos

Tema 4:
Análisis y Diseño Orientado a Objetos

Tema 4-2: Pruebas y calidad del software

Universidad de Alcalá
Departamento Ciencias de la Computación

Contenidos

- Tema 4-2: Pruebas y calidad del software

1. Introducción
2. Verificación y validación
3. Pruebas de software
4. Documentación de programas Java

Introducción

Las pruebas solo pueden demostrar la presencia de errores, no su ausencia

Edsger Dijkstra - 1972

- Se debe comprobar
  - que el programa que se está desarrollando satisface la especificación, y
  - que da la funcionalidad que se espera.
- Un error en el desarrollo se paga mucho más caro cuanto más tarde se encuentra.
- El principio de Pareto es aplicable a la prueba del software ("donde hay un defecto, hay otros").
- Las pruebas deberían empezar por "lo pequeño" y progresar hacia "lo grande".
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.

Verificación y validación

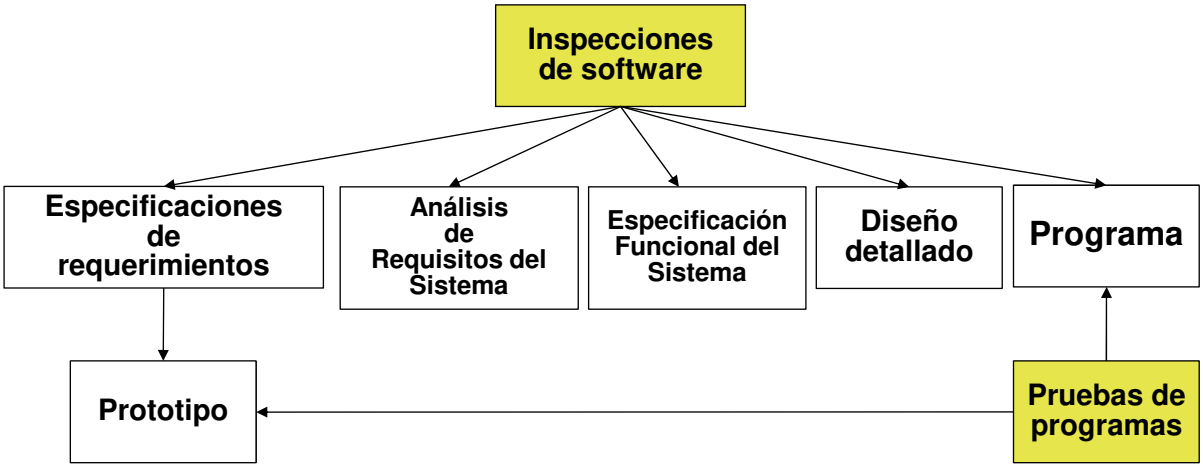
- **Verificación y Validación:**
- Verificación:
  - ¿Se está construyendo el producto correctamente? Comprobar que el producto está de acuerdo con la especificación.
  - El proceso de evaluación de un sistema (o de uno de sus componentes) para determinar si los productos de una fase dada satisfacen las condiciones impuestas al comienzo de dicha fase.
- Validación:
  - ¿Se está construyendo el producto correcto? Asegurar que el producto satisface las expectativas del cliente.
  - El proceso de evaluación de un sistema (o de uno de sus componentes) durante o al final del proceso de desarrollo para determinar si satisface los requisitos marcados por el usuario.
- La verificación y la validación tienen lugar en cada etapa del proceso de desarrollo software.

Verificación y validación

- Dentro de este proceso existen dos pasos complementarios para el análisis y comprobación de sistemas:
  - Inspecciones de software:
    - Análisis y comprobación de los documentos de requerimientos, diagramas de diseño y código fuente.
    - Solo pueden comprobar la correspondencia entre un programa y su especificación (verificación).
    - No pueden demostrar que el software es operacionalmente útil.
  - Pruebas de software:
    - Ejecución de la implementación con datos de prueba.

Verificación y validación

- Inspecciones de software se pueden utilizar en todas las fases del desarrollo:



- Las pruebas solo son posibles cuando se dispone de un prototipo o de una versión ejecutable del programa.

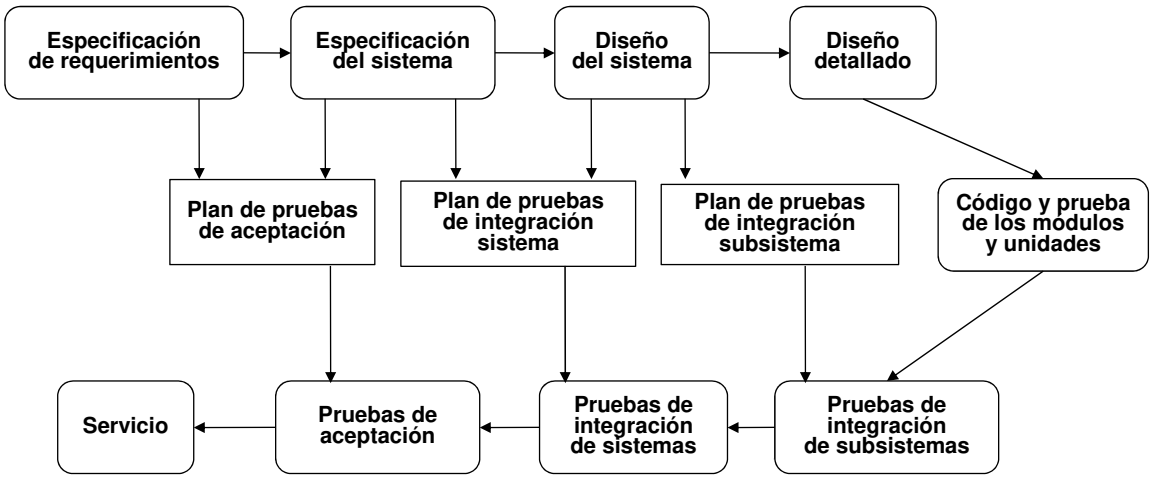
Pruebas de software

- Pruebas de software:
  - Es la técnica principal de verificación y validación.
  - Implica la ejecución del programa con datos similares a los reales.
  - Los defectos se descubren analizando la salida del programa.

- La depuración es el proceso que localiza y corrige los errores detectados.

Pruebas de software

- Planificación:



- Es necesaria para obtener el máximo provecho de las pruebas.

Pruebas de software

- Tipos de pruebas:
  - Pruebas de integración:
    - sistema / subsistema
    - entrega o aceptación
    - rendimiento
  - Pruebas unitarias o de componentes.
  - Pruebas de sistemas OO.

Pruebas de software: Pruebas de integración

- Pruebas de integración: sistema / subsistema
  - La integración de un sistema se realiza a partir de sus componentes. Se comprueba que los componentes pueden funcionar juntos.
  - Integración descendente:
    - primero se desarrolla el esqueleto del sistema y se le añaden los componentes.
  - Integración ascendente:
    - primero se integran los componentes de la infraestructura y se le van añadiendo las nuevas funcionalidades.
  - Se debe realizar una aproximación incremental, añadiendo componentes de forma sucesiva.

Pruebas de software: Pruebas de integración

- Pruebas de integración: entrega / aceptación
  - Intentan demostrar al cliente que el sistema satisface los requerimientos.
  - Las pruebas se derivan de la especificación del sistema.
  - Hay que comprobar que:
    - los resultados son los esperados a entradas definidas.
    - las situaciones de error son obtenidas por entradas que las provocan.

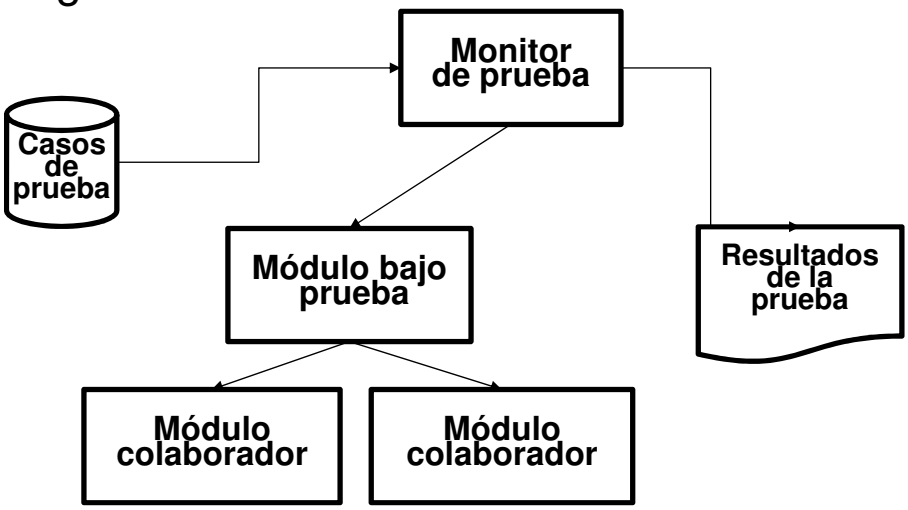
Universidad de Alcalá
Departamento Ciencias de la Computación

Pruebas de software: Pruebas de integración

- Pruebas de integración: rendimiento / estrés
  - Se hacen con el sistema integrado.
  - Se prueban propiedades como:
    - Rendimiento: puede con la carga esperada.
    - Fiabilidad.
  - Se debe diseñar un perfil operacional intentando reflejar la combinación real de transacciones.
  - Funciones:
    - Probar el comportamiento de fallo.
    - Sobrecargar el sistema.

Pruebas de software: Pruebas unitarias

- Pruebas unitarias
  - Son las pruebas de forma independiente de cada una de las clases que forman el programa.



Universidad de Alcalá
Departamento Ciencias de la Computación

Pruebas de software: Pruebas unitarias

- Pruebas unitarias
  - Pruebas de integración ascendente:
    - Se combinan módulos del nivel más bajo en grupos.
    - Se construye un monitor de prueba para gestionar los casos de prueba.
    - Se prueba el grupo.
    - Se eliminan los monitores sustituyéndolos por módulos reales.
    - Se realizan pruebas de regresión:
      - Repetir ciertos casos de prueba que funcionaban antes de sustituir por los módulos reales, así se asegura que no se introducen nuevos errores.

Pruebas de software: Pruebas unitarias

- Pruebas unitarias
  - Pruebas de integración descendente:
    - Se utiliza como control del programa principal.
    - Se construyen los módulos colaboradores inmediatamente subordinados.
    - Se van sustituyendo los módulos colaboradores por módulos reales.
    - Se prueba cada vez que se integra un nuevo módulo.
    - Se continúa reemplazando los módulos colaboradores hasta llegar a los terminales.

Pruebas de software

- Pruebas de sistemas OO
  - Niveles de pruebas:
    - Métodos y operaciones individuales de la clase.
    - Clases individuales.
    - Agrupaciones de objetos (integración).
    - Sistema.
  - Pruebas de clases:
    - Probar todas las operaciones.
    - Probar todos los atributos.
    - Ejecutar los objetos en todos los estados posibles.

- Pruebas de sistemas OO
  - Pruebas de integración de clases:
    - Consiste en asegurar que las clases y sus instancias, conforman un software que cumple con el comportamiento definido.
    - Pruebas basadas en escenarios:
      - Se definen los escenarios (casos de uso) que especifican como se utiliza el sistema.
      - Se comienza probando los más probables, luego los menos comunes y se termina con los excepcionales.
    - Pruebas basadas en subprocesos:
      - Son pruebas de cadenas de eventos.
      - Analizar las respuestas obtenidas al introducir un evento concreto o conjunto de eventos.

Documentación de programas Java

DOCUMENTACIÓN DE PROGRAMAS

- Cuando se llega a la fase de implementación del sistema una de las tareas a realizar además de la codificación del código y su prueba es la documentación, para ello conforme se codifican clases es conveniente ir documentándolas.
- En Java hay que seguir unas sencillas reglas para comentar los programas y utilizar la herramienta **javadoc** incluida en el jdk.
- Todos los comentarios introducidos en el código mediante /** y */ serán incluidos en la documentación del código generada por javadoc.
- Los comentarios de documentación describen las líneas de código que se sitúan a continuación. Se pueden incluir etiquetas html ya que la documentación se genera en este lenguaje.
- Solo se procesan los comentarios de documentación inmediatamente anteriores a una clase, interfaz, método o atributo. Si no se proporciona un comentario de documentación para un método heredado, el método "hereda" el comentario de su padre.

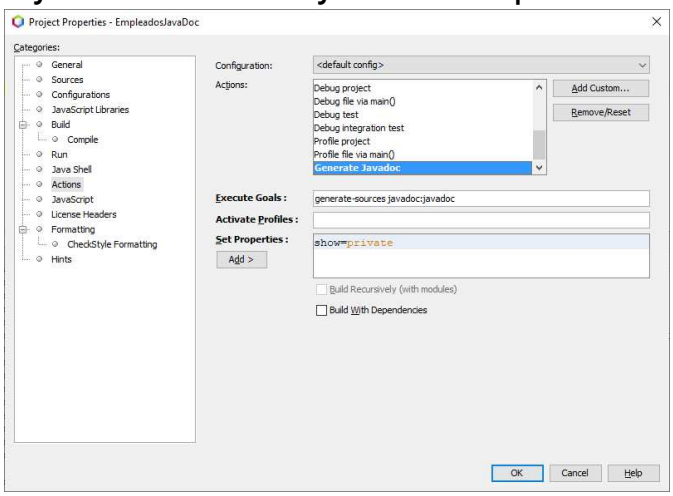
Universidad de Alcalá
Departamento Ciencias de la Computación

DOCUMENTACIÓN DE PROGRAMAS

- Los comentarios de documentación pueden incluir etiquetas que almacenan tipos de información particulares. Todas las etiquetas empiezan con el símbolo @.
- Las etiquetas más interesantes son las siguientes:
  @param: documenta un solo parámetro de un método.
  @return: documenta el valor de retorno de un método.
  @throws o @exception: documenta una excepción lanzada por un método.
  @author: especifica el autor del código
  @version: especifica la versión del código
- Para generar la documentación (si no tenemos entorno de desarrollo), debemos ir al directorio donde tengamos el código fuente y ejecutar el siguiente comando: *javadoc *.java -private –d javadoc* de esta forma generamos la documentación en un directorio llamado javadoc de todos los programas.

DOCUMENTACIÓN DE PROGRAMAS

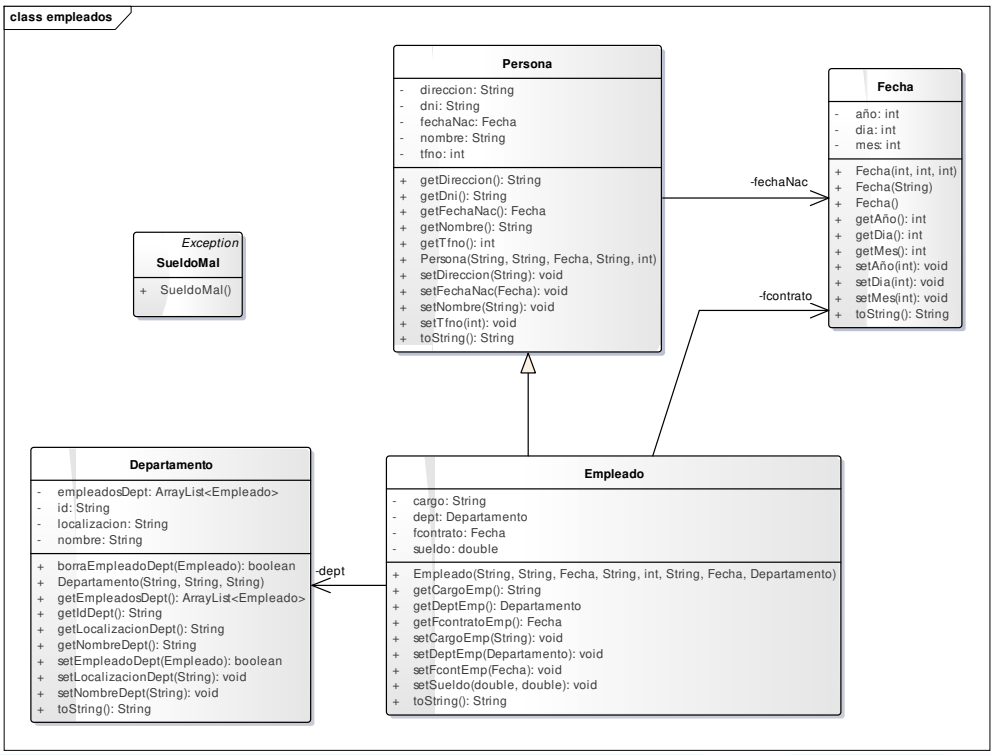
- NetBeans: Para generar la documentación de un proyecto, lo marcamos y con el botón derecho del ratón en el menú contextual seleccionamos “Generate Javadoc”.
- Previamente debemos modificar las propiedades del proyecto para que incluya los atributos y métodos privados:



Universidad de Alcalá
Departamento Ciencias de la Computación

DOCUMENTACIÓN DE PROGRAMAS

- EJEMPLO:



```
public class Persona {
  /** Atributo para almacenar el DNI de una persona */
  private String dni;
  /** Atributo para almacenar el nombre de una persona */
  private String nombre;
  /** Devuelve el Nombre de una persona */
  public String getNombre() { return this.nombre; }
  /** Establece el Nombre de una persona */
  public void setNombre(String nombre) { this.nombre = nombre; }
}

  /** Establece el sueldo del empleado
   * @param horas Horas trabajadas
   * @param precioHora Precio de la hora trabajada
   * @throws SueldoMal El sueldo es incorrecto */
  public void setSueldo(double horas, double precioHora) {
    try {
      if (horas <= 0 || precioHora <= 0) throw new SueldoMal();
      this.sueldo = horas * precioHora;
    } catch (SueldoMal sm) { System.out.println( "\n" + sm.getMessage() ); }
  }

  /** Imprime el nombre y el departamento del Empleado
   * @return Información de los datos del empleado */
  public String toString() {
    return super.toString() + " # Departamento: " + dept.toString();
  }
```