

PROGRAMACIÓN II


BIMESTRE II


Angel Chuncho / angel.chuncho@epn.edu.ec

Clase #01 - Revisión del examen

17/01/2023

En la presente clase se realizó la revisión y corrección del examen correspondiente al primer bimestre. La clase se inició preguntando a cada uno de los compañeros presentes acerca de que se les había complicado más entender y realizar solo con ver el diagrama UML para así obtener una perspectiva general de como se desarrolló el examen.

El ingeniero procedió a resolver el examen con todas las indicaciones de la rubrica tal y como se puede evidenciar en la figura 2.  Figura 1. Diagrama UML del ejercicio del examen del I Bimestre

 Figura 2. Resolución del examen por parte del ingeniero

Clase #02 - Librerías, UML, Try catch

18/01/2023

Hoy se ve:

1. Agregar librerías en el código
2. Herencia en UML
3. Programas "inmortales" / manejo de excepciones

La interfaz gráfica cuenta si le gusta al usuario.

Las librerías se empaquetan en un archivo .jar, en nuestro idioma coloquial sería un .zip, estas librerías permiten utilizar código que alguien ya creó. Una vez descargada la librería para poder implementarla se la debe agregar así:

En el asistente de java me dirijo a Referenced Library se da clic en el + y se selecciona la librería deseada. Para poder usarla debo revisar la información del creador de la librería para conocer qué comandos puedo utilizar y como utilizar la librería.

Para definir de forma gráfica la asociación o relación entre clases se la realiza mediante una línea sin puntas de flecha (relación bidireccional).

En la relación también puede haber una flecha, eso significa quien tiene la relación (relación unidireccional). Otra relación es la multiplicidad que se representa con el símbolo * lo que significa "varios", se utilizan listas.


Para que el programa no se cuelgue y siga avanzando se debe manejar las excepciones, para ello se utiliza el try catch.

En el try se pone el código que hace que el programa se cuelgue, es decir, try evita que el programa se cuelgue si la línea ingresada en try da error, entonces ahora interviene el catch el cual maneja la excepción y da alternativas o una solución al, un problema con esta función es que cuando el programa se cuelga en una

línea entonces lo que está por debajo de dicha línea con error ya no se va a ejecutar.

Una opción es usar finally que significa "te tienes que ejecutar con o sin errores". Estamos utilizando arquitectura en capas.

Una forma más elegante de controlar excepciones es throw que significa burbujear es decir "suelto para que la siguiente capa la coja".

Figura 3. Ejemplo de relacion entre clases Figura 4. Uso del try catch

Clase #03 - Custom Exception

19/01/2023

Custom exception hace referencia a personalizar las excepciones, para ello se utiliza una nueva clase y se la implementa en la capa framework, esto con la finalidad de poder mandar mensajes personalizados cuando salte una excepcion en la ejecucion y asi poder detectar las fallas mas rapido.

Una ventaja de personalizar las excepciones es cuando tenemos un programa grande y cuando personalizamos podemos ir a la clase donde creamos la personalizacion y podemos comentar y asi evitar que salga saltando esa excepcion en la consola lo cual evita que la consola se inunde de mensajes pero el programa seguira funcionando.

Figura 5. Uso de excepciones personalizadas


Clase #04 - Data y Arquitectura N-Tier

20/01/2023

La arquitectura n-Tier es una de las más tradicionales y usada en todo lenguaje de programación. Esta arquitectura hace segmentación por capas y cada una de esas capas tiene una responsabilidad dentro de la aplicación (es un simil a los órganos del cuerpo humano) pero todas funcionan o deben funcionar de forma integrada. Cuando una capa comienza a fallar va a generar problemas. En el curso se estudiara las siguientes capas:

- **Capa Framework:** se la puede absolver en cualquier otra capa y en cualquier momento por eso en el bosquejo se ubica de manera lateral, es decir, cualquier otra capa puede invocar a cualquier componente que se encuentre dentro de esta capa.
- **Capa User Interfaz (UI):** en esta capa se ubican todas las interfaces y no debe ir nada mas que eso.
- **Capa Business Logic (BL):** esta capa es la encargada de la logica de negocios o regla de negocios, es decir, la logica son los calculos y procesos que se deben dar mientras que las reglas son las cosas que deben cumplirse.
- **Capa Data Access Component (DAC):** permite conectarse a la base de datos y acceder a los datos que ahi se encuentran, esta capa solo puede ser invocada por la capa superior la cual es la Business Logic y asi sucesivamente hasta salir, no se puede romper este orden.

Esta arquitectura es util para genera escalabilidad y organizacion su desventaja es que mientras mas capas tenga nuestra aplicacion implica tambien que va a tener mas complejidad. Cabe recalcar que cada capa es un paquete ya que si no estan en paquete no se pueden escalar.

Figura 5. Arquitectura n-Tier de la app Tinder Pet

Clase #05 - Base de Datos SQLite

25/01/2023

0. Instalar extensión para Visual Code llamada SQLite
 1. Crea el directo de la base de datos
 2. Crea la base de datos (nombre.db)
 3. Crear carpeta (directorio) de SQL Script
 4. Shemma.sql (Aqui se colocan las estructuras de las tablas, es decir, ayuda a crear las tablas por código)
- A continuación se presenta el archivo shemma.sql

```
/**
 * pat_mic : 20.ene.2k23
 * script de base de datos
 CRUD
 CREATE --> Insert
 READ   --> leer
 UPDATE --> actualizar
 DELETE --> borrar
 */
--Revisar el entorno de trabajo
.version
.database
.show
.tables
-- Con DROP TABLE se elimina la tabla deseada
DROP TABLE PERSONA;
DROP TABLE MascotaTipo;

CREATE TABLE MascotaTipo
(
    IdMascotaTipo INTEGER PRIMARY KEY AUTOINCREMENT,
    Nombre VARCHAR(10) NOT NULL,
    Estado VARCHAR(1) NOT NULL DEFAULT('A')
);

INSERT INTO MascotaTipo(Nombre, Estado) VALUES ("Perros", "A");
INSERT INTO MascotaTipo(Nombre) VALUES ("Gatos");
INSERT INTO MascotaTipo(Nombre) VALUES ("Peces");
INSERT INTO MascotaTipo(Nombre) VALUES ("Cuyes");
--DELETE FROM MascotaTipo WHERE IdMascotaTipo > 9;
SELECT * FROM MascotaTipo;

UPDATE MascotaTipo SET Estado = "X"
WHERE IdMascotaTipo in (6,12);

SELECT * FROM MascotaTipo WHERE Nombre like '%e%';

CREATE TABLE PERSONA
(
```

```

    ID INTEGER PRIMARY KEY,
    NOMBRE VARCHAR(10),
    APELLIDO VARCHAR(10)
);
-- DROP TABLE T1;
-- DROP TABLE T2;
INSERT INTO PERSONA (ID, NOMBRE, APELLIDO) VALUES (1, "Pepe ", "perez");
INSERT INTO PERSONA (ID, NOMBRE, APELLIDO) VALUES (2, "Ana", "Suarez");
INSERT INTO PERSONA (ID, NOMBRE, APELLIDO) VALUES (3, "Juan",
"Sanchez");INSERT INTO PERSONA (ID, NOMBRE, APELLIDO) VALUES (4, "Lucas Juan",
"Montalvo");

SELECT ID, NOMBRE, APELLIDO FROM PERSONA;
SELECT ID, NOMBRE, APELLIDO FROM PERSONA WHERE ID = 2;
SELECT ID, NOMBRE, APELLIDO FROM PERSONA WHERE NOMBRE LIKE '%JUAN%';-----

-----
CREATE TABLE PET
(
    ID INTEGER PRIMARY KEY,
    NOMBRE VARCHAR(10),
    EDAD INTEGER
);
--DROP TABLE PET;
INSERT INTO PET (ID, NOMBRE, EDAD) VALUES (1, "VALUMA", 1);
INSERT INTO PET (ID, NOMBRE, EDAD) VALUES (2, "JUANA", 2);
INSERT INTO PET (ID, NOMBRE, EDAD) VALUES (3, "COMOTU", 3);
SELECT * FROM PET;
-----

```


Cuando se quiere matar a una tabla entera se utiliza la palabra **DROP**, se utiliza el **DELETE** para borrar una fila de la tabla.

Con la palabra **AUTOINCREMENT** la base de datos se hace cargo sola de añadir el id por lo cual podemos ahorrar código

Clase #06 - Asociación y Composición

08/02/2023

- **Agregación:** es un tipo de asociación que indica que una clase es parte de otra clase (composición débil). Los componentes pueden ser compartidos por varios compuestos (de la misma asociación de agregación o de varias asociaciones de agregación distintas). La destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición. La agregación se representa en UML mediante un diamante de color blanco colocado en el extremo en el que está la clase que representa el "todo".
- **Composición:** es una forma fuerte de composición donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor. Los componentes constituyen una parte del objeto compuesto. De esta forma, los componentes no pueden ser compartidos por varios objetos compuestos. La supresión del objeto compuesto conlleva la supresión de los componentes. El símbolo de composición es un

diamante de color negro colocado en el extremo en el que está la clase que representa el "todo"
(Compuesto).  Figura 6. Arquitectura n-Tier de la app Tinder Pet

Semana 7 - Exposiciones del proyecto

Del 27/02/2023 Al 03/03/203

28/02: Tiempo para finalizar proyectos

01/03 - Grupo 1: El proyecto de este consistio en un juego de puzle en el cual se debia cruzar un laberinto desde un punto de partida hasta un punto de llegada en el tiempo establecido, utiliza como perifericos el mouse y teclado.

02/03 - Grupo 4: El proyecto del grupo consistio en un juego el cual se basa en el ping pong y el periferico utilizado es una palanca de videojuegos.

03/03 - Grupo 5: El grupo presento un juego llamado pat pat el cual consiste en manejar un patito y evitar obtaculos con el objetivo de recorrer la mayot distancia posible y el puntaje va aumentando conforme el tiempo, el dispositivo utilizado es el mando de una consola.

Semana 8 - Exposiciones del proyecto

Del 06/03/2023 Al 10/03/203

07/03 - Grupo 6: El trabajo de este grupo consistio en un inventario mediante la implementacion de una base de datos, el dispositivo utilizado fue un lector de codigo de barras.

08/03 - Grupo 2: El proyecto del grupo consistio en un inventario de productos y realizar ventas con su respectiva factura, el dispositivo utilizado es un lector de codigo de barras.

09/03 - Grupo 3: El proyectado realizado por el grupo consistio en una sistema vacacional el cual permite registrar materias y genera un documento pdf con un codigo QR, el dispositivo necesario es un celular con camara.

10/03: Explicación de la rubrica del examen para el segundo bimestre.