Repaso General Examen Final

JAVA

Java es un lenguaje de programación. Orientado en el paradigma orientado a objetos (su fortaleza). Uno de los 3 lenguajes más utilizados a nivel mundial.

Existen distintos tipos de datos:

- Primitivos: byte, short, int, long, booleanos, entre otros.
- Instancias de clases: String, Integer
- Datos de objetos: Creados por el programador.

Estructuras de Control:

Nos permite determinar el orden de ejecución de las sentencias de un programa. Recordemos que los programas se ejecutan una instrucción tras otra, por lo tanto, son herramientas necesarias y útiles para lograr concretar nuestra lógica de negocio.

- Tipos de estructuras de Control:
    - Condicionales: IF / IF ELSE / SWITCH
    - Repetitivas: WHILE / DO WHILE / FOR

Subprogramas - Funciones:

Herramienta que nos permite dividir problemas complejos en pequeños procesos para fácil resolución y lectura.

Pueden tener retorno o no.

Pueden recibir argumentos para operar dentro de la función. Siempre que enviamos argumentos a una función, debemos respetar el orden y tipo de datos al invocar la función (en esta etapa se llama parámetro)

Los Objetos en JAVA:

Esta forma de programar nos permite individualizar los datos, dividiendo en partes la existencia de datos. Es decir, sus atributos, sus métodos, sus creaciones y sus manipulaciones.

El objeto es un prototipo, un molde de “algo”. Es la forma mas cercana de simular en programación las cosas de la vida real, teniendo sus propiedades específicas.

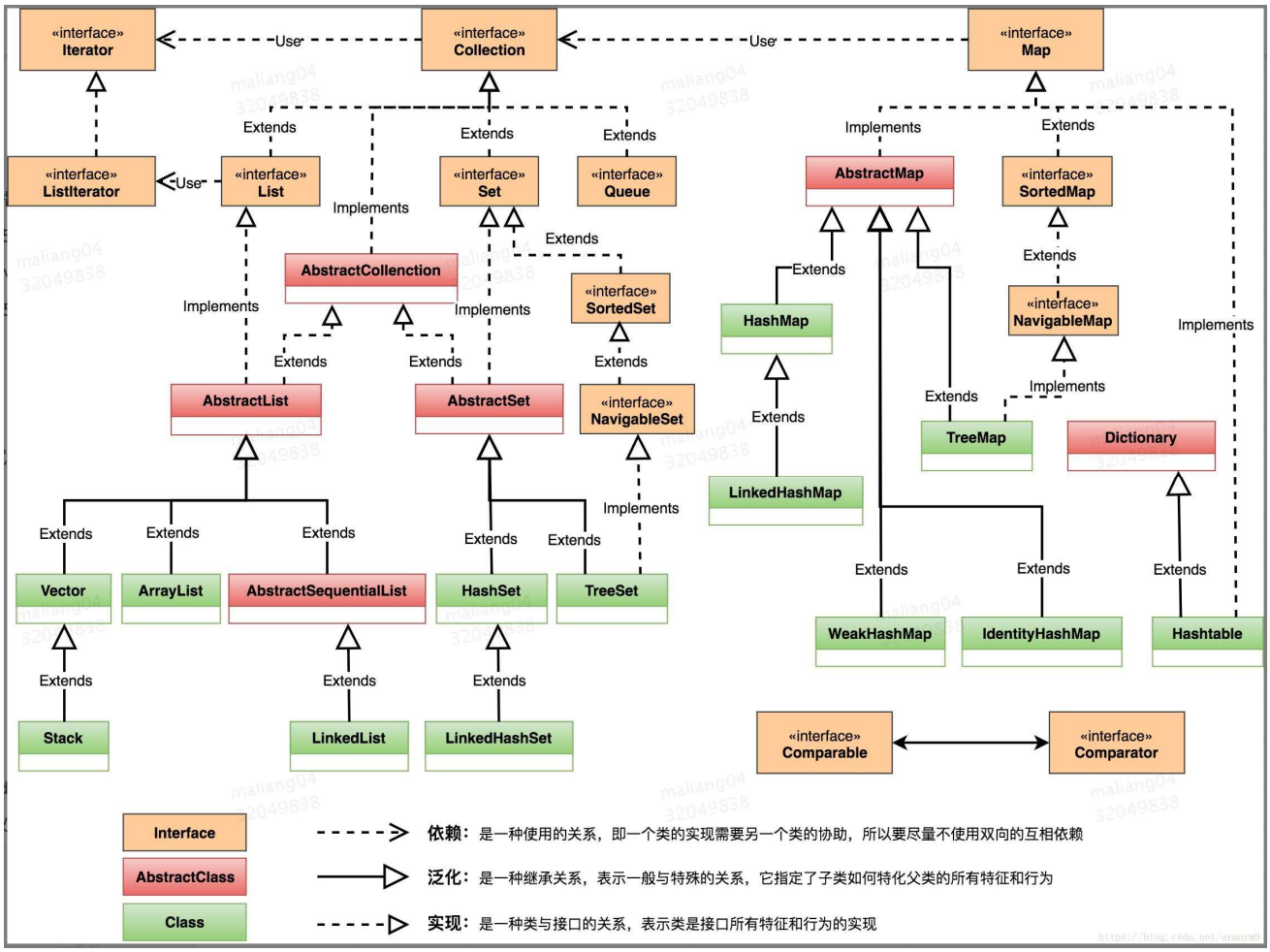
Puntos a tener en cuenta de un objeto:

- Clases: Moldes
- Atributos: Son características definidas.
- Métodos: Comportamiento

Las Colecciones:

Las colecciones representan un grupo de objetos. Podemos almacenar cualquier tipo de ellos. Por lo tanto, por definición, no admite datos primitivos.
Son de tamaño dinámico, es decir, su tamaño no es fijo.

En java implementa un conjunto de clases e interfaces para implementar su uso.



Listas(List) No poseen un orden especifico. Admiten duplicidad de datos

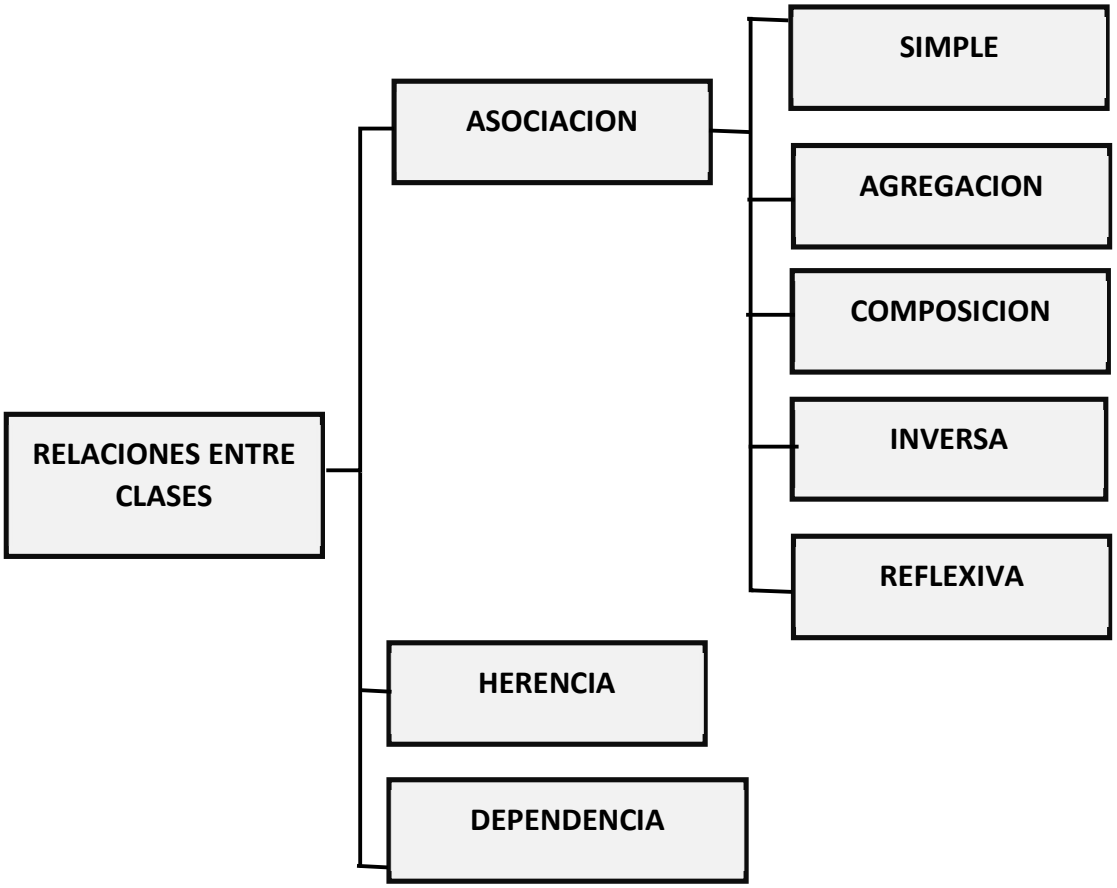
Conjuntos(Set): Poseen un orden especifico. No admiten duplicidad de datos

Algunas formas de recorrerlas:

- FOR
- FOR EACH
- ITERATOR

Las Relaciones entre clases:

Es la herramienta que nos permite representar de forma mas real el comportamiento de mis OBJETOS.
Es decir, mis clases no trabajan de forma independientes, tiene vinculaciones con otras.



- TIENE (Tener) : Define una conexión entre clases
  - Asociación Simple. Relación mas débil.
  - Agregación (Usa / Contiene) Ej: Una persona usa anteojos
  - Composición (Posee) Ej: Una persona posee órganos

- ES (Ser)
  - Herencia

Recordar que, dentro de la programación, tenemos los UML, que es un lenguaje unificado de modelado. Este es un lenguaje grafico para visualizar, especificar, construir y documentar sistema.

Como declaro una relación???? Declaro un atributo en mi clase del "tipo" de la clase con la que quiero crear la relación. Tener en cuenta si es único elemento o muchos.

La herencia, es un mecanismo de implementación el cual elementos más específicos incorporan la estructura y comportamiento de elementos más generales.

Es la capacidad que tiene una clase de heredar los métodos y atributos de otra ya existentes de otra.
Las que hereda → Subclase / Clase Hija
Quien transmite → Superclase / Clase Madre

Principales objetivos:

- Simplificar el código, reutilizando alguno ya existente.
- Clasificar los tipos de datos por "variedad", acercando el modo de programar al modo de razonar humano.

Como implementarlo:

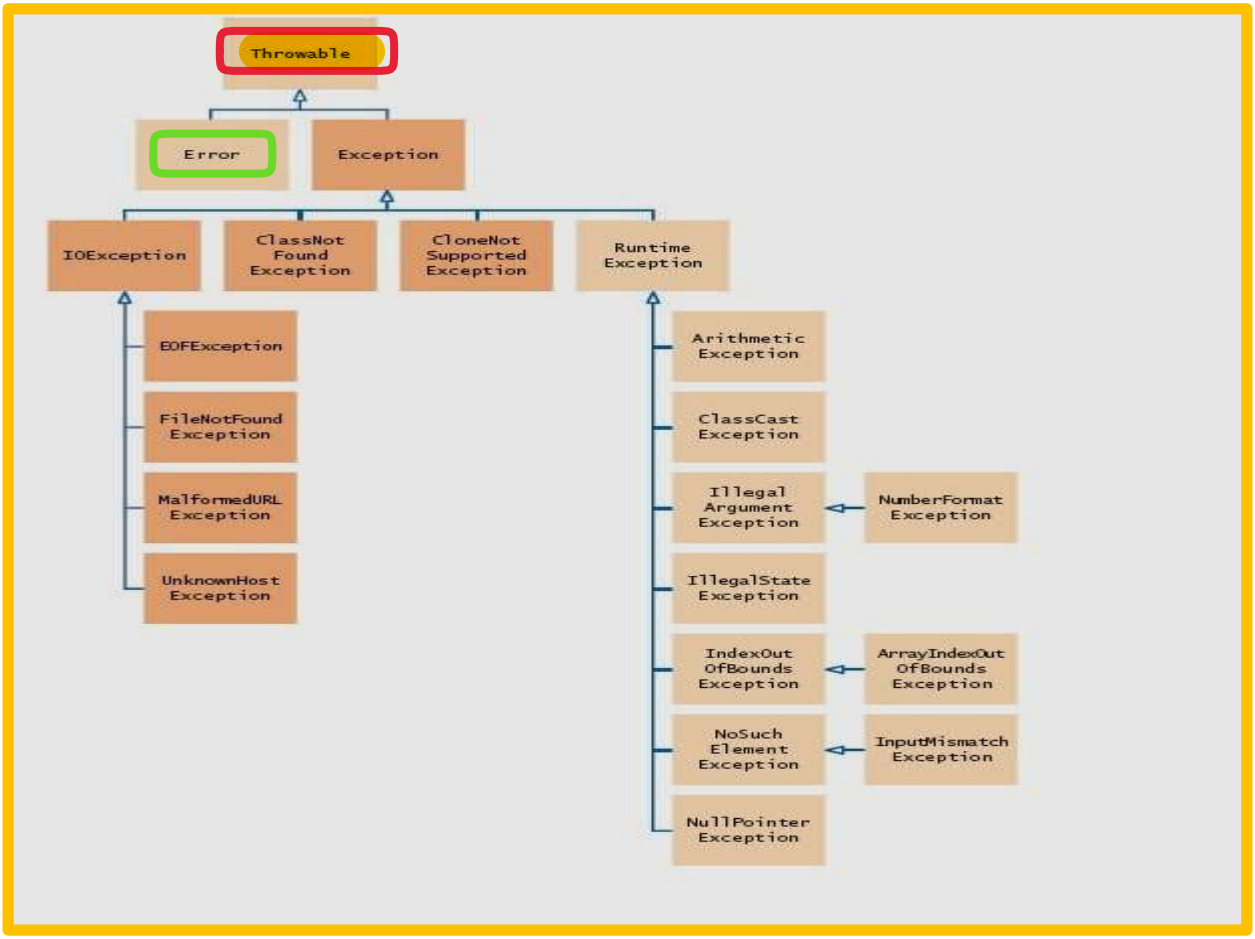
- Superclase: Declaro sus atributos (protected), creo sus constructores, métodos GET y SET para las clases externas a la familia
- Superclase: Declaro sus atributos específicos. Incorporo " ***extends nombreClaseMadre*** ". Creo su constructor, invocando con "super()" al constructor de su clase madre {Tanto en el vacío como el que recibe parámetros). Creo sus métodos GET / SET de atributos específicos de la clase, si es necesario.

Las Excepciones:

La excepción es una abreviación de la frase "Evento Excepcional". Ocurre durante la ejecución de un programa e interrumpe el flujo normal de las instrucciones de un programa.

El manejo de excepciones lo hacemos a través del uso de la clase Exception permitiendo soportar los "errores". Es decir, que si algo no sucede como lo previsto inicialmente, este decidido por el programador como debe continuar el flujo del programa. OJO no corregimos el error.

Al utilizar esta herramienta, creamos un objeto especial, con la capacidad de cambiar el flujo normal de ejecución. Cuando se detecta un error, una excepción debe ser lanzada. Este objeto es creado con la información de la misma (tipo y estado del programa cuando ocurrió).



Palabras reservadas:

- **TRY** : Donde incorporo todas aquellas instrucciones que pueden largar el error.
- **CATCH**: Donde trata la excepción. Puede existir múltiples concatenaciones, recordando siempre en nivel de jerarquía de las mismas.
- **FINALLY:** Bloque que siempre se ejecuta, sin importar si entro o no al bloque CATCH. Es decir, toda instrucción que este en este bloque, se ejecuta siempre como salida de la estructura Try-Catch.
- **THROW**: Para lanzar manualmente las excepciones, relacionado a “Crear mis propios objetos del tipo Exception”. Facilitando así, la devolución de la descripción del error ocurrido.
- **THROWS:** Lista las excepciones que un método puede lanzar. Puedo concatenar varios de ellos(en el método o la clase especifica).

MySQL

Las bases de datos son un conjunto organizado de datos utilizados para modelar un aspecto relevante de la realidad. Estos datos se almacenan de manera permanente, para ser procesados y transformados en información.

La principal característica de las bases de datos relacionales es que los elementos se relacionan como un conjunto de tablas, con columnas y filas respectivamente.

Los elementos se relacionan con las relaciones pre establecidas.

Utilizan SQL (Lenguage de Consultas Estructurado). Permite recuperar, modificar, crear y recuperar información.

ELEMENTOS DE UNA BASE DE DATOS (DATABASE / SCHEMA)

- **TABLAS** → Representa a un conjunto de objetos, es decir, *un grupo* de objetos del mismo tipo de datos
- **FILAS** → Representa un objeto con existencia independiente, es decir, un *registro*.
- **COLUMNAS** → Representa la característica especifica de cada objeto, es decir, sus *atributos*.

Tenemos comandos, funciones y clausulas de agrupamiento que nos permiten realizar nuestra manipulación de información de forma precisa, especifica y concreta al dato que queremos crear / obtener.

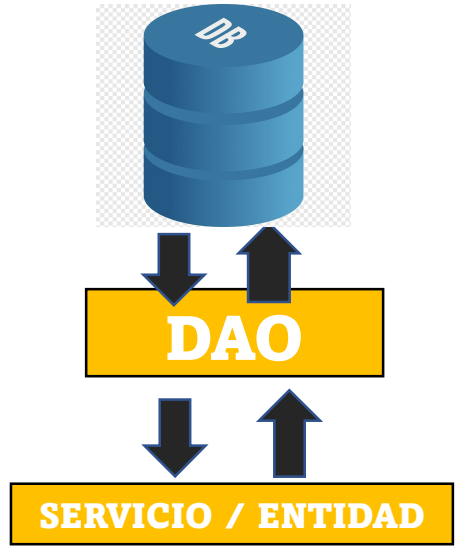
JDBC

Recurso que nos permitirá conectar a Java con nuestra base de datos. Utilizando un patrón de arquitectura DAO (Data Acces Object)

Base de Información

Objeto de Transferencia de Datos

Lógica Aplicación



Profe Adri Bestilleiro

egg

Herramienta:

JDBC Driver: Librería que nos permitirá realizar la conexión, es el controlador que proporciona el acceso a los datos.

MAQUETA DE MI PROYECTO:

*Paquete Entidades – Puedo unificar según tipo Objeto
*Paquete Servicios – Puedo unificar según tipo Objeto
*Paquete DAO - Persistencia
*Paquete Principal
*Otras Utilidades

QUE CONTIENE CADA CLASE:

- **Entidad**: Sin modificaciones. creo el "Molde" de mis futuros objetos
- **Servicio**: Realizo la lógica actual para A -B - M. Recordar crear una instancia de mi clase DAO.
- **DAO por cada entidad:** Realizo la lógica necesaria para los CRUD (create – read – update – delete) . Acá se encuentran mis consultas SQL nativas.
- ***Clase DAO, superclase:*** Dejo mis funciones encargadas de conectarse – desconectarse – insertarModificarEliminar – consultas.
  - Atributo de clase Connection: Ocupado de crear la conexión con la base de datos especifica.
  - Atributo de clase ResultSet: Almacena mis registros recopilados según previa condición analizada.
  - Atributo de clase Statment: Para dar información de mis querys.

JPA

API de persistencia, creada para Java, que utiliza ORM – Mapeo de objetos relacionales.

Como funciona? Tenemos anotaciones o etiquetas que nos permite declarar el modo de relacionar nuestras clases y atributos con las tablas de nuestra base de datos. Estas anotaciones, afectan a la línea siguiente de donde están declaradas.

Algunas anotaciones:

- @Table
- @Column
- @Id
- @Entity

- @Temporal
- @<<Relaciones>>

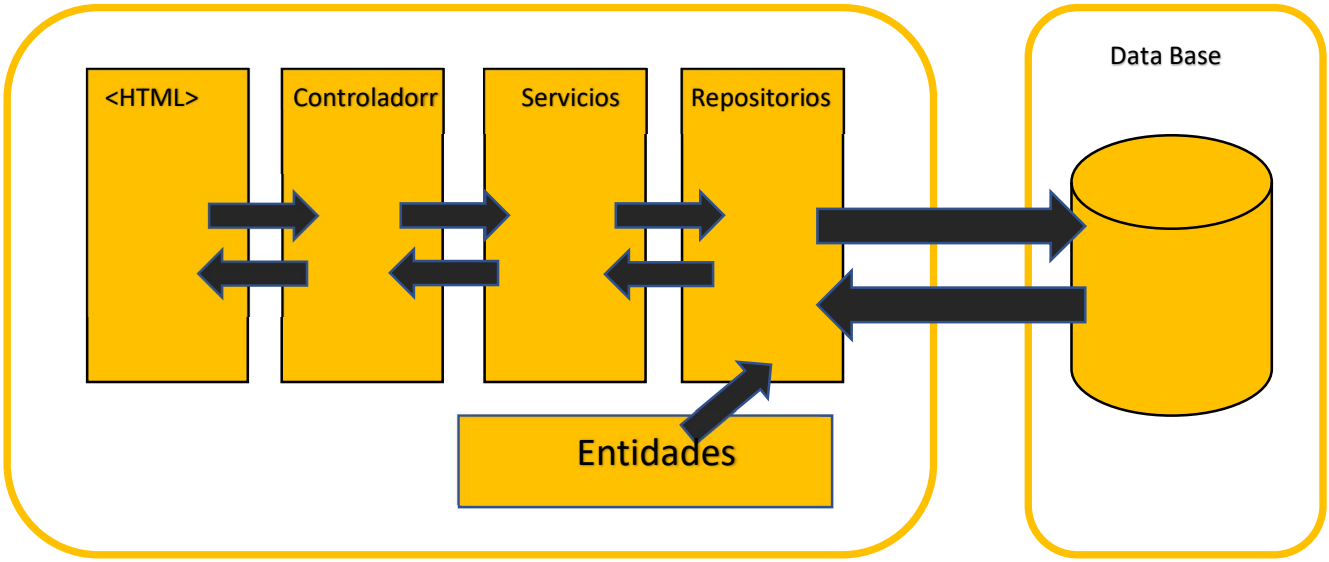
Como logra trabajar JPA:

- Declarando la **entidad** como tal, para que luego se convierta en una tabla.
- **Atributo de la clase EntityManagerFactory**: Encargado de abrir la conexión a la base de datos, utilizando nuestra unidad de persistencia declarada.
- **Atributo de la clase EntityManager**: Nace gracias a la existencia del EntityManagerFactory. Me permite acceder a los distintos métodos (persistencia, eliminación, obtener resultados de consulta)

SPRING

Es un **framework** que nos permite crear aplicaciones Java de forma sencilla, liviana, con alto rendimiento.

Integra de forma sencilla distintas conexiones: correo / transacciones / seguridad / mensajes / base de datos / entre otros.
Utiliza un modelo MVC. Recordando que cada capa debe ocuparse de realizar su tarea especifica dentro de nuestro proyecto.



Como funciona Spring para esta arquitectura???

Posee anotaciones que indican el comportamiento y funcionamiento de cada uno de los componentes.

Deben ser declaradas antes del atributo, método o clase al que van a afectar.

Algunas anotaciones:

- @Service
- @Repository
- @ Controller
- @Autowired
- @RequestMapping

Basado en 3 pilares:

- Inyección de dependencias →Módulos, dependencias y utilidades provistas por Spring Framework. Recordar que esta configuración queda explicitada en el archivo POM de nuestro proyecto.
- Programación orientada a aspectos →Relacionado a la independencia entre las capas de la información y la sincronización para vincular la misma.
- Abstracción de servicios empresariales → Relacionado a la sistematización y simplificación de ciertas tareas (Ej. Conexión)