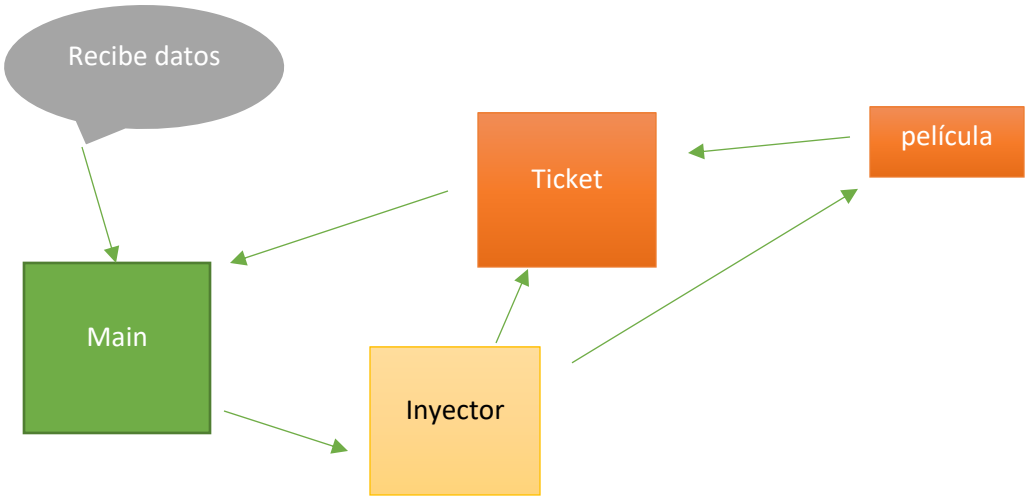
Examen Semana 2
Emmanuel Reyes Hernandez

Inyección de Dependencias

La inyección es un patrón de diseño que se ocupa para poder delegar responsabilidades, este se podría entender como no dejar que la clase main se encargue de inicializar un objeto, este solo se encargara de recibir un dato y lo demás no tendrá idea de lo que pasa, el solo recibirá un objeto o un dato. Habrá otras clases que se encarguen de crear los objetos o hagan la lógica de negocio.

Esto es difícil de comprender, es algo que se tendrá que llevar a la práctica ya que todo es muy abstracto y si no esta uno familiarizado con lo que se está realizando es fácil perderse en el proceso.

Adelante intentare plasmar como es que funciona la inyección de dependencias.



Una explicación de lo siguiente seria que main recibe la información, esta llama a el inyector, este se va a encargar de crear el objeto ticket y el objeto película, pero se lo inyectara a ticket y este regresara a main como forma de objeto, inyector solo fue el medio para crear nuestros objetos que deseábamos.

De igual manera inyector podría crear una variedad de objetos y los inyectaría a otros si es necesario.

Un ejemplo siempre seria yo le digo a una persona que guarde una caja en otra caja y ese ultima me la regresa, obtendré las cosas que contengan en esas dos cajas, pero yo nunca guarde una caja, solo se la pedí.

Arquitectura MVC

MVC es una propuesta de arquitectura de software para separar el código a través de diferentes responsabilidades, manteniendo diferentes capas encargadas de realizar tareas distintas y poder separar un proyecto por módulos o equipos de trabajo.

Su principal fundamento es la separación del código por tres capas llamas **Model**, **View**, **Controller**. A pesar de que MVC tiene bastante años su uso en proyectos sigue presente y mas por todos los FrameWorks que emplean el diseño para la arquitectura de una página Web.

Model

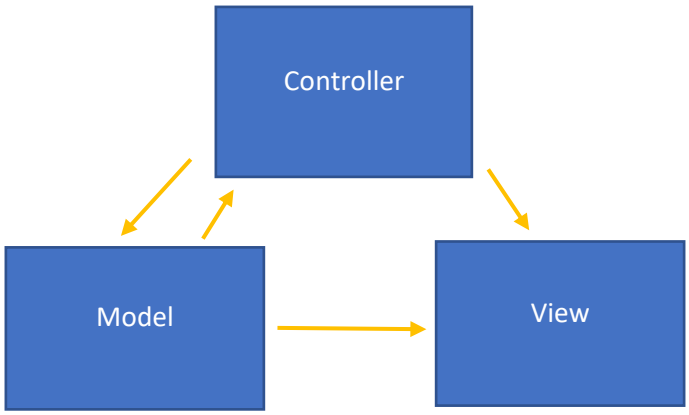
En esta capa es donde se manejan los datos, normalmente es el vinculo que acarreara los datos, podría contener funciones, querys o maneras de acceder a las bases de datos, pero actualmente hay mejores métodos para acceder a ella y se recomienda mejor ocupar el motor de persistencia de SQL esto permite trabajar con abstracción.

View

En las vistas solo contendrá código que nos permita mostrar la interfaz al usuario y este código normalmente es HTML, PHP, CSS, JS. Generalmente aquí es donde se trabaja datos, pero sin acceder a ellos, igual se diseña como es que se quieren mostrar los datos que obtengamos

Controller

En él se tendrá lo necesario para responder a las acciones y peticiones que solicite la aplicación, en pocas palabras es una capa la cual nos sirve como medio de comunicación entre el Model y el View, pero no se encarga de manipular datos de manera directa, ni mostrar ningún tipo de dato.



Final

La palabra reservada final es un modificador de acceso con usos múltiples, todo depende de donde sea aplicado, anexare un programa donde muestro los distintos tipos usos de final.

En un primitivo: Hace que este valor sea constante y no pueda ser modificado posteriormente de haber sido declarado, normalmente a estos se les conoce como Constantes estos normalmente de deberán definir una vez sean declarados o si es que deseamos podría ser iniciado en el constructor si es que lo tenemos como un atributo de la clase.

En un método: Hace que cualquier clase que quiera ocupar este método no podrá utilizarlo ya que no se podría hacer un Override y este no será posible, generando un error.

En un Objeto: Al declarar un objeto como final este no podrá cambiar su variable de referencia.

En una clase: Al definir una clase final no podría heredar sus atributos, haciendo imposible realizarle un extends. Otro motivo para utilizar el final en una clase es para hacer un objeto inmutable.

Exceptions

El manejo de excepciones es algo que nos ayudara al momento de crear un código a prueba de fallas comunes, en java contamos con dos tipos de exceptions las Checked y las Uncheked.

Uncheked: Son toda aquella excepción que el IDE no identifique y solo serán visibles al momento de la ejecución del programa a pesar de que no nos marque ningún tipo de error solo podríamos visualizar el error corriendo nuestro programa y es por lo mismo que son llamadas Runtime Exceptions.

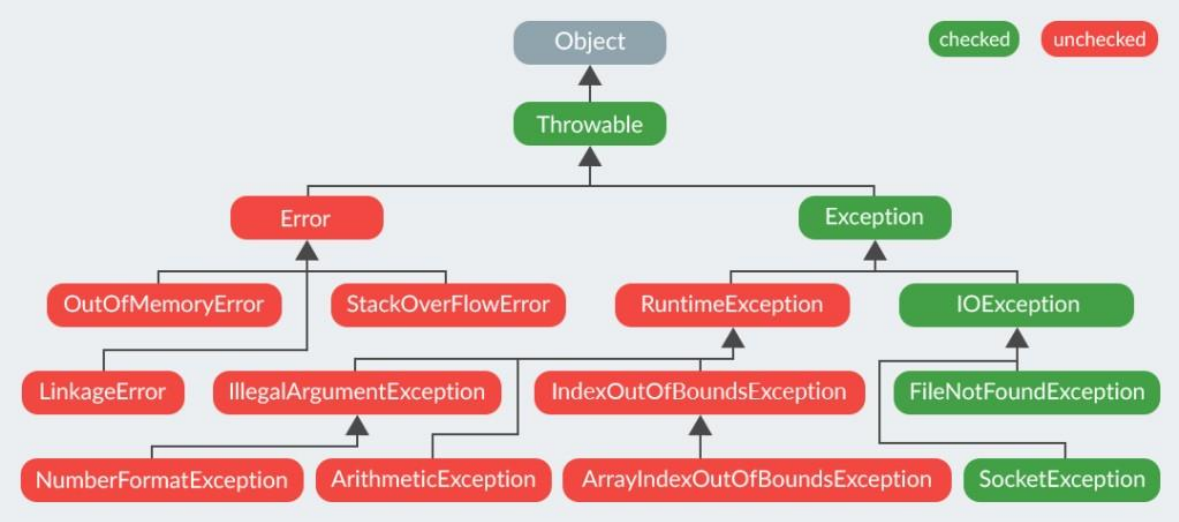
Checked: Estas son más fáciles de identificar y normalmente nuestro IDE nos apoyara si es que requerimos utilizar el manejo de excepciones.

En el ramo de las excepciones hay uno que no podremos controlar y simplemente pasara si es que ocurre un error. Se sabe que hay igual diferente tipo de errores, pero no podremos hacer nada por controlar o seguirle la pista a ese error en concreto.

Para hacer uso de las excepciones debemos meter nuestro código en un bloque llamado try catch donde se podrá hacer visible esta exception y nos será posible cacharla para poder visualizarla y hacer algún cambio o modificación requerida, de

igual manera nos sirve para que a pesar de que ocurra esta exception nuestro programa no deje de funcionar y siga su flujo.

En este bloque de try catch contamos con bloque llamado finnally y aquí podremos ejecutar código que nos importe ejecutar, aunque arroje la excepción, normalmente el uso de este bloque es usado para cerrar objetos que tengamos en memoria.



Procesos Sincronos y Asincrono

Los métodos **síncronos** bloquean la ejecución de los subprocesos hasta que el cliente recibe una respuesta del servicio.

Los métodos **asíncronos** terminan de ejecutarse inmediatamente, devolviendo el control al subproceso que realiza la llamada sin esperar una respuesta.

Actualmente se busca ocupar procesos asíncronos ya que nos permitiría realizar varias peticiones al mismo tiempo, plantea el no limitar la cantidad de procesos que puedas realizar con las limitantes del servidor.

Realizar procesos asíncronos hace que la experiencia del usuario sea mas agradable ya que no hacemos esperar a que este disponible su petición del usuario

Lambdas

Las expresiones lambdas son funciones anónimas, no es necesario que sean instanciadas, solo es necesario definir el comportamiento de lo que se quiere.

Una expresión lambda se define de la siguiente manera x -> {Cuerpo}, la x son los parámetros que se requieren el cuerpo de la lambda se define el comportamiento, si se requiere más de una línea se debe de colocar llaves para delimitar el bloque de Código.

Las expresiones se pueden definir por consumidores, proveedores, funciones Unarios y binarios, predicados.

Todo esto nos permite adentrarnos y aplicar la programación funcional, haciendo nuestro código mas limpio y preciso.

En los siguientes programas se emplean todo lo visto anteriormente, en combinación y casos únicos.

Donde se emplean varios temas vistos es en la Inyección de Dependencias, se empleo , enums, clases abstractas, final, manejo de excepciones, arrays, clases anónimas.