Nombre: Ignacio Emmanuel Isaac Medina
Fecha: 02-Dic-2022
Evaluación escrita Semana 2

1. Explica que es la inyección de dependencias

Se trata de un método para lidiar con el alto acoplamiento entre clases, el cual es un problema presente en un mal diseño de sistemas que utilizan el paradigma de programación orientada a objetos. En la inyección de dependencias una entidad externa coordina los objetos en el sistema, de tal modo que ahora la clase implementado ya no se encarga de crear el objeto, solo sabe de su existencia.

Esto es bueno ya que cualquier modificación se realiza sobre el inyector, en lugar de la clase misma en la que se pretende implementar esta metodología.

Existen 3 maneras de realizar la inyección de dependencias:

- Inyección por variable.
- Inyección por *setters*.
- Inyección por constructor.

El diagrama de la Figura 1 muestra un esquema general de la inyección por dependencias:

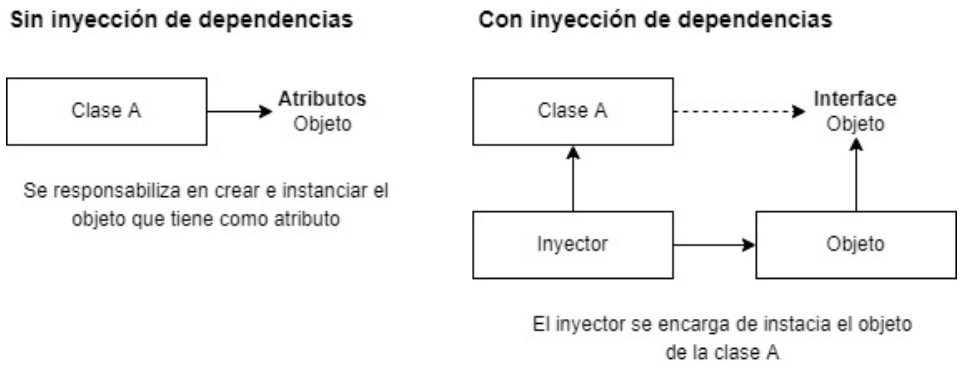

Figura 1.- Diagrama de la inyección de dependencias.

El código para esta pregunta se encuentra dentro del paquete *injection_dependency*, y simula un sistema de gestión de bancos, cada tipo de banco contiene a su vez una cuenta, todas las cuentas tienen nombre de usuario y número de usuario, sin embargo, dependiendo del banco puede, o no, tener atributos extra:

- BBV: RFC.
- Citibanamex: Salario.
- Banorte: Nada.

Al final se imprime el nombre del banco y se llama el método de Bienvenida, que imprime un mensaje y la información de la cuenta que posee. Para el caso de Citibanamex, si el salario supera los $2000.0, se imprime un mensaje señalando que la cuenta puede pedir un crédito.

2. Diagrama y explica una arquitectura web usando MVC.

Una arquitectura Model-View-controller (MVC), realiza una separación del código en 3 partes, cada una responable de realizar una serie bien definida de tareas:

Nombre: Ignacio Emmanuel Isaac Medina
Fecha: 02-Dic-2022
Evaluación escrita Semana 2

- Modelo: Se encarga de hacer el manejo de los datos y se los puede enviar al controlador y la vista, según sea necesario.
- Vista: Es la responsable de construir la lógica de presentación. El hecho de construir, no significa que tenga que ejecutarlo o sea directamente el responsable de mostrar la presentación-
- Controlador: Es el encargado de escuchar la petición y decide quién lo va a atender, puede delegar la responsabilidad al modelo o a la vista. Su comportamiento se asemeja al del director de una orquesta, no hace lógica de negocio.

La Figura 2 muestra un diagrama sencillo del patrón MVC.

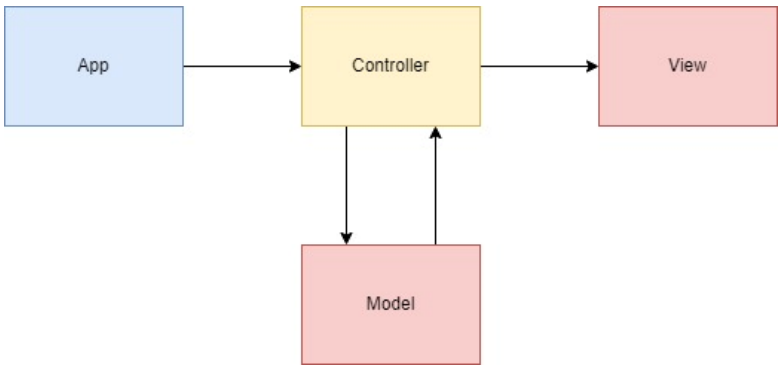

Figura 2.- Esquema básico del modelo MVC.

Cuando nos encontramos en un sistema web, la complejidad del sistema entero crece, debido a que se involucran diferentes tecnologías. Tal y como se muestra en la Figura 3. Los dispositivos se conectan al servidor mediante su navegador, el cual es capaz de ejecutar código JS y renderizar páginas creadas con HTML y CSS. Mediante una conexión HTTP/HTTPS y utilizando los verbos de este protocolo hace una petición al servidor donde se tiene alojada la lógica de la página web, el cual corre uno de los lenguajes de *backend* (C#, Java, Python, etc.). En el *backend* se puede tener múltiples arquitecturas y patrones de diseño, sin embargo, para este ejemplo se tiene una arquitectura monolítica implementando un patrón multicapa MVC. La capa que contiene este patrón se comunica con la capa de lógica de negocios, a su vez, esta se conecta con la capa de base de datos que hace uso de alguna de los dialectos SQL disponibles.

Nombre: Ignacio Emmanuel Isaac Medina
Fecha: 02-Dic-2022
Evaluación escrita Semana 2

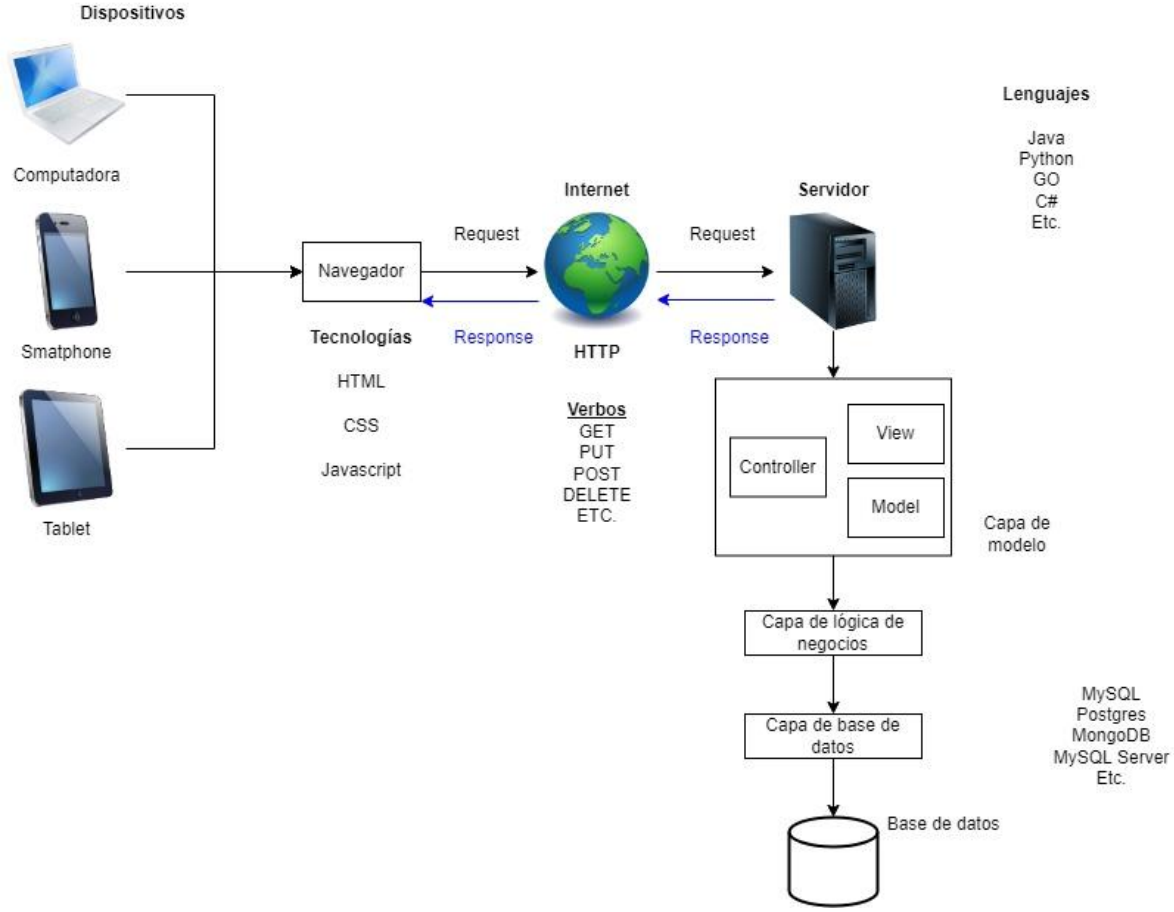

Figura 3.- Sistema web utilizando una arquitectura MVC.

3. Explica los diferentes tipos de excepciones.

Las excepciones son objetos que Java, que son creados cuando una situación anormal sucede en el programa. Pueden ser creadas por la JVM o por nuestro programa. Las excepciones heredan de la clase *Throwable* y son de dos tipos:

- Runtime Exceptions (Unchecked): El compilador no checa si el método donde se está arrojando maneja la excepción, son errores que suceden durante el tiempo de ejecución del programa, de ahí su nombre. Usualmente están asociados a errores lógicos o uso incorrecto de las APIs. Algunos ejemplos son: ArrayIndexOutOfBoundsException, ClassCastException y IndexOutOfBoundException.
- Others Exceptions (Checked): La JVM no te permite compilar el programa, por ese motivo también se le conoce como excepción de tiempo de compilación, por este motivo son más sencillas de detectar. Algunas de las excepciones que suceden son: IOException, SQlException, FileNotFoundException e InvocationTargetException.

Adicionalmente, hay otro tipo de objeto que hereda de la clase *Throwable* llamados errores, estos indican un problema seria que una aplicación no debe de intentar hacer el *catch*. La Figura 4 presenta el diagrama completo de como se relacionan las excepciones en Java.

Nombre: Ignacio Emmanuel Isaac Medina
Fecha: 02-Dic-2022
Evaluación escrita Semana 2

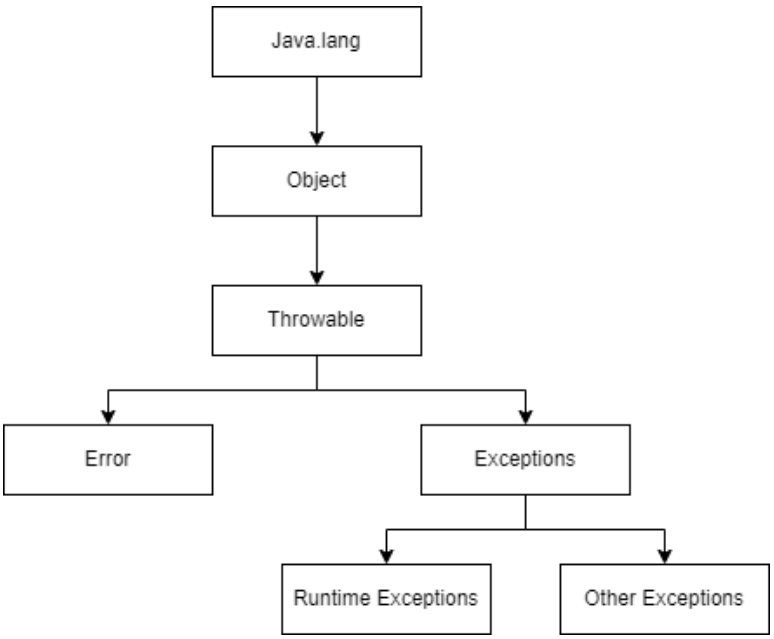

Figura 4.- Tipos de Excepciones disponibles en Java.

4. Explica los 4 propósitos del final con código.

El código se encuentra en el paquete *final_usages* dentro del directorio asociado a

- Final en una variable primitiva: Hace que la variable se vuelve constante, por lo tanto, su valor no puede ser cambiado una vez inicializado.
- Final en una variable de tipo objeto: impide que se pueda cambiar la referencia a la que apunta la variable de referencia.
- Final en un método: a los métodos de este tipo no se les puede realizar un cambio en su lógica, en otras palabras, no se les puede aplicar un *Overriding*.
- Final en una clase: las clases de este tipo no pueden ser heredadas

5. Exponer un código con multicatch, try with resources.

El código se encuentra en los paquetes *multicatch_usage* y *try_resource_usage*.

Para el ejemplo del multicatch estoy provocando 3 tipos diferentes de excepciones:

- División entre cero (ArithmeticException)
- Utilizar un índice de un arreglo fuera de los límites (ArrayIndexOutOfBoundsException).
- Llamar un atributo de una variable de referencia nula (NullPointerException).

Mientras que para el try-resource intentaré abrir un documento que no existe. En este caso estamos ignorando las 2 posibles excepciones propias al manejo de archivos: IOException y FileNotFoundException. Estas dos son de tipo *checked*, por lo que se le tiene que dar un tratamiento.

6. Explica la diferencia entre los procesos síncronos y asíncronos.

Nombre: Ignacio Emmanuel Isaac Medina
Fecha: 02-Dic-2022
Evaluación escrita Semana 2

Los procesos síncronos y asíncronos difieren respecto a los tiempos de ejecución. En un proceso síncrono se tiene que realizar la ejecución de los bloques de código de manera secuencial, debido a ello se convierte en un proceso más tardado. Por otra parte, los procesos asíncronos se realizan de forma paralela, brindado la oportunidad de reducir los tiempos de ejecución de un programa de manera significativa. Otra diferencia importante es la cantidad de recursos, ya que, debido a su naturaleza paralela, un proceso asíncrono requiere de una mayor cantidad de recursos, mientras que un proceso síncrono se puede realizar en dispositivos con características más limitadas.

7. Realizar ejemplo de lambdas

El código se encuentra dentro del paquete *lambdas_example*.

Se realizó un sistema para una escuela, con los siguientes componentes:

- Escuela: Interfaz funcional para definir los posteriores filtros.
- Alumno: Clase que contiene la información de los alumnos de una escuela, entre los que se tiene: nombre, matrícula, edad, si está activo, peso y altura.
- Dirección: Clase que define y ejecuta las funciones lambda.

Se hicieron 2 tipos de operaciones:

- Operaciones de filtrado con una interface funcional personalizada para filtrar los alumnos según su edad (mayores a 20), si están activos y si su matrícula empieza con “1.”
- Operaciones de cálculo de IMC del grupo, mediante la interface *Function* de Java.