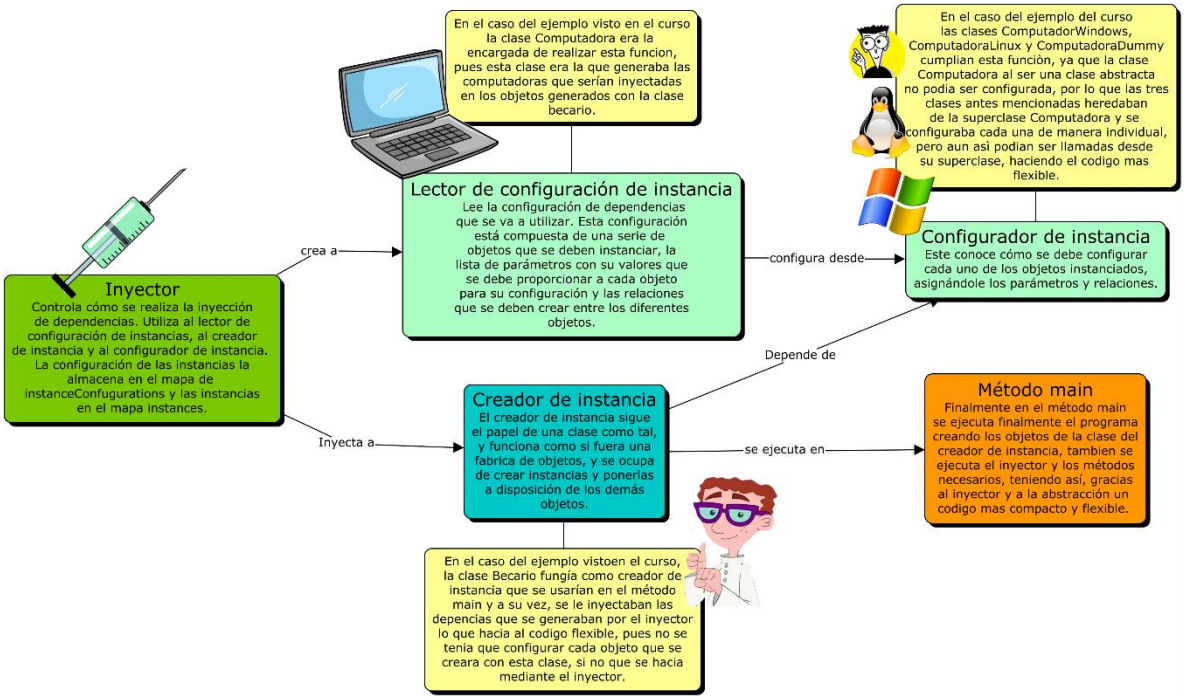
Examen semana 2

Manuel Mauricio Chulim Alamilla

Pregunta 1.- Explica que es la inyección de dependencias

-Con diagrama

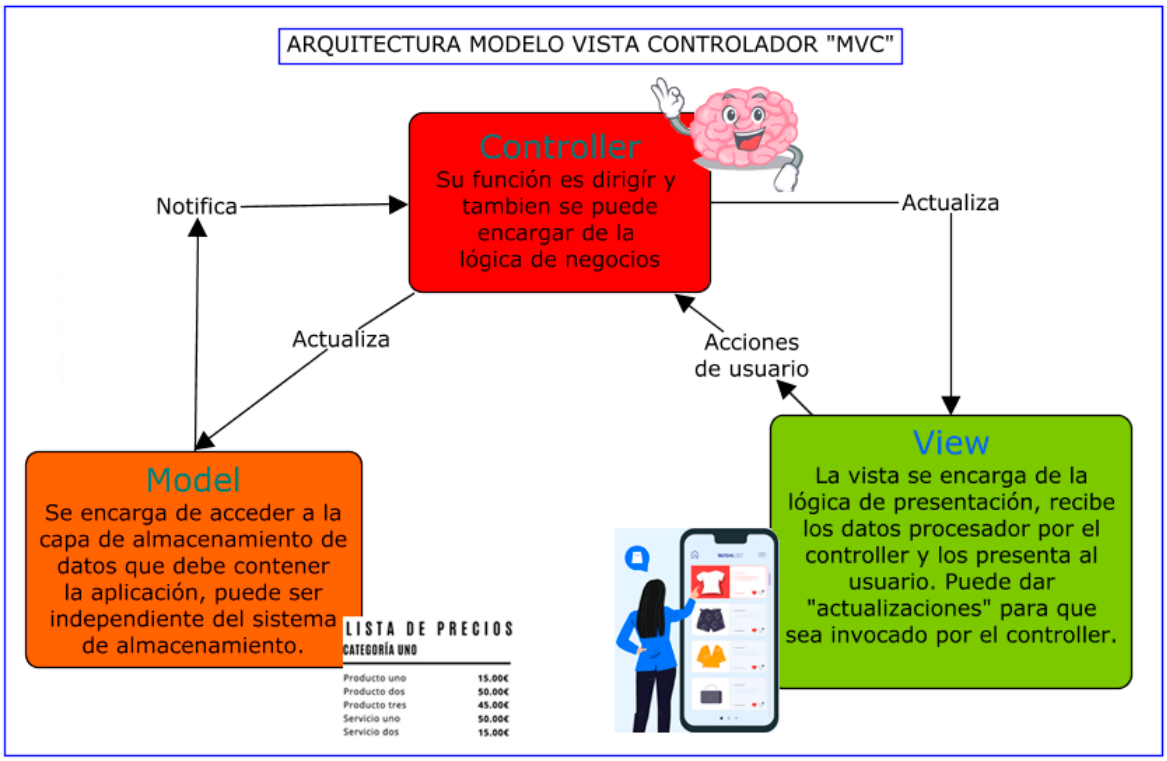


-Con código

El código se puede encontrar en la carpeta de exámenes semana 2, en el paquete de nombre “**inyector.de.vehiculos**”.

En dicho código se tiene a la clase abstracta “AutoF1” que es heredada por las clases: “Ferrari”, “RedBull”, “Mercedes”, “Alpine”, para son inyectadas mediante la clase “**InyectorEquipo**” a la clase “Piloto” según sea el caso, haciendo uso de la clase enum “Escuderia” y del inyector. Finalmente, en la clase “**Principal**” se crean los objetos tipo Piloto a los que se les inyecta su escudería (es decir su auto).

Pregunta 2.- Diagrama y explica una arquitectura web usando MVC.



La arquitectura web MVC es un patrón de diseño de software usado para poder implementar datos, lógica de control de negocios y una interfaz gráfica para el usuario. Un factor muy importante en este patrón de diseño es el hecho que se tienen tres “partes” que componen el diseño y es de suma importancia la división o delegación del trabajo.

Al tener una división de trabajo, se logra que al tener que realizar mantenimiento a la aplicación programada, se pueda realizar de forma más eficiente.

La forma de división del trabajo que propone el modelo MVC, es la siguiente:

Modelo. El modelo se encarga de enlazarse con la base de datos que se requiera para poder ejecutar una aplicación, define los datos que se van a usar en la aplicación, y si los datos que contiene son modificados notifica a la vista.

Para entender el trabajo del Modelo (o Model), se puede poner como ejemplo una aplicación que se dedique a la venta de ropa en línea. En este caso el Modelo tomaría los datos de la ropa que va a vender en la aplicación de algún sitió como podría ser MySQL, donde obtendría precios, modelo de ropa, tallas, etc. Y tendría estos datos disponibles para ser manipulados por el controlador o bien para pasárselos a la vista.

Vista. La vista se encarga de la lógica de presentación hacia el usuario, define como deben de mostrarse los datos en la presentación, es como su nombre lo indica la

forma en que se observan mediante una interfaz gráfica la tienda en línea para el usuario.

En el caso de la aplicación de una tienda en línea de ropa, la vista seria la forma en que se presenta la tienda de ropa en línea, por ejemplo, se podría presentar únicamente como una lista de las ropas que se pueden comprar con un botón de agregar, o bien podría ser una página donde se muestren las fotos de las ropas que están en venta, con los precios, descuentos, botones de agregar, botones para ver más ropas, un buscador, etc.

Controlador. En el caso del controlador, se encarga de manejar la lógica de negocios que se necesite en la aplicación, así mismo se encarga de delegar al modelo o bien a la vista las partes del trabajo que estos deban de realizar. Una analogía del controlador (o controller), es que funciona como un cerebro para la aplicación, a la vez que realiza el código necesario entre los datos y la interfaz gráfica.

Continuando con el ejemplo de la página web de una tienda de ropa en linea. Si un usuario selecciona en la interfaz gráfica comprar dos prendas, el controlador ejecuta las acciones que se tienen programadas en los botones de la interfaz gráfica y a su vez, manipula al Modelo para que se obtengan los datos necesarios para la vista, y de ser necesario realiza las acciones necesarias para entregar datos a la vista que sean manipulados por el controlador.

El objetivo de este tipo de arquitectura web MVC es como se menciona anteriormente, que se pueda dar un mantenimiento más eficiente a la aplicación, pues se requiere hacer un cambio a alguna de sus partes, por ejemplo agregar más ropas o colocar nuevos precios solamente se tendría que dar mantenimiento al Modelo, sin tocar el Controlador ni la Vista, y de manera similar si se quiere tener otro tipo de interfaz gráfica solamente se le daría mantenimiento a la Vista sin tocar al Modelo ni al controlador. Si se tuviera una aplicación sin esta división de trabajo, el mantenimiento sería más complicado, pues si se quisiera cambiar solo los datos a usar también se tendría que modificar la interfaz gráfica y la lógica de negocios, cosa que no sucede en la arquitectura MVC.

Pregunta 3.- Explica los diferentes tipos de excepciones.

En java existen tanto los errores como las excepciones, en el caso de los errores, no hay forma de manejarlos, un ejemplo de un error seria que un disco duro de donde un programa debe de obtener datos esta dañado, por lo que no se puede acceder a dicha información, en este caso se lanzaría un erro que no se puede tratar. Mientras una excepción, se da una situación que puede ser resuelta, tratada o bien avisar al usuario mediante algún tipo de advertencia.

Ahora bien, en el caso de las excepciones, las hay de dos tipos, las RuntimeExceptions y las OtherExceptions (o también llamadas IOException). Cabe mencionar que las excepciones son errores en tiempo de compilación.

Las **RuntimeExceptions o Excepciones No Comprobadas** son un conjunto de las excepciones que pueden tener lugar durante el proceso de ejecución de un programa, pero no necesariamente se tienen que tratar o capturar (todas las demás excepciones deberán ser tratadas o capturadas).

Un ejemplo de las RuntimeException sería cuando se solicita un índice de un arreglo y dicho arreglo no contiene ese índice solicitado, por lo que se ejecutaría el ArrayIndexOutOfBoundsException, que nos está indicando que está ocurriendo la situación ya mencionada; otro caso sería el dividir entre cero, que arrojaría otra RuntimeException.

OtherExceptions o también llamadas Excepciones Comprobadas, son ajenas al programador, por ejemplo, cuando un programa debe de buscar un archivo en cierta ubicación, sin embargo, algo o alguien ajeno al programador o al programa en si elimina dicho archivo, por lo que no se podría encontrar el archivo antes mencionado.

Sin embargo, cuando se tienen Excepciones Comprobadas, java dispone de ciertas maneras de tratarlos, para que se pueda ejecutar el programa, aun si se tuviera una excepción, esto con el tan conocido “try” y “catch”. Esto nos sirve para intentar ejecutar ciertas líneas de código con el “try” (que probablemente falle) y en caso de que se tenga una excepción, entonces ejecutará otra línea de código “catch”.

Un ejemplo puede ser el siguiente: Se desea abrir un archivo .csv, entonces se pondría dentro de un “try” una línea de código que habrá el archivo .csv, y en un “catch” se pondría una línea de código que envíe una alerta al usuario que no se encontró el archivo .csv solicitado. Por lo que en caso de encontrar el archivo .csv, este se abriría y no se enviaría la alerta del “catch”; pero en caso de no encontrar el archivo solicitado el “try” no se ejecutaría, y en su lugar se enviaría la alerta diciendo que no se encontró el archivo .csv solicitado.

4.- . Explica los 4 propósitos del final con código.

El código se puede encontrar en la carpeta de exámenes semana 2, en el paquete de nombre “**examen.Final**”, en el código se encuentran las explicaciones de los cuatro diferentes propósitos del final, esto es: en variable primitiva, en variable de referencia, en un método y en una clase.

5.- Exponer un código con multicacth, try with resources.

El código se puede encontrar en la carpeta de exámenes semana 2, en el paquete de nombre “multicathYAutoCloseable.

En dicho paquete se encuentran tres clases que son excepciones programadas las cuales son “CuentaBloqueada”, “DatosErroneos” y “FondosInsuficiente”.

También se encuentra la clase “PrincipalTransferenciaMulticatch” que como su nombre lo indica es una clase en donde se implementa únicamente el multicatch haciendo uso de las tres clases que son excepciones programadas.

Así mismo se puede encontrar la clase “Transferencia” que sirve como generador de objeto en la clase “**PrincipalTransferenciaMulticatchYAutoCloseable**”, en este caso la clase principal ejecuta un código que incluye tanto multicatch, como try with resources. Esto lo hace empleando a la clase “Transferencia”, que es en donde se implementa el AutoCloseable.

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

Los métodos síncronos o “sync” evitan la ejecución de los subprocesos hasta que el cliente recibe una respuesta del programa. Un código síncrono es aquel código donde cada instrucción espera a la anterior para ejecutarse.

Los procesos asíncronos o “async” terminan de ejecutarse inmediatamente, devolviendo el control al subproceso que realiza la llamada sin esperar una respuesta. Un código asíncrono no espera a las instrucciones diferidas y continúa con su ejecución. Por lo general la asincronía permite tener una mejor respuesta en las aplicaciones y reduce el tiempo de espera del cliente.

Algo importante que difiere entre ambos tipos de procesos, es el hecho de si el proceso es bloqueante o no bloqueante. Lo anterior se refiere a que, si se tiene un hilo programación en el caso del síncrono, se tendría que ejecutar el código paso por paso y no se podría continuar en el siguiente paso del código a menos que se termine de ejecutar el paso que se está procesando en el código; esto es, el hilo esta bloqueado hasta y se debe ejecutar paso por paso hasta finalizar el programa.

Ahora bien, en el caso de la programación asíncrona se ejecutan los diferentes procesos de código y al iniciar una tarea del proceso que tarda mucho en finalizar, se coloca esa tarea en un segundo plano mientras se continua con la ejecución del resto de la línea de código y realiza lo mismo en el caso de encontrarse con otro proceso que tarde en finalizarse. Ahora bien, cuando estos procesos tardados logran finalizarse se notifica que los procesos han acabado y se colocan en el hilo de ejecución de nuevo.

Una analogía que se hace con estos métodos es el cocinar, pues si se maneja preparar un pastel de manera síncrona, se tendría que hacer cada paso de uno en uno, lo que haría que el proceso sea tardado. Mientras que, si se cocina un pastel de manera asíncrona, el proceso tomará menos tiempo, pues si se pueden hacer dos pasos a la vez (como por ejemplo precalentar el horno a la vez que se amasa la harina) esto ahorrara tiempo en el proceso.

7.- Realiza un código aplicando lambdas

El código se puede encontrar en la carpeta de exámenes semana 2, en el paquete de nombre “lambdas”.

En dicho paquete se puede encontrar la clase “Piloto” que es la que sirve para generar un objeto, también se puede encontrar la clase “PredicadoPiloto” que es la que genera un operador funcional para la clase piloto. Ahora bien, ambas clases son usadas en la clase “**PrincipalSinTest**”, que es en donde se hace uso de las funciones lambdas usando el “PredicadoPiloto”.

Finalmente se puede encontrar la clase “**PrincipalTest**”, que en este caso también se usa la clase “Piloto” para generar los objetos tipo Piloto, pero en este caso se hace uso de los operadores funcionales que nos ofrece java como lo son el “Predicate”, el “IntPredicate” y el “DoublePredicate”.