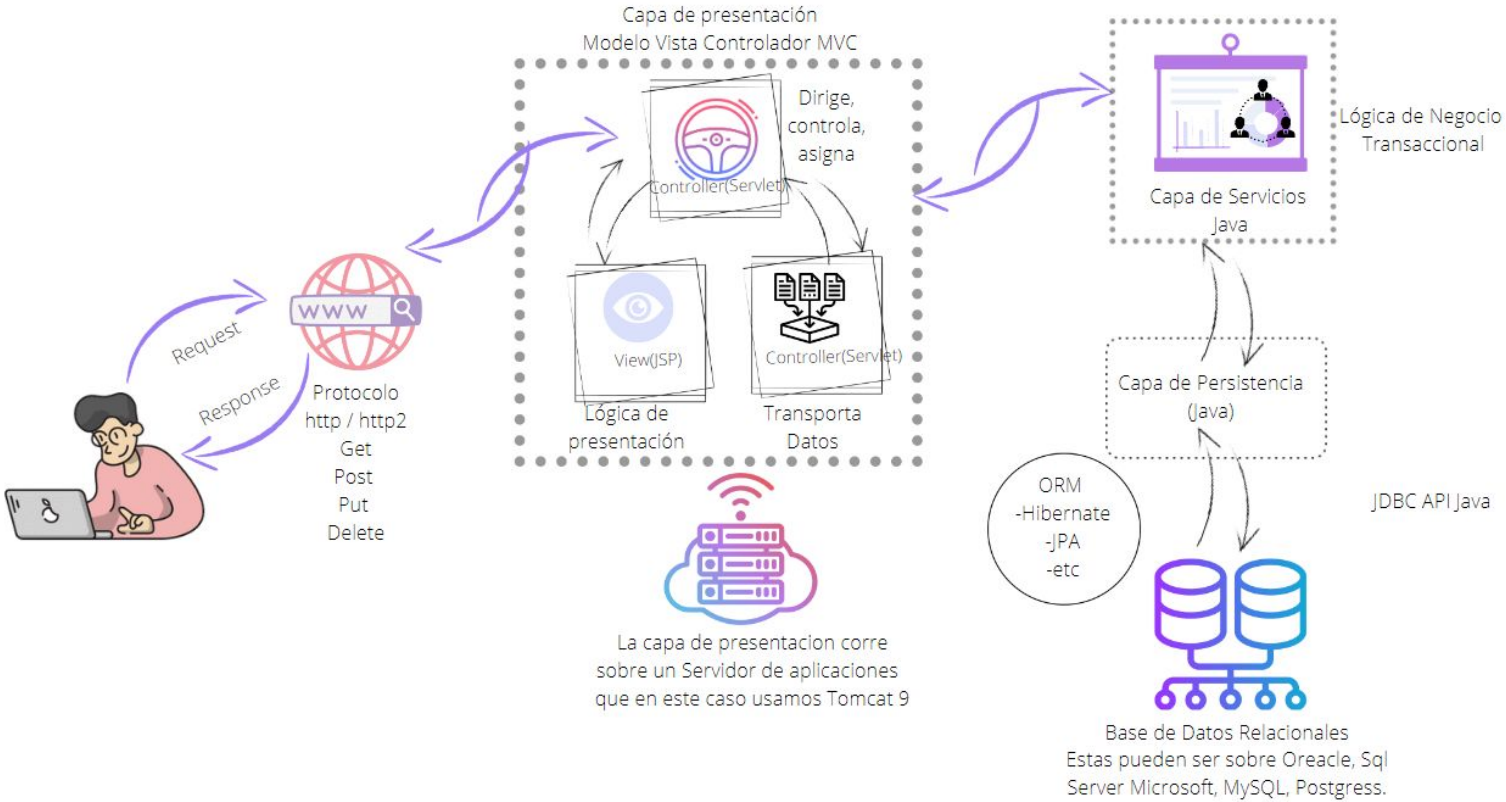
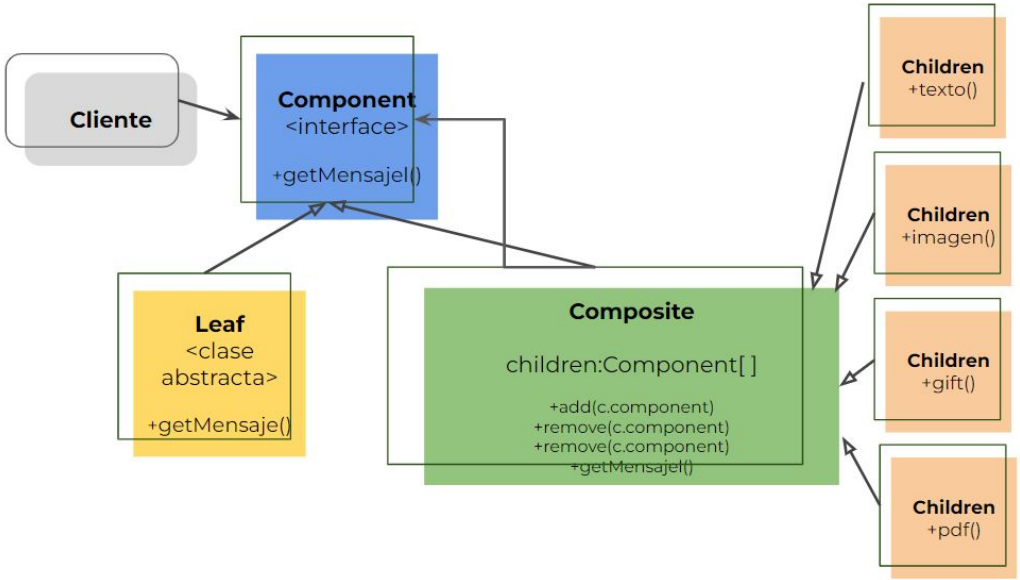
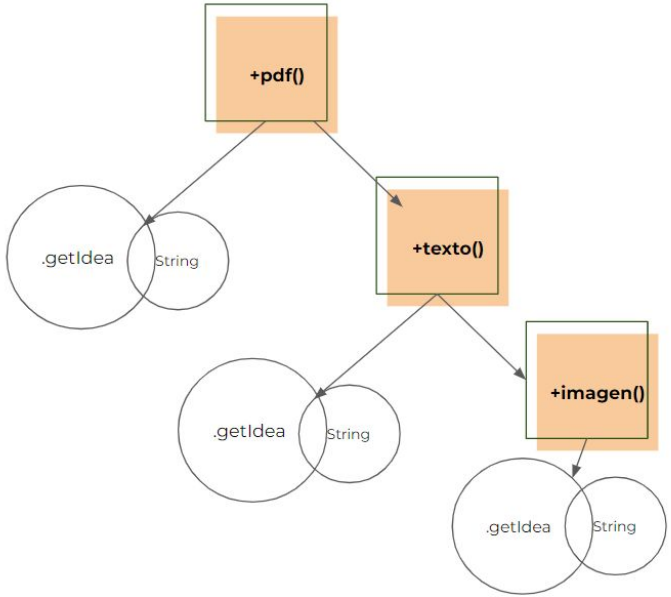
Examen 2_ ANA LAURA VÁZQUEZ SOLACHE

1. Dibuja el diagrama de interacción de una aplicación web.



2. Desarrolla un código donde apliques el patrón composite con pruebas unitarias





3. Resumen de cómo aplicar Scrum.

La base del Scrum son los tiempos definidos y el trabajo en equipo.

En esta **metodología ágil**, tenemos que identificar a los actores principales, siendo estos el **Product Owner** el cual es el encargado de la visión del producto, producir, lograr, toma en cuenta riesgos y recompensas, al **product manager** que es el responsable de vigilar el cumplimiento de la metodología Scrum dentro del equipo así como el seguimiento de los Daylis, encargado de eliminar los retrasos y finalmente el **equipo de desarrollo** los cuales no deben ser mayores a 9 personas.

La guía del Team Scrum es la **bitácora del producto**, que está conformada por una lista de lo que debe llevarse a cabo para concretar el proyecto, está se encuentra organizada por orden de prioridad, deben planearse así mismo los **Sprints**, fechas en las cuales se designan alcances y revisiones del proyecto con temporalidades no mayores a un mes.

Es importante en esta metodología hacer visible el trabajo, para llevar un seguimiento activo, por lo general se lleva a cabo con una **tabla Scrum**,en la cual se indican tareas pendientes, en proceso y terminadas. Este seguimiento es a través de **Daylis, o Scrum Diarios** en los cuales cada integrante del equipo responde a las preguntas siguientes:

1. ¿Qué hiciste ayer?
2. ¿Qué harás hoy?
3. ¿Hay algún obstáculo?

Terminando los **Sprint se lleva a cabo una revisión** con cliente e interno para mejorar de forma continua el proyecto, en los cuales se demuestra los alcances obtenidos así como realizar una **retrospectiva**, en el cual a nivel interno se evalúa cuáles fueron son las áreas de oportunidad del equipo, de mejora y de ajustes.

4.Explica el tema de las excepciones.

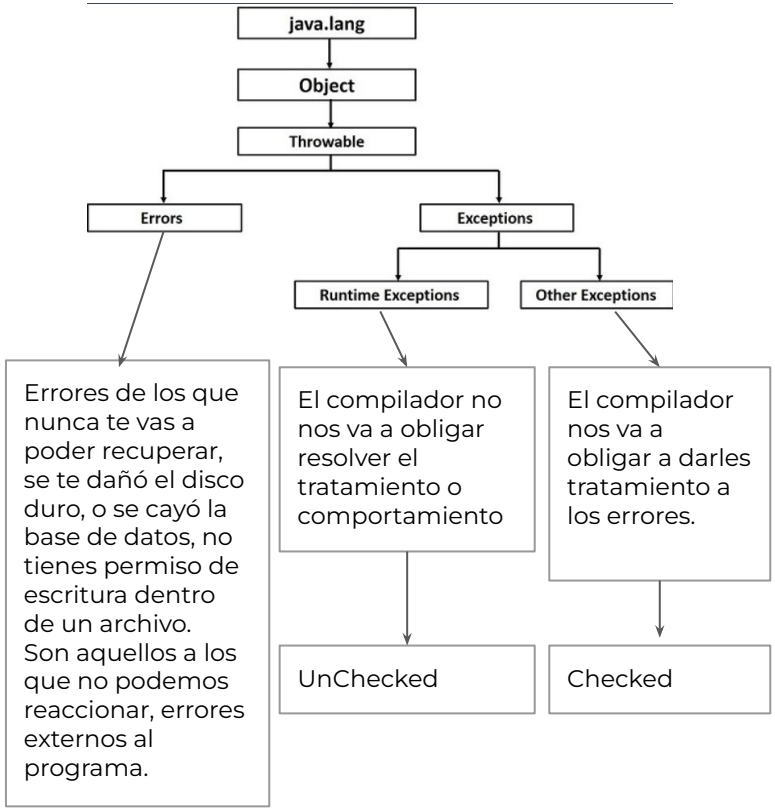
Las excepciones se heredan del paquete java.lang, de la clase object, existen las **Runtime Exceptions y Other Exceptions .**

Dentro de las Runtime también conocidos como **Uncheked,** el compilador **NO** nos obliga a darles tratamiento pero los podemos cachar, con un try & catch.

Dentro de las Other exceptions o también conocidas como **Checked** el compilador **SI** nos obliga a darles tratamiento pero al igual que el anterior podemos insertarlo dentro de un **try & catch** o bien **“throw”** delegarle la responsabilidad al método o clase, se puede hacer sobre la main pero no debería hacerse, y tampoco cachar solo la excepción ya que es muy genérico, hay que darles tratamiento individual a cada una.

Dentro de los try & catch es opcional poner el **“Finally”**, y este es un código que se ejecuta te equivoques o no te equivoques, hay un método en el cual imprime en consola la excepción el cual es .printStackTrace().

Cuando consideramos servicios web, y conexiones a base de datos al trabajar con DataSource que se resuelve en tiempo de ejecución, podemos utilizar un **try-able resource** el cual funciona metiendo el código dentro de los paréntesis del try, para que sean cerradas de forma automática, la condición es que todas las clases metidas aquí deben implementar una interfaz **AutoClosable.**



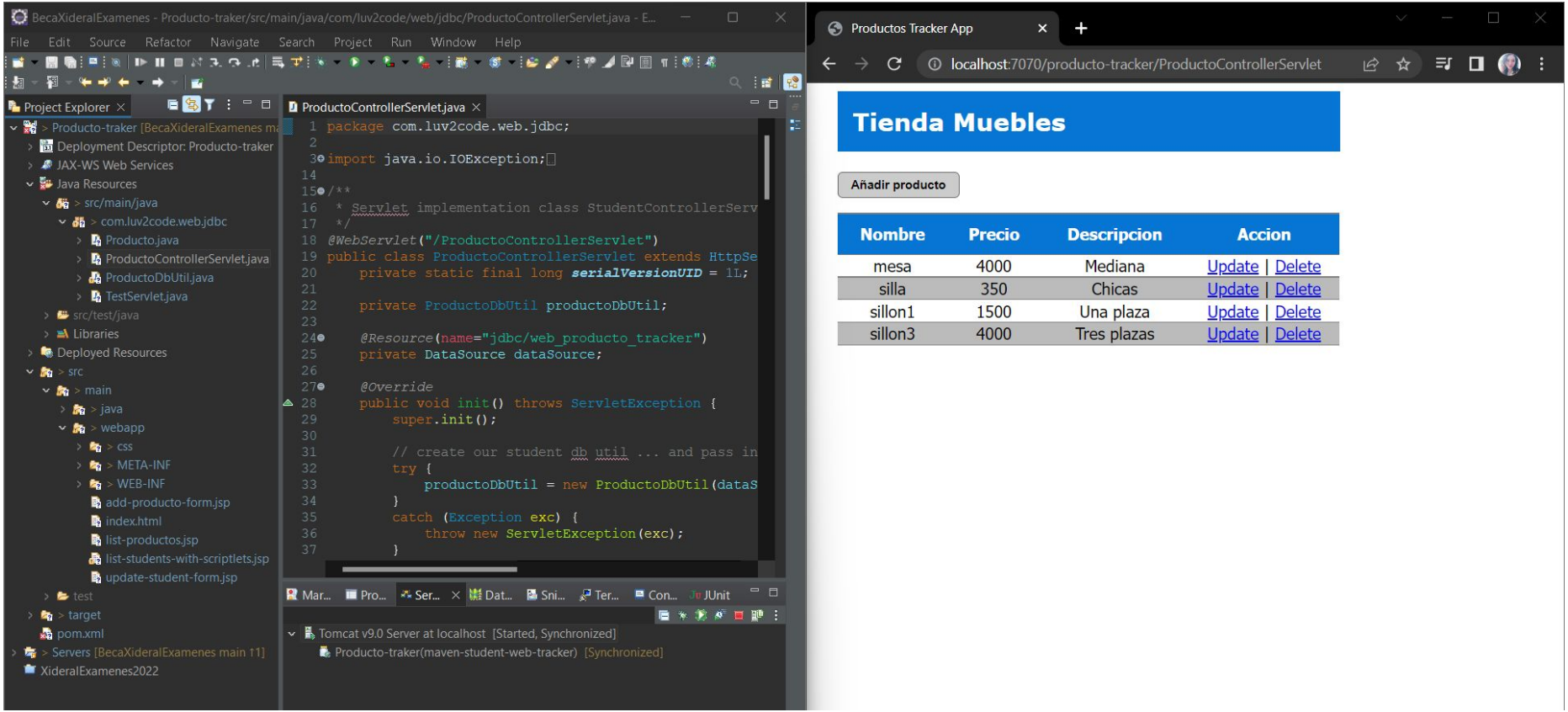
Ejemplos Exceptions

Cheked - Other Exceptions

ClassNorFoundException
InterruptedException
FileNotFoundException
Aquí el compilador nos obliga a darles tratamiento.

UnCheked - Runtime Exceptions

ArithmeticException
Cuando llevamos a cabo una operación aritmética no válida por el código
NullPointerException
Cuando la variable de referencia no apunta a un objeto y tratamos de regresar en consola o hacer algo con el objeto.
ArrayIndexOutBoundsException - sale en los arrays cuando tratamos de indicar un índice no existente


BecaXideralExamenes - Producto-traker/src/main/java/com/luv2code/web/jdbc/ProductoControllerServlet.java - E...
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Producto-traker [BecaXideralExamenes ma
Deployment Descriptor: Producto-traker
JAX-WS Web Services
Java Resources
src/main/java
com.luv2code.web.jdbc
Producto.java
ProductoControllerServlet.java
ProductoDbUtil.java
TestServlet.java
src/test/java
Libraries
Deployed Resources
src
main
java
webapp
css
META-INF
WEB-INF
add-producto-form.jsp
index.html
list-productos.jsp
list-students-with-scriptlets.jsp
update-student-form.jsp
test
target
pom.xml
Servers [BecaXideralExamenes main 11]
XideralExamenes2022
ProductoControllerServlet.java
1 package com.luv2code.web.jdbc;
3 import java.io.IOException;
15 /**
16 * Servlet implementation class StudentControllerServ
17 */
18 @WebServlet("/ProductoControllerServlet")
19 public class ProductoControllerServlet extends HttpSe
20 private static final long serialVersionUID = 1L;
22 private ProductoDbUtil productoDbUtil;
24 @Resource(name="jdbc/web_producto_tracker")
25 private DataSource dataSource;
27 @Override
28 public void init() throws ServletException {
29 super.init();
31 // create our student db util ... and pass in
32 try {
33 productoDbUtil = new ProductoDbUtil(dataS
34 }
35 catch (Exception exc) {
36 throw new ServletException(exc);
37 }
Mar... Pro... Ser... Dat... Sni... Ter... Con... JUnit
Tomcat v9.0 Server at localhost [Started, Synchronized]
Producto-traker(maven-student-web-tracker) [Synchronized]
Productos Tracker App
localhost:7070/producto-tracker/ProductoControllerServlet
Tienda Muebles
Añadir producto
Nombre
Precio
Descripcion
Accion
mesa 4000 Mediana Update | Delete
silla 350 Chicas Update | Delete
sillon1 1500 Una plaza Update | Delete
sillon3 4000 Tres plazas Update | Delete