Guía, resumen y recolección de errores

1. Conceptos básicos

a. Producto vs Proyecto

Producto y Proyecto son cosas **distintas**. Aunque relacionadas.

Proyecto es un **proceso,** una actividad que realizamos, que empezamos y finalizamos.
Recordaran la definición de *"esfuerzo conjunto para alcanzar ciertos objetivos en base a las tres variables: alcance, tiempo y costo"*.

Producto se refiere más a una **cosa** que al proceso de realizarla.
Sin embargo son conceptos **relacionados,** aunque no siempre de la misma forma. Acá vemos algunas:
Tenemos un único proyecto cuyo objetivo es la construcción de un único producto. En general una versión específica del producto. Ejemplos:

- v1.0.0 = la primer versión
- v1.1 = una siguiente versión que incluye nuevas funcionalidades.
- v2.0 = una nueva versión que incluye un conjunto de nuevas funcionalidades que dan transforma al producto de forma significativa. Generalmente hay un vínculo con la idea de incompatibilidad hacia atrás. Aunque no necesariamente.
- En estos dos últimos casos trabajamos sobre un producto ya existente.

b. ¿Qué es un package?

El paquete (package)

- Los paquetes son una forma de organizar grupos de clases. Un paquete contiene un conjunto de clases relacionadas bien por finalidad, por ámbito o por herencia.
- Los paquetes resuelven el problema del conflicto entre los nombres de las clases. Al crecer el número de clases crece la probabilidad de designar con el mismo nombre a dos clases diferentes.
- Las clases tienen ciertos privilegios de acceso a los miembros dato y a las funciones miembro de otras clases dentro de un mismo paquete.

La palabra reservada import

Para importar clases de un paquete se usa el comando **import**. Se puede importar una clase individual

```
import java.awt.Font;
```

o bien, se puede importar las clases declaradas públicas de un paquete completo, utilizando un arterisco (*) para reemplazar los nombres de clase individuales.

```
import java.awt.*;
```

Para crear un objeto *fuente* de la clase *Font* podemos seguir dos alternativas

```
import java.awt.Font;
```

```
Font fuente=new Font("Monospaced", Font.BOLD, 36);
```

O bien, sin poner la sentencia **import**

```
java.awt.Font fuente=new java.awt.Font("Monospaced", Font.BOLD, 36);
```

Normalmente, usaremos la primera alternativa, ya que es la más económica en código, si tenemos que crear varias fuentes de texto.

Se pueden combinar ambas formas, por ejemplo, en la definición de la clase *BarTexto*

```
import java.awt.*;
public class BarTexto extends Panel implements java.io.Serializable{
//...
}
```

c. ¿Qué es un csv?

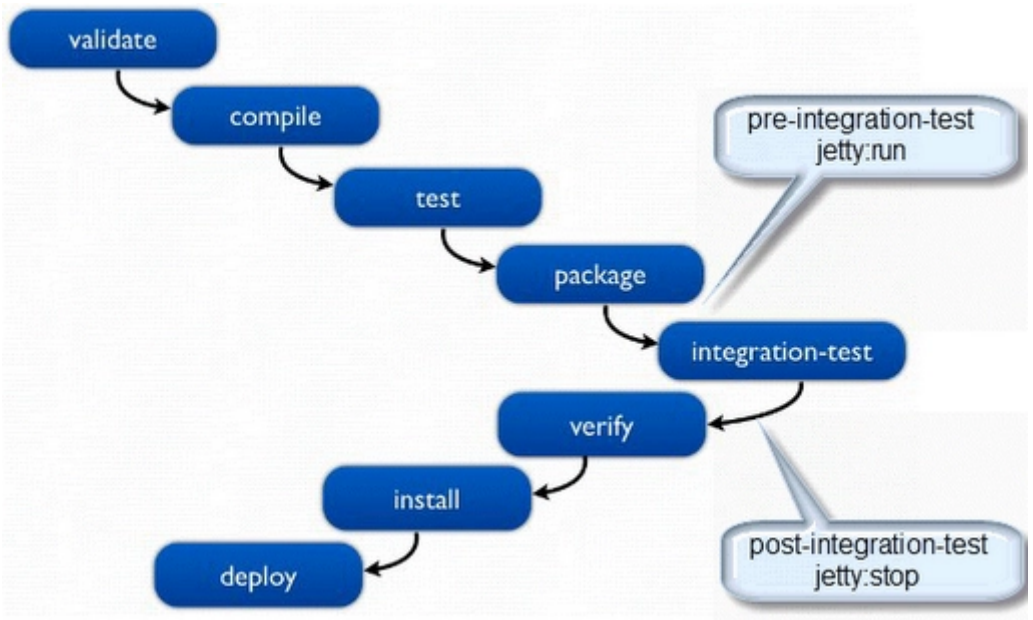
Las siglas CSV vienen del inglés "Comma Separated Values" y significan valores separados por comas. Dicho esto, un archivo CSV es cualquier archivo de texto en el cual los caracteres están separados por comas, haciendo una especie de tabla en filas y columnas. Las columnas quedan definidas por cada punto y coma (;), mientras que cada fila se define mediante una línea adicional en el texto. De esta manera, se pueden crear archivos CSV con gran facilidad (lo explicamos más adelante). Es por esto que los archivos .csv están asociados directamente a la creación de tablas de contenido.
Un archivo CSV suele identificarse con el programa Excel, el cual se basa en cuadrículas que conforman una tabla en filas y columnas. Lo más común es leer archivos CSV desde Excel, ya que el programa (aunque no en las versiones más antiguas) identifica automáticamente los separadores y forma la tabla sin tener que hacer nada por nuestra parte. Hay distintos separadores, usados con mayor o menor frecuencia dependiendo de la región en la que estemos. Como las tablas se usan mayoritariamente para almacenar valores numéricos, hay conflictos entre los formatos decimales de Europa (que se usan comas) con los separadores de coma. Es por ello que en Europa suele usarse el punto y coma, mientras que en EEUU y otros países de habla inglesa suele utilizarse la coma, ya que ello usan el punto como formato decimal.
La facilidad de pasar de una tabla a CSV y viceversa, junto al poco espacio de almacenamiento y exigencias de cómputo que requieren, hace que estos archivos sean prácticamente universales, y al identificar Excel los separadores de manera automática es obvio que todo el mundo relacionado mínimamente con la ofimática y tareas de productividad esté al tanto de los archivos CSV. De todas formas, no siempre funciona el identificador automático de separación de Excel, por lo que si estás en este caso puedes consultar esta guía para abrir correctamente un archivo CSV en Excel.

d.

2. Maven

a. Comandos

- $ mvn compile – compila el proyecto y deja el resultado en target/classes
- $ mvn test – compila los test y los ejecuta
- $ mvn package – empaqueta el proyecto y lo dejará en taget/autentiaNegocio-1.0-SNAPSHOT.jar
- $ mvn install – guarda el proyecto en el repositorio
- $ mvn clean – borra el directorio de salida (target)



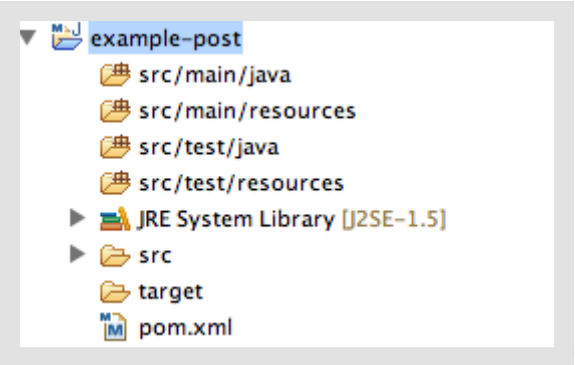
b. Estructura de carpetas de un proyecto Java con Maven

Maven tiene cuatro carpetas fuente por defecto:

- src/main/java : donde guardaremos nuestras clases java fuente. Debajo de esta carpeta situaremos nuestras clases en distintos paquetes.
- src/main/resources : aquí almacenaremos los recursos (ficheros xml, ficheros de propiedades, imagenes, …) que pueda necesitar las clases java de nuestro proyecto. Igualmente aquí tienen que ir los ficheros de configuración de Spring o Hibernate por ejemplo.

- src/test/java : en dicha carpeta se guardan las clases de test que se encargarán de probar el correcto funcionamiento de nuestra aplicación. Aquí por ejemplo podemos guardar nuestros test unitarios de JUnit.
- src/test/resources : en esta carpeta guardamos los recursos que usan los recursos.

Podemos ver una estructura ejemplo en el siguiente proyecto que hemos creado en Eclipse:



El fichero pom.xml está practicamente vacío, ¿cómo sabe entonces Maven que tiene que usar el contenido de las cuatro carpetas para construir el proyecto?. El fichero pom.xml que nos enseña el editor de Eclipse, no es el fichero pom real (llamado pom efectivo) que usa nuestro proyecto. Nuestro fichero pom.xml hereda una serie de propiedades de un fichero pom padre o super pom en él que viene indicado las propiedades estándar que usa Maven. Si queremos podemos ver su contenido desde consola con el siguiente comando.

```
mvn help:effective-pom
```

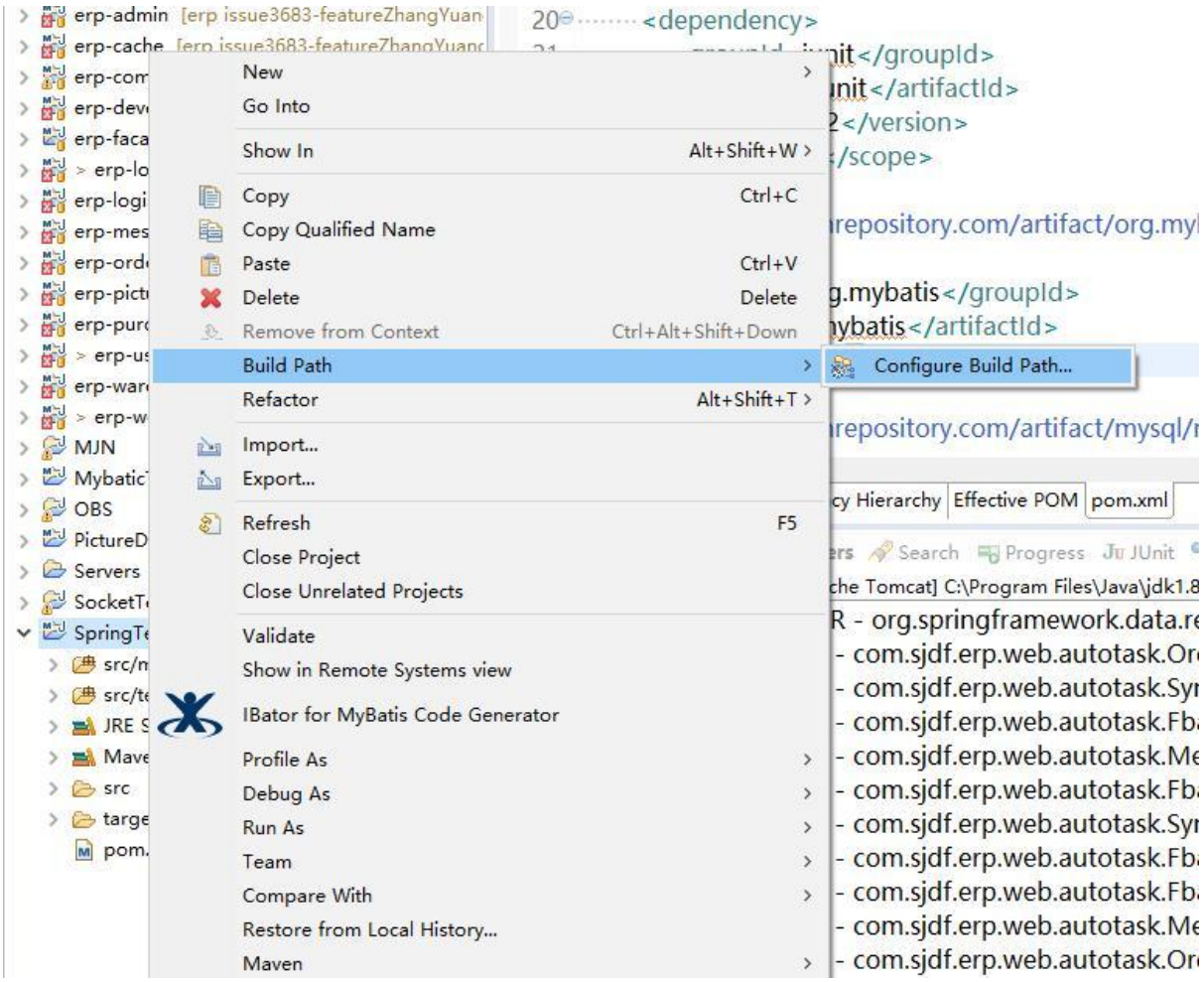
```
<build>
    <sourceDirectory>/Users/hop/example-post/src/main/java</sourceDirectory>

<scriptSourceDirectory>/Users/hop/example-post/src/main/scripts</scriptSourceD
irectory>

<testSourceDirectory>/Users/hop/example-post/src/test/java</testSourceDirector
y>
    <outputDirectory>/Users/hop/example-post/target/classes</outputDirectory>

<testOutputDirectory>/Users/hop/example-post/target/test-classes</testOutputD
irectory>
    <resources>
      <resource>
        <directory>/Users/hop/example-post/src/main/resources</directory>
      </resource>
    </resources>
    <testResources>
      <testResource>
        <directory>/Users/hop/example-post/src/test/resources</directory>
      </testResource>
    </testResources>
    <directory>/Users/hop/example-post/target</directory>
    <finalName>example-post-0.0.1-SNAPSHOT</finalName>
    <pluginManagement>
...
</build>
```

c. Si no tiene la carpeta src/main/resources

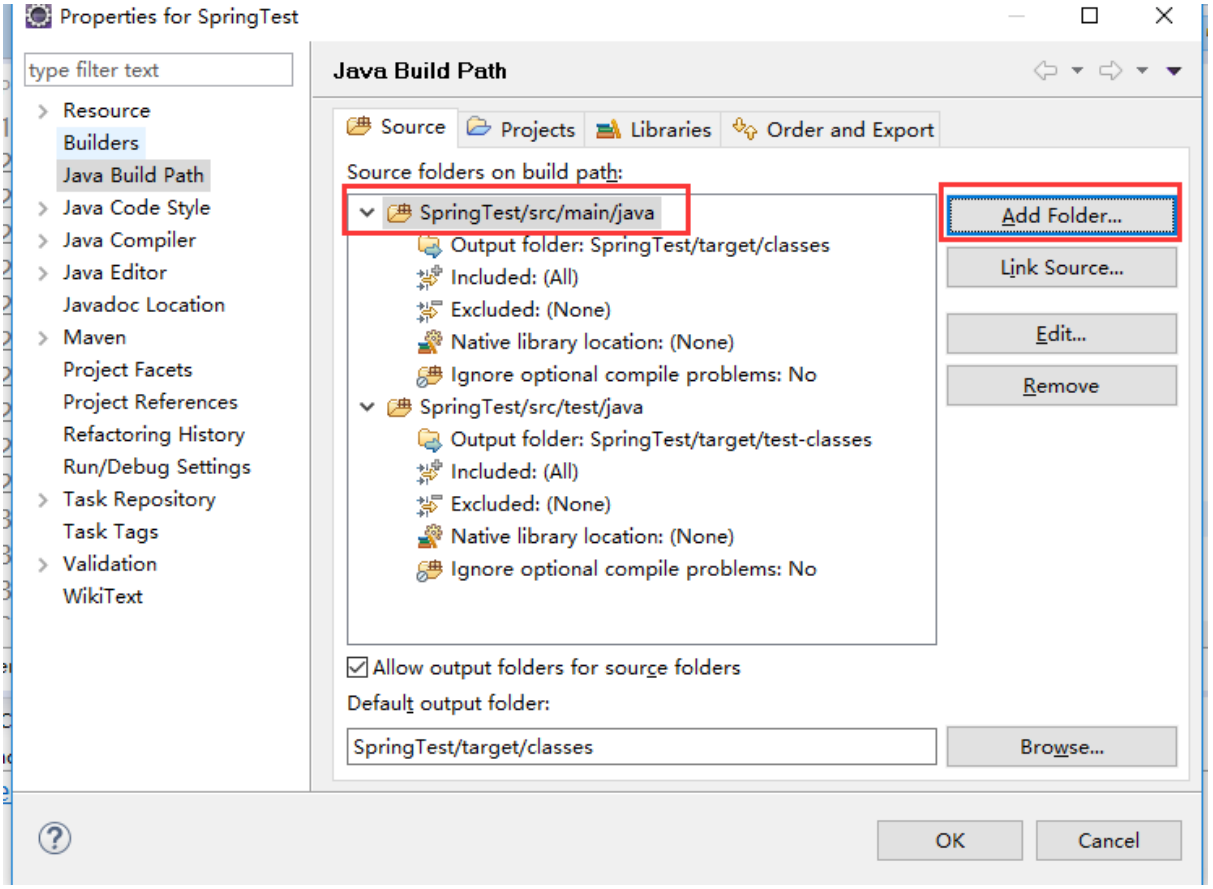
Este es un proyecto maven recién creado. Se puede ver que el proyecto no tiene la carpeta src / main / resources donde se almacenan los archivos de configuración.

solución:

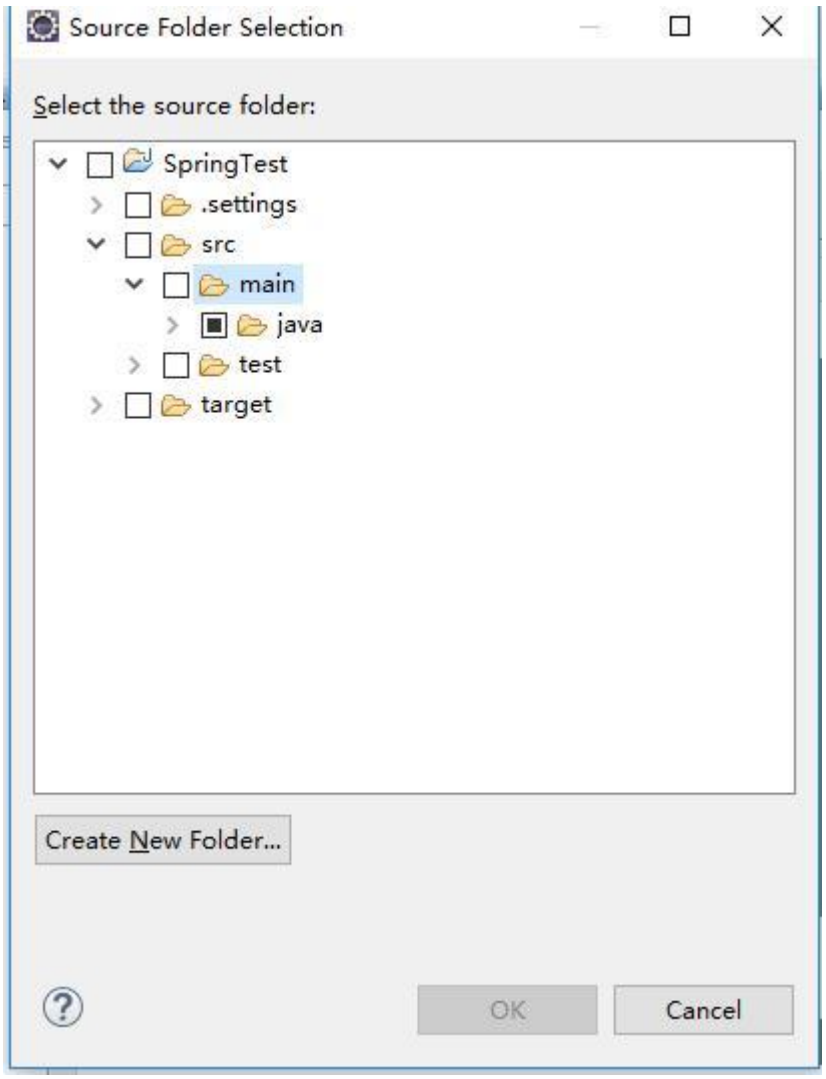
(1) Seleccione el proyecto, haga clic con el botón derecho y seleccione como se muestra en la figura: Ruta de construcción -> Configurar ruta de construcción



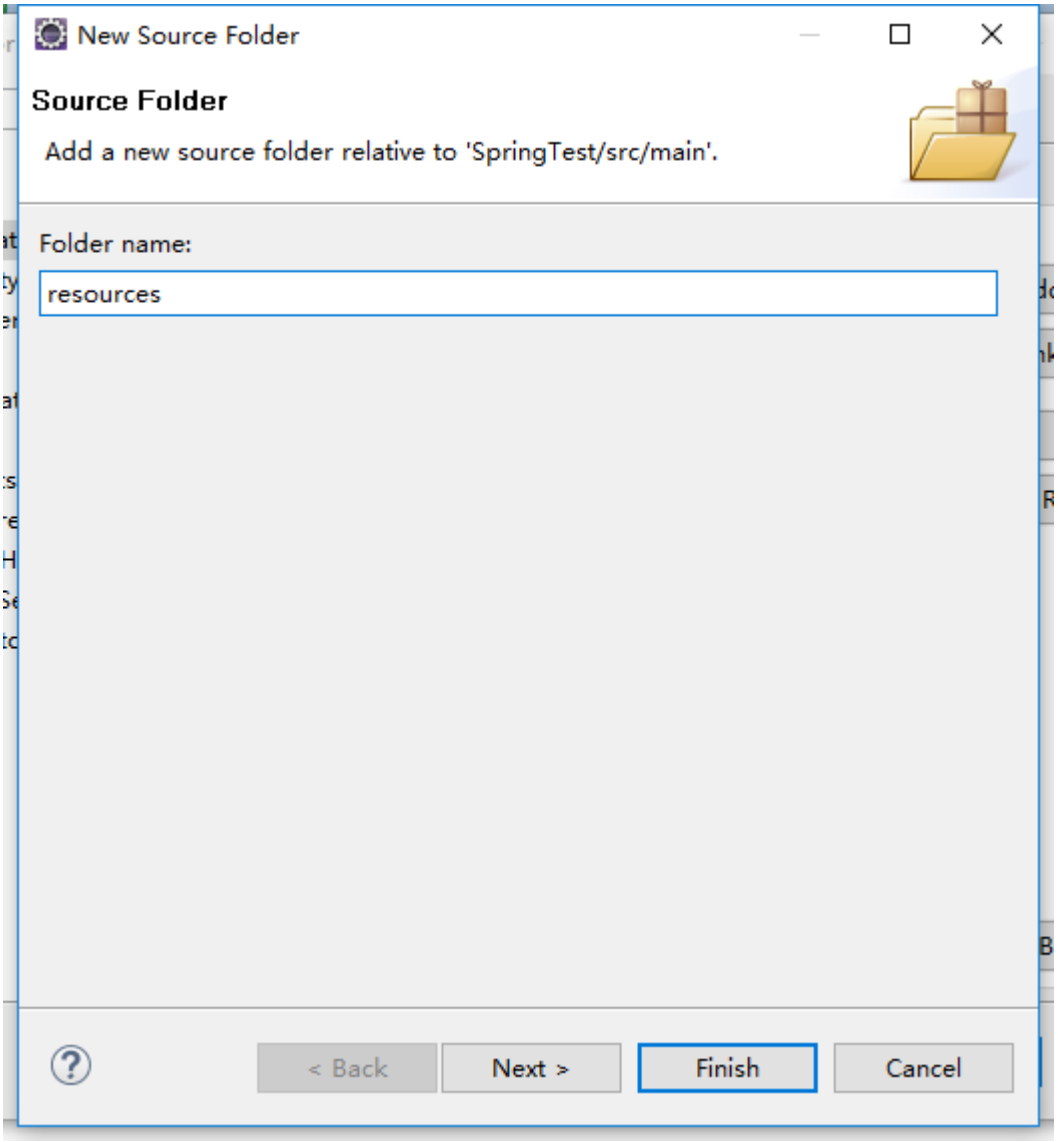
Seleccione el directorio src / main / java, luego haga clic en Agregar Folder



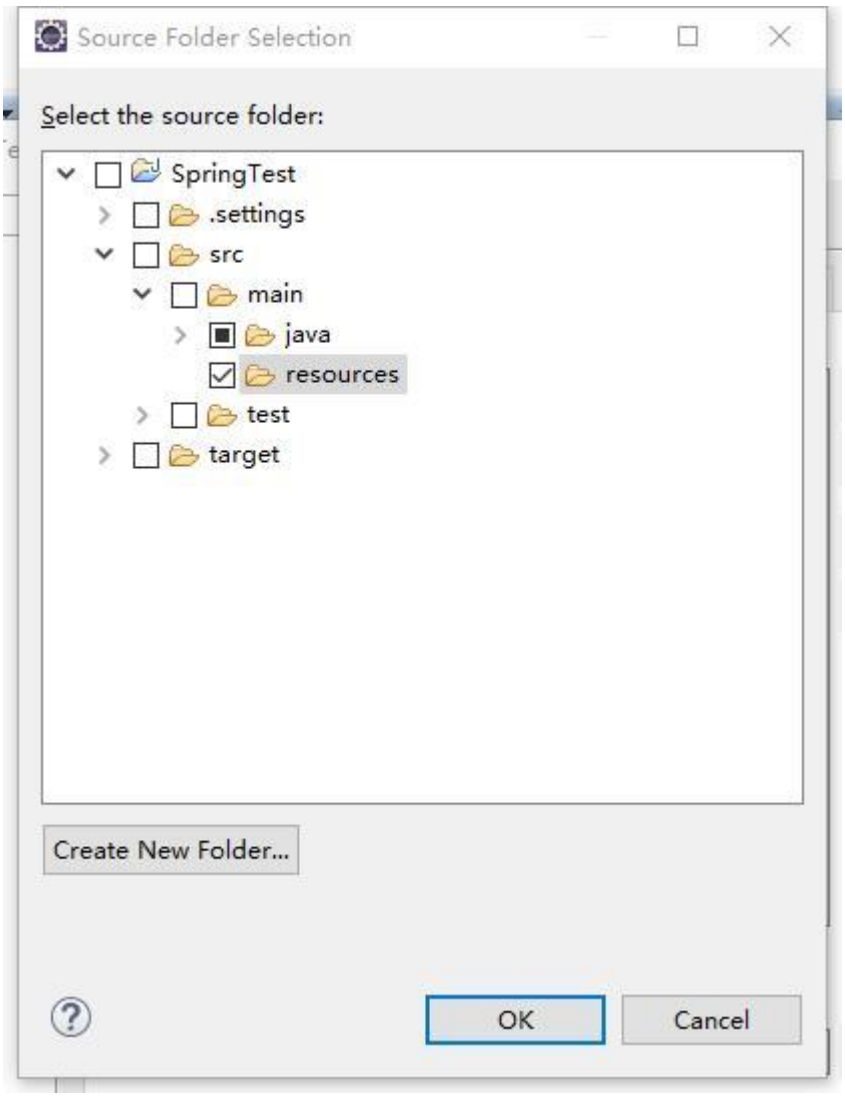
Seleccione principal y haga clic en Crear nuevo Floder a continuación



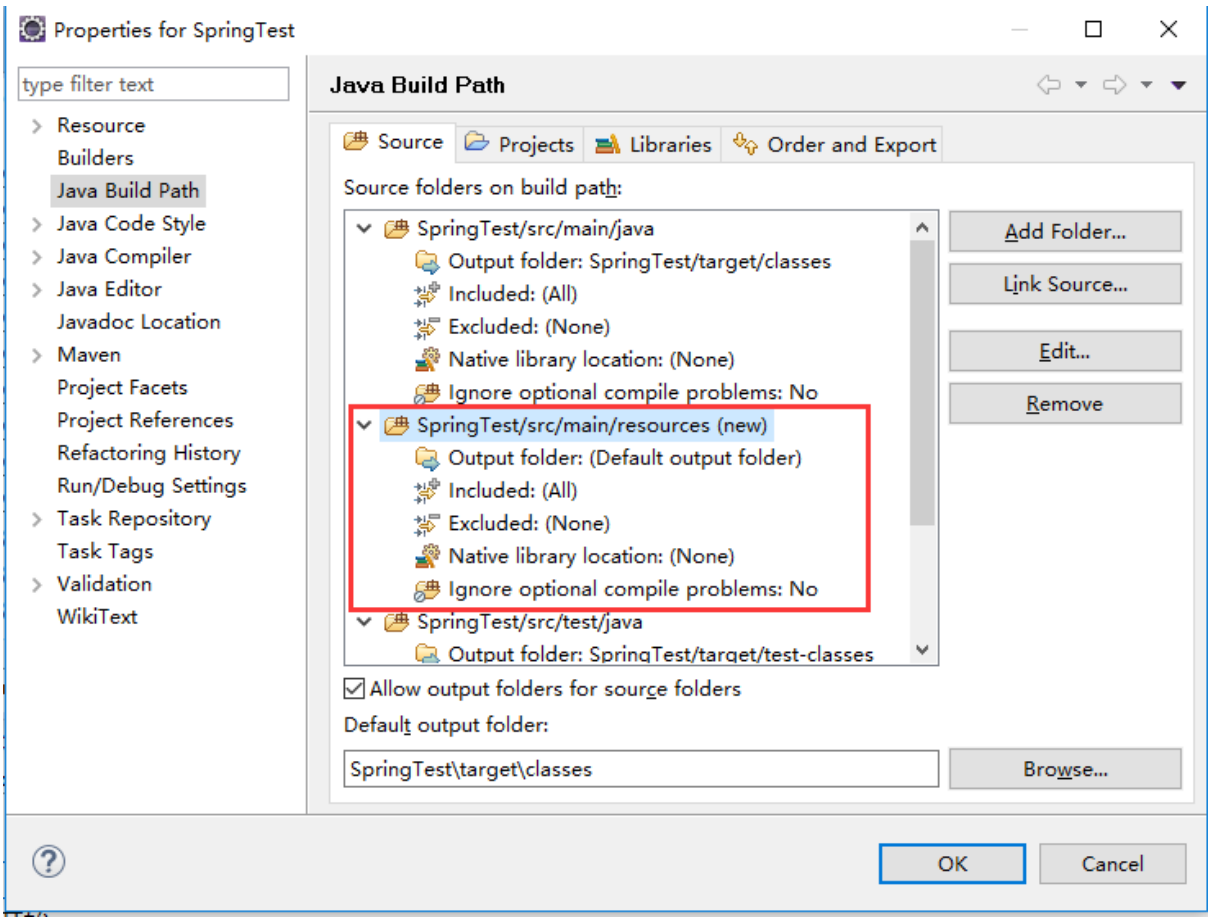
Luego ingrese recursos y haga clic en Finalizar



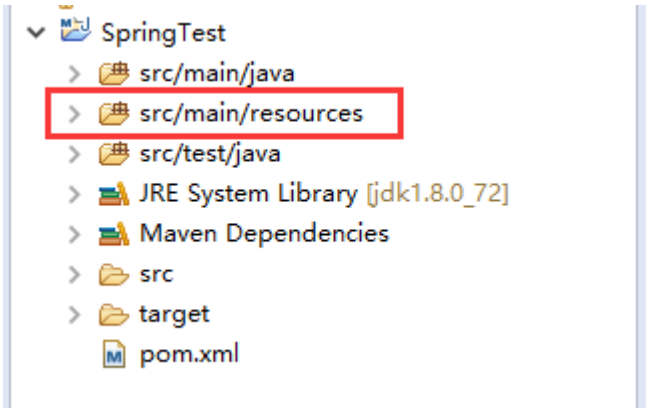
En este momento, hay una carpeta de recursos en la interfaz anterior y luego haga clic en Aceptar.



En este momento, regrese a la interfaz de nivel superior. En este momento, puede ver que ... / src / main / resoures se ha generado. Continúe haciendo clic en Aceptar y la interfaz se cerrará automáticamente.



¡De vuelta al proyecto, esta carpeta ha sido creada! ¡Así, el problema ha sido resuelto!



d. Guía de convenciones de nomenclatura en groupId, artefactId y version

- **groupId** identifica de forma única su proyecto en todos los proyectos. Un ID de grupo debe seguir las reglas de nombres de paquetes de Java . Esto significa que comienza con un nombre de dominio invertido que controlas. Por ejemplo, org.apache.maven,org.apache.commons
  Maven no hace cumplir esta regla. Hay muchos proyectos heredados que no siguen esta convención y, en su lugar, usan ID de grupo de una sola palabra. Sin embargo, será difícil obtener una nueva ID de grupo de una sola palabra aprobada para su inclusión en el repositorio de Maven Central.
  Puede crear tantos subgrupos como desee. Una buena manera de determinar la granularidad del groupIdes usar la estructura del proyecto. Es decir, si el proyecto actual es un proyecto de varios módulos, debe agregar un nuevo identificador al archivo groupId. Por ejemplo, org.apache.maven, org.apache.maven.plugins,org.apache.maven.reporting
- **artefactId** es el nombre del jar sin versión. Si lo creaste, puedes elegir el nombre que quieras con letras minúsculas y sin símbolos extraños. Si se trata de un tarro de terceros, debe tomar el nombre del tarro tal como se distribuye.
  p.ej. maven,commons-math
- version si la distribuye, puede elegir cualquier versión típica con números y puntos (1.0, 1.1, 1.0.1, ...). No use fechas, ya que generalmente se asocian con compilaciones SNAPSHOT (nocturnas). Si se trata de un artefacto de terceros, debe usar su número de versión, sea cual sea, y por extraño que parezca. Por ejemplo,
  2.0, 2.0.1,1.3.1

e. Versionado en Maven

Como vimos un artefacto se identifica por tres atributos: groupId, artifactId y finalmente su versión.
Estas tres cosas las definimos en el pom.xml. Y, en principio no hay restricciones, en particular acá nos interesa la "version".
Ejemplos de versiones que podríamos usar:

- 1.0
- 1
- tiger
- 1.0-alpha
- 1.0-alpha01
- 1.0-alpha01-BUILD20130809160523

Realmente a maven le importa poco lo que escribamos ahí. Así que en principio podría ser cualquier cosa.
Sin embargo existe una convención sobre estas versiones, y diferentes plugins o incluso maven se van a comportar distinto en base a esa convencion de nombres de versiones.
En general se identifican dos tipos de versiones:

- RELEASE
- SNAPSHOT

f. ¿Qué es SNAPSHOT?

Se refiere a una versión del proyecto que está en tiempo de desarrollo. Una especie de WORK-IN-PROGRESS o "Under Development". Parecido al concepto de Build en contraposición al de Release.
En maven la convención es agregar el sufijo "-SNAPSHOT". Todo lo que antecede a este prefijo realmente no le importa a maven, puede tener letras, números, lo que sea (aunque ya veremos que también nos viene bien tener una convención).
Ejemplo:

- 1.0-SNAPSHOT: se lee cómo "la que va a ser la versión 1.0"
- 1.0.1-SNAPSHOT: futura versión 1.0.1
- 1.0-alpha01-SNAPSHOT: lo que va a ser la versión 1.0-alpha01
- etc.

Una consecuencia de esto es que las versiones snapshots son mutables, es decir que van evolucionando y cambiando.
Si bajamos un snapshot hoy, quizás no sea igual al de ayer, o la semana pasada.

3. Interfaces vs Clase abstracta (Herencia)

a. ¿Qué es una clase abstracta?

Una clase que tiene la palabra clave abstracta en su declaración se llama clase abstracta. Las clases abstractas deben tener cero o más métodos abstractos. es decir, métodos sin cuerpo. Puede tener múltiples métodos concretos.
Las clases abstractas le permiten crear planos para clases concretas. Pero la clase heredera debería implementar el método abstracto.
Las clases abstractas no se pueden instanciar.
Razones importantes para usar interfaces

- Las interfaces se utilizan para lograr la abstracción.
- Diseñado para admitir la resolución de métodos dinámicos en tiempo de ejecución
- Le ayuda a lograr un acoplamiento flojo.
- Le permite separar la definición de un método de la jerarquía de herencia

Razones importantes para usar la clase abstracta

- Las clases abstractas ofrecen funcionalidad predeterminada para las subclases.
- Proporciona una plantilla para futuras clases específicas.
- Le ayuda a definir una interfaz común para sus subclases
- La clase abstracta permite la reutilización del código.

b. ¿Qué es la interfaz?

La interfaz es un modelo que se puede utilizar para implementar una clase. La interfaz no contiene ningún método concreto (métodos que tienen código). Todos los métodos de una interfaz son métodos abstractos.
No se puede instanciar una interfaz. Sin embargo, se pueden crear instancias de clases que implementan interfaces. Las interfaces nunca contienen variables de instancia, pero pueden contener variables finales estáticas públicas (es decir, variables de clase constante)

c. Diferencias

Una clase abstracta le permite crear una funcionalidad que las subclases pueden implementar o anular, mientras que una interfaz solo le permite indicar la funcionalidad pero no implementarla. Una clase puede extender solo una clase abstracta mientras que una clase puede implementar múltiples interfaces.

Parámetros	Interfaz	Clase abstracta
Velocidad	Lento	Rápido
Herencias Múltiples	Implementar varias interfaces	Solo una clase abstracta
Estructura	Métodos abstractos	Métodos abstractos y concretos
Herencia/Implementación	Una clase puede implementar múltiples interfaces	La clase puede heredar solo una clase abstracta
Modificadores de acceso	La interfaz no tiene modificadores de acceso. Todo lo definido dentro de la interfaz se asume como modificador público.	La clase abstracta puede tener un modificador de acceso.

Cuándo usar	Es mejor usar la interfaz cuando varias implementaciones comparten solo la firma del método. Jerarquía polimórfica de tipos de valor.	Debe usarse cuando varias implementaciones del mismo tipo comparten un comportamiento común.
Campos de información	la interfaz no puede contener campos de datos.	la clase puede tener campos de datos.
Valor predeterminado de herencia múltiple	Una clase puede implementar numerosas interfaces.	Una clase hereda solo una clase abstracta.
campos definidos	No se pueden definir campos	Una clase abstracta le permite definir tanto campos como constantes
Herencia	Una interfaz puede heredar múltiples interfaces pero no puede heredar una clase.	Una clase abstracta puede heredar una clase y varias interfaces.
Constructor o destructores	Una interfaz no puede declarar constructores o destructores.	Una clase abstracta puede declarar constructores y destructores.
Límite de Extensiones	Puede extender cualquier número de interfaces.	Puede extender solo una clase o una clase abstracta a la vez.

4. Programas y Procesos

a. Conceptos de Recurso y SO

- Recurso: Un recurso, o recursos del sistema, es cualquier componente físico o virtual de disponibilidad limitada en un sistema informático. Cada dispositivo conectado a un sistema informático es un recurso. Cada componente interno del sistema es un recurso.
- Un sistema operativo es el conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software.

b. Programa vs Proceso

Un programa es una secuencia de instrucciones escrita en un lenguaje dado. Un proceso es una instancia de ejecución de un programa, caracterizado por su contador de programa, su palabra de estado, sus registros del procesador, su segmento de texto, pila y datos, etc. (Similar a una clase e instancias de ese tipo de dato)

c. JDK, JRE y JVM

- Java Development Kit

El JDK lo compila y transfiere el código de bytes al JRE.

- JRE es el Java Runtime Environment

Por el contrario, el JRE contiene bibliotecas de clases, que respaldan los archivos, y la JVM.

- La JVM es un dispositivo informático interpretativo responsable de la ejecución de códigos de bytes en un programa de Java compilado. La JVM traduce los códigos de bytes de Java a las instrucciones nativas de la máquina del host. (Para cualquier SO se encarga de traducir a instrucciones nativas de máquina del host). Utiliza estos componentes de software para ejecutar el código intermedio en cualquier dispositivo.

d. Heap, Stack y Garbage collector

- Heap vs Stack: Heap es una estructura dinámica de datos utilizada para almacenar datos en ejecución. A diferencia de la pila de ejecución que solamente almacena las variables declaradas en los bloques previo a su ejecución, el heap permite reservar memoria dinámicamente, es decir, es el encargado de que la «magia» de la memoria dinámica ocurra. Las variables globales y estáticas también son almacenadas en él.
- Garbage Collector (GC) administra de forma automática la memoria, ya que es el encargado de liberar los objetos que ya no están en uso y que no serán usados en el futuro.Cuando creamos aplicaciones suficientemente grandes como para comenzar a perder el control de absolutamente todos los objetos que estamos creando, podemos caer en errores humanos

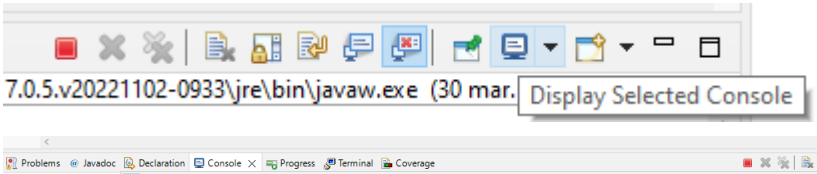
e. Ejecutar varios procesos de un programa en Eclipse

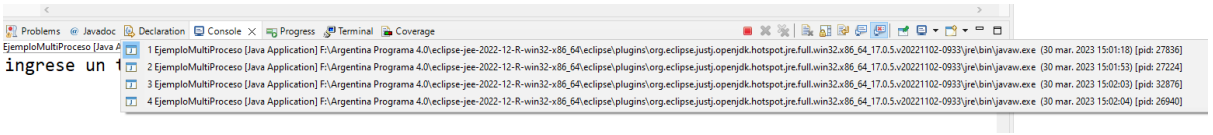
Clase utilizada en el repositorio de ArgentinaPrograma4.Examples

miprimerproyecto.MemoryStackHeap.EjemploMultiProceso

- Se presiona para que se ejecute varias veces con el botón 
- Luego en la vista de “Console” se pueden visualizar todos los procesos en ejecución del mismo programa:

Haciendo click en el siguiente botón:





f. OutOfMemoryError

Este error se produce cuando no hay espacio suficiente para asignar un objeto en el montón de Java. En este caso, el garbage collector no puede dejar espacio disponible para acomodar un nuevo objeto y el almacenamiento dinámico no se puede expandir más.

Hay un ejemplo en el repositorio de ejercicios.

g. StackOverFlow

A stack overflow is a type of buffer overflow error that occurs when a computer program tries to use more memory space in the call stack than has been allocated to that stack.
Un desbordamiento de pila es un tipo de error de desbordamiento de búfer que ocurre cuando un programa de computadora intenta usar más espacio de memoria en la pila de llamadas que el que se ha asignado a esa pila.

Hay un ejemplo en el repositorio de ejercicios.

5. MySQL

a. Qué es MySQL

Los dos programas más utilizados del paquete son MySQL Server y MySQL Workbench, ya que gracias a ellos tendremos la posibilidad de trabajar en las bases de datos. MySQL es un entorno de base de datos creado por Oracle con licencia pública general y también comercial que es capaz de trabajar con estructuras de datos relacionales.
Debemos tener en cuenta que, aun pudiendo formar parte del ecosistema Hadoop, con aplicaciones como Apache o Spark, este paquete no es de código libre y creado por la comunidad como Apache, sino que está patrocinado por la propia empresa Oracle.

- MySQL Server: es el paquete principal y se trata de un software gestor de bases de datos relacionales, potente y muy utilizado. Es capaz de crear bases de datos, son sus respectivas tablas, vistas y relaciones. Además de poder realizar su edición y consultas a dichos datos.
- MySQL Workbench: por su parte, este software nos va a proporcionar potentes opciones de administración de bases de datos relacionales, además de utilizar consultas SQL mediante un entorno visual a golpe de clic.

b. Tutorial de instalación

https://docs.google.com/document/d/1pqwdrWuaXzfu-00X1X-CtzJbi1y1Eg1wPLz0HQqNCvk/edit?usp=sharing

c. Conceptos

- Clave Primaria (PK)

Las claves primarias (Primary Key) son las que identifican de manera única cada fila o registro de una tabla, es decir, que no se puede repetir en una tabla el valor de un campo o columna que se le es asignado como PK

- Clave Foránea (FK)

Una clave foránea (Foreign Key) es un campo de una tabla “X” que sirve para relacionar con otra tabla “Y” en la cual el campo de esta tabla es una clave primaria (PK). Esta clave nos permite mantener la integridad referencial en la base de datos.

- DDL - Definición de datos

- CREATE

```
-- creacion de tablas con el comando:
-- CREATE TABLE nombre de tabla(
-- nombre de la columna   TIPO DE DATO,

-- Las KEYS son las columnas que pueden ser AUTO_INCREMENT.
CREATE TABLE idioma(
    id INT(11) NOT NULL AUTO_INCREMENT,
    codigo VARCHAR(45),
    nombre VARCHAR(45) NOT NULL,      -- NOT NULL que no sea vacio
    PRIMARY KEY (id)
);
```

- ALTER

```
-- ALTER
-- Agregamos la columna
ALTER TABLE estudiante ADD COLUMN idioma_id INT(11);
-- Asignamos la FK a la columna creada
ALTER TABLE estudiante ADD FOREIGN KEY (idioma_id) REFERENCES idioma(id);
```

- DROP: para eliminar una tabla por completo

- DML Manipulación de datos
  - INSERT: Carga de información

```
-- INSERT carga de información
INSERT INTO estudiante(nombre, apellido, edad)
VALUES ("Juan","Perez", 22);
INSERT INTO estudiante(nombre, apellido, edad)
VALUES ("Nahuel","Ramirez", 12);

-- AGREGO IDIOMAS
INSERT INTO idioma(codigo,nombre)
VALUES ("ENG","Ingles"),("ESP","Español"),("FRA","Frances");

-- AGREGO CURSOS
INSERT INTO curso(codigo,nombre)
VALUES ("JAVA","Programación en JAVA"),("PYTHON","Programación en Python");
```

  - UPDATE: Modificaciones

```
-- UPDATE modificación

UPDATE estudiante
SET nombre = "Pedro",
apellido = "Escobar"
WHERE id = 1;
```

  - DELETE: Borrados selectivos

```
-- DELETE borrados selectivos

DELETE FROM estudiante
WHERE id = 2;
```

- SELECT - Consultas/query

```
-- VISUALIZACIÓN DE DATOS

SELECT * FROM estudiante;
SELECT * FROM idioma;

SELECT nombre,apellido FROM estudiante WHERE edad > 20;
```

d. Relación

Para saber qué son las relaciones en las bases de datos es necesario hablar del tipo de base de datos relacionales. En estas bases de datos, la información se almacena en diferentes tablas, distribuida en filas y columnas.
La relación de una base de datos es el vínculo que se establece entre distintos elementos de las tablas que la conforman. En este tipo de relaciones es fundamental el uso de los campos de llave primaria (primary key) que son los que se relacionan con otros registros de otras tablas.
Es importante destacar que, a la hora de definir las relaciones entre los campos de distintas tablas en una base de datos, los nombres de los mismos no tienen por qué ser iguales. Sin embargo, sí es necesario a la hora de establecer estas relaciones, que el tipo de datos de los campos enlazados sea el mismo.

Las relaciones en las bases de datos son claves para establecer concordancias en las asignaciones y garantizar la integridad referencial de la información (que los datos no se modifiquen o varíen durante el proceso).
Gracias a las relaciones se mantiene una lógica y consistencia entre todos los datos que almacena. Además, las relaciones evitan que se dupliquen los registros dentro de una base de datos.

Relación uno a uno (1:1)

Se produce cuando la relación se realiza solo entre un registro de una tabla con un registro de otra. Es una de las relaciones más utilizadas, ya que permiten una relación de tipo exclusivo.
En esta relación, los campos establecidos como primary key de ambas tablas están enlazados.
Algunos ejemplos de este tipo de relaciones en bases de datos 1:1 lo encontramos en la asignación de banderas por país, donde cada bandera solo se puede asignar a un único país. Lo mismo ocurre con otros datos como las matrículas de coche (cada coche solo puede tener un único número de matrícula, y cada matrícula solo puede corresponder a un coche en concreto) o el número de serie de un producto (cada producto tiene un número único de serie, y cada número de serie corresponde a un único producto).

Relación uno a varios (1:N)

En este tipo de relación de uno a varios, un registro de una tabla puede enlazarse a varios registros de otra tabla. La primera key está vinculada a varios registros de otra tabla. Esta relación es la más utilizada en las bases de datos relacionales.
Veamos ejemplos de este tipo de relación 1:N:

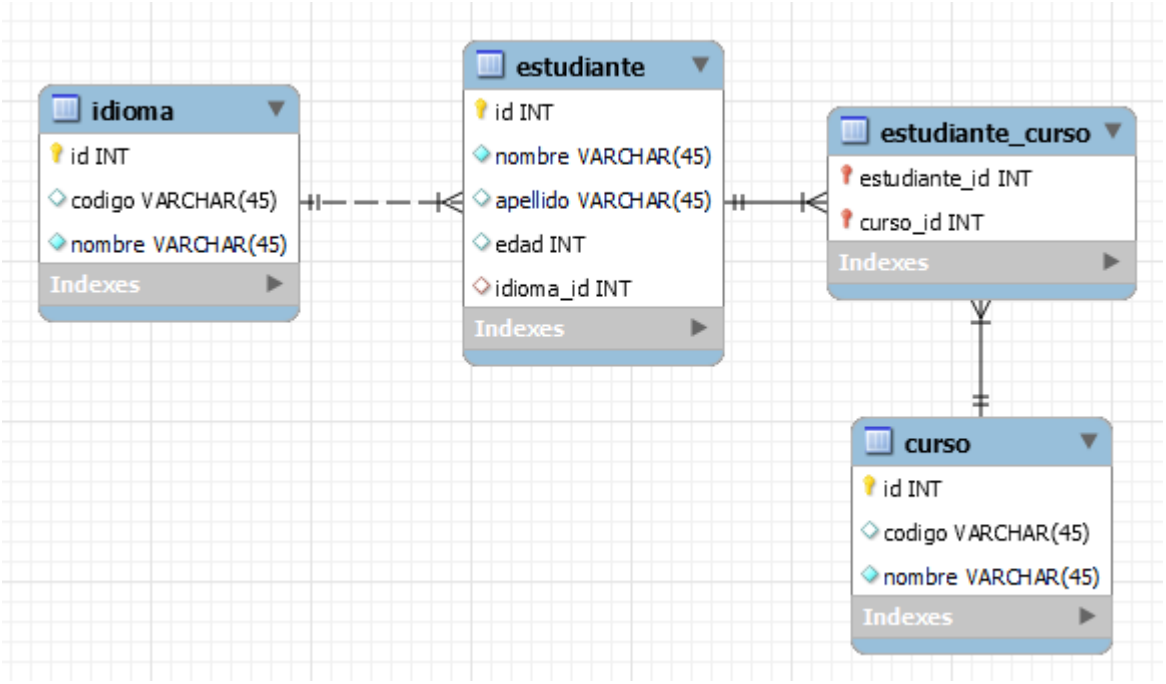
- Dirección postal que puede relacionarse con varias personas que pueden vivir en el mismo.
- Nombre de una empresa que puede relacionarse con diversos trabajadores de la misma.

Nombre de cliente que puede tener distintos pedidos de venta.

Relaciones varios a varios (N:N)

Cuando varios registros de una tabla pueden relacionarse con varios registros de otra tabla se está ante una relación del tipo varios a varios. Este tipo de relación es la menos habitual en una base de datos relacional.
Un ejemplo de este tipo de tablas N:N lo podemos ver con una tabla de equipo de fútbol y otra de jugadores. Se puede dar el caso de que en un equipo de fútbol juegan varios jugadores y, que además, un jugador haya jugado en varios equipos de fútbol.
Otro ejemplo lo podemos encontrar en una base de datos con una tabla de asignaturas y otra de profesores. Varios profesores pueden dar una misma asignatura, así como un profesor puede impartir varias asignaturas.

e. Ejemplo



- ***Creación de base de datos y seleccionarla***

```
CREATE DATABASE PRUEBA;
-- Seleccionamos la base con la cual queremos arrancar
USE PRUEBA;
```

- ***Creación de tablas***

```
-- DDL - Definición de datos
-- CREATE, ALTER y DROP
```

```
-- creacion de tablas con el comando:
-- CREATE TABLE nombre de tabla(
-- nombre de la columna   TIPO DE DATO,

-- Las KEYS son las columnas que pueden ser AUTO_INCREMENT.
CREATE TABLE idioma(
        id INT(11) NOT NULL AUTO_INCREMENT,
        codigo VARCHAR(45),
        nombre VARCHAR(45) NOT NULL,     -- NOT NULL que no sea vacio
        PRIMARY KEY (id)
);

CREATE TABLE estudiante(
        id INT(11) NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(45) NOT NULL,
        apellido VARCHAR(45),
        edad INT(3),
        PRIMARY KEY (id)
);

CREATE TABLE curso(
        id INT(11) NOT NULL AUTO_INCREMENT,
        codigo VARCHAR(45),
        nombre VARCHAR(45) NOT NULL,     -- NOT NULL que no sea vacio
        PRIMARY KEY (id)
);

CREATE TABLE estudiante_curso (
         estudiante_id INT(11) NOT NULL,
        curso_id INT(11) NOT NULL,
        PRIMARY KEY (estudiante_id,curso_id),
        FOREIGN KEY (curso_id) REFERENCES curso(id),
        FOREIGN KEY (estudiante_id) REFERENCES estudiante(id)
);
```

- ***Modificación de una tabla***

```
-- ALTER
-- Agregamos la columna
ALTER TABLE estudiante ADD COLUMN idioma_id INT(11);
-- Asignamos la FK a la columna creada
ALTER TABLE estudiante ADD FOREIGN KEY (idioma_id) REFERENCES idioma(id);
```

- ***Manipulación de datos***

```
-- DML - Manipulación de datos
```

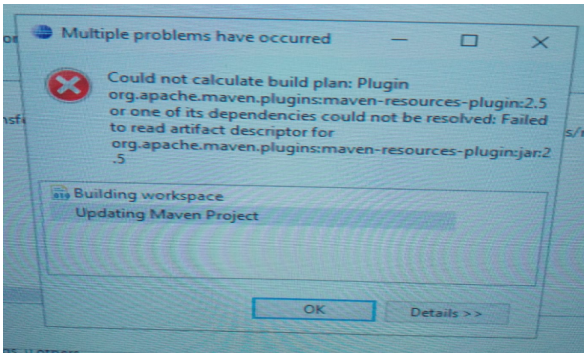
```
-- INSERT carga de información
INSERT INTO estudiante(nombre, apellido, edad)
        VALUES ("Juan","Perez", 22);
INSERT INTO estudiante(nombre, apellido, edad)
        VALUES ("Nahuel","Ramirez", 12);

-- AGREGO IDIOMAS
INSERT INTO idioma(codigo,nombre)
        VALUES ("ENG","Ingles"),("ESP","Español"),("FRA","Frances");

-- AGREGO CURSOS
INSERT INTO curso(codigo,nombre)
        VALUES ("JAVA","Programación en JAVA"),("PYTHON","Programación en Python");

-- UPDATE modificación

UPDATE estudiante
        SET nombre = "Pedro",
        apellido = "Escobar"
WHERE id = 1;

-- DELETE borrados selectivos

DELETE FROM estudiante
WHERE id = 2;
```

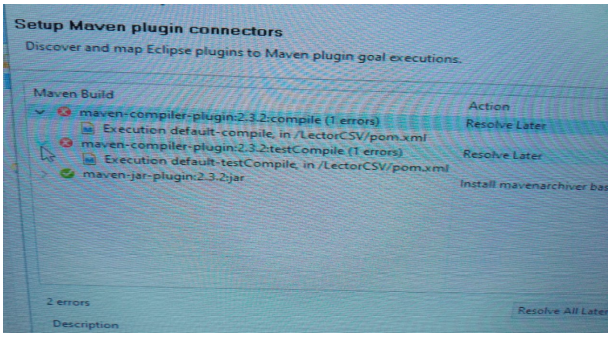
- ***Consulta***

```
-- VISUALIZACIÓN DE DATOS

SELECT * FROM estudiante;
SELECT * FROM idioma;

SELECT nombre,apellido
FROM estudiante
WHERE edad > 20;
```

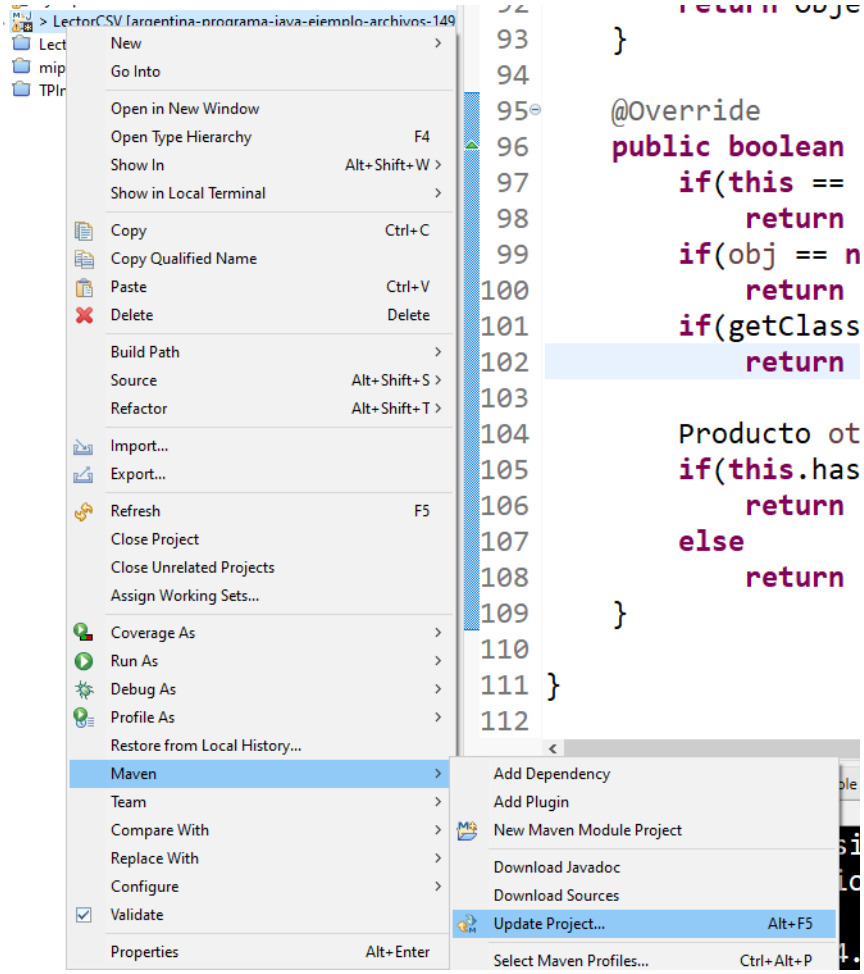
6. Errores registrados

a. Could not calculate build plan / Source option 5 is no longer supported

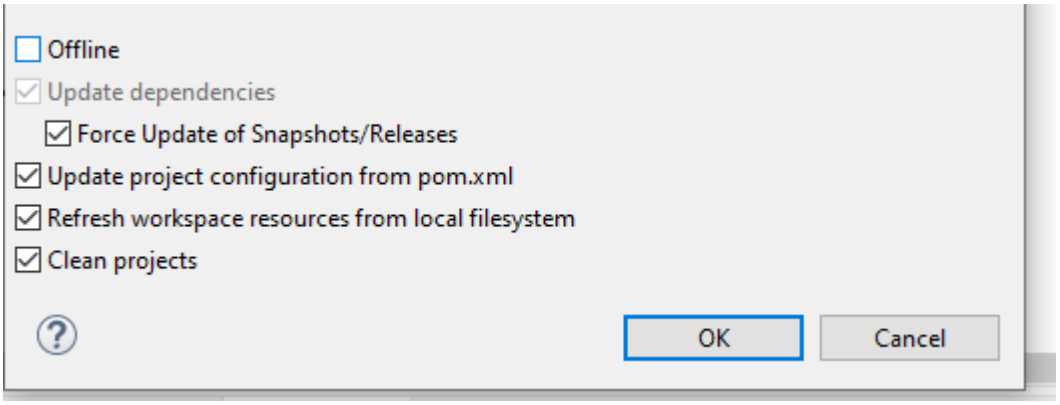




Solución I:
Probar con Update project:



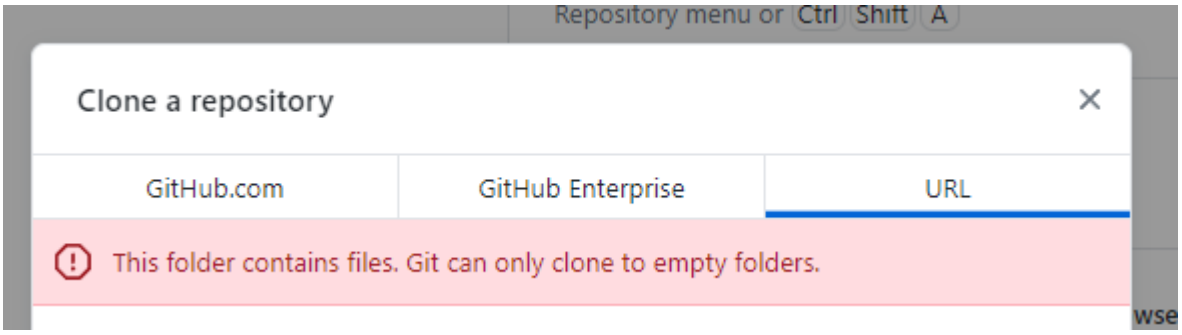
Seleccionar el proyecto que genera conflictos y además tildar la opción “Force Update of Snapshots/Releases”



b. com.opencsv.exceptions.CsvDataTypeMismatchException

Debemos tratar de revisar que no tengamos un conflicto en los campos del csv con el tipo de campo donde queremos almacenar el valor.
Para eso, dentro del bloque/informe del error debemos buscar la frase “Caused by:...”
Como por ejemplo:
Caused by: org.apache.commons.beanutils.ConversionException: Error converting from 'String' to 'Integer' For input string: "gana"
Podemos identificar en esa sentencia que hay un error al tratar de guardar el valor “gana” (un string) como un campo Integer.

c. GitHub: This folder contains files. Git can only clone to empty folders



Tenemos que seleccionar otro directorio donde clonar el repositorio remoto. Esto se debe a que solo podemos tener un repositorio en cada directorio.
Como así también, puede ser que dentro del directorio donde queremos descargar el repositorio, ya esté descargado el repositorio en cuestión.

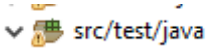
d. cvc-elt 1.a: Cannot find the declaration of element project

Esto puede surgir de acuerdo a la versión de la IDE y configuraciones del entorno. Una solución factible es cambiar http por https en la primera línea del pom.xml

<project xmlns="https://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

e. Maven no corre los test creados

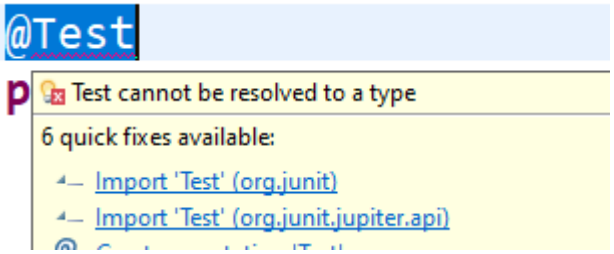
Como primera aclaración, las clases de Test deben estar dentro de la carpeta:



Para este curso usamos las siguientes dependencias para realizar los test unitarios, para eso tenemos que verificar dentro del POM que estén las adecuada:

```
  <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.2.0</version>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.junit.platform</groupId>
        <artifactId>junit-platform-runner</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
</dependency>
```

Luego, debemos prestar atención a las librerías que importamos. Por más que eclipse no nos arroje más errores, no quiere decir que hayamos importado la librería correcta. Ejemplo:



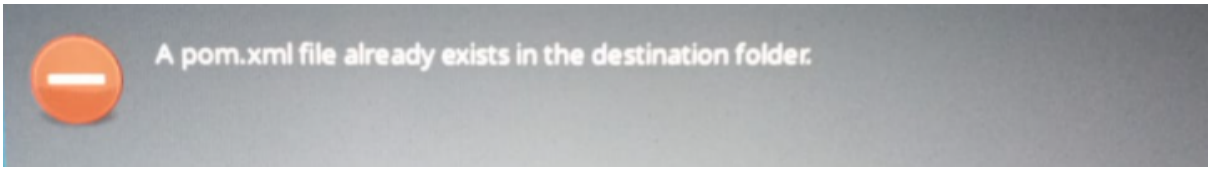
Eclipse nos ofrece en este caso dos alternativas, la correcta es la primera. Y lo mismo sucede con los “Assert” que implementemos. Podríamos por ejemplo poner:

import static org.junit.Assert.*

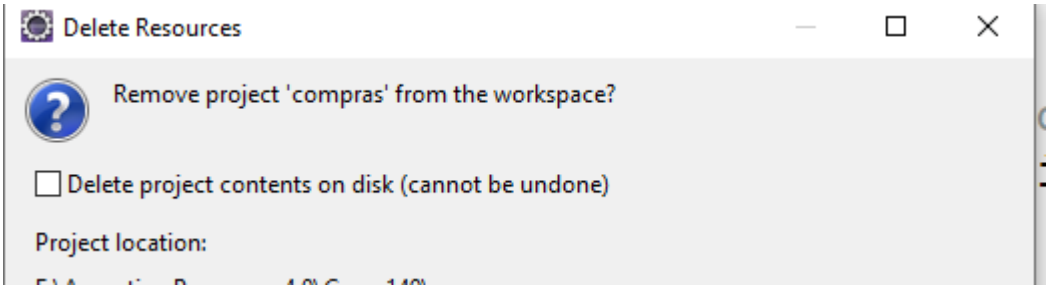
O bueno, si no importar de acuerdo al uso, pero tienen que estar dentro del paquete “org.junit.Assert.”. Por ejemplo:

```
import static org.junit.Assert.assertEquals;
```

f. A pom.xml file already exists in the destination folder



El error se debe a que queremos crear un proyecto Maven “New Maven Project” dentro de un directorio que ya tiene un proyecto. Debemos seleccionar otro directorio para poder crearlo.
Debemos prestar atención, que a veces borramos un proyecto de eclipse y no necesariamente del disco. Para eso, es recomendable revisar dentro del directorio donde queremos crear el proyecto si, si ya no existe.
Podemos prestar atención que cuando borramos un proyecto de Eclipse:



Nos ofrece la opción “Delete project contents on disk”, si no la seleccionamos, el proyecto continuará almacenado dentro del disco. Tener mucho cuidado con la utilización de esta opción, no hay vuelta atrás si no tenemos alojado el código en un repositorio remoto.

g. Maven Error: Could not find or load main class

Se debe a que dentro de nuestro pom.xml no declaramos la clase por la cual el programa debe ejecutarse, en la cuál se encuentra el método main.
La solución que voy a mencionar, está relacionada a cómo lo solucionamos de manera rápida en los proyectos que hicimos durante la cursada.
Lo que hacemos a modo práctico, es agregar la sección <build> dentro del pom.xml con las siguientes características, y dentro de la sección “<mainClass></mainClass>” es donde debemos ingresar nuestra clase por la cual queremos que se inicie nuestro programa.

```
<build>
        <plugins>
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-shade-plugin</artifactId>
                        <version>3.2.4</version>
                        <executions>
                                <execution>
                                        <phase>package</phase>
                                        <goals>
                                                <goal>shade</goal>
```

```
                                        </goals>
                                        <configuration>
                                        <transformers>
                                                <transformer
                                                implementation="org.apache.maven.plugins.shade.r
                                                esource.ManifestResourceTransformer">
                                                <mainClass>org.argentinaprograma.compras.App</m
                                                ainClass>
                                                </transformer>
                                        </transformers>
                                        </configuration>
                                </execution>
                        </executions>
                </plugin>
        </plugins>
</build>
```

h. No suitable constructor found for FileReader(java.lang.String, java.nio.charset.Charset)

El error se debe a que el método FileReader para utilizar con el segundo parámetro donde podamos especificar el Encoding del archivo está disponible a partir de la versión Java 13.

Por eso lo que hay que hacer es agregar en el pom lo siguiente:

- En la parte de properties

```
<properties>
        <java.version>13</java.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

- En la parte de build:

```
<build>
        <plugins>
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-shade-plugin</artifactId>
                        <version>3.2.4</version>
                        <executions>
                                <execution>
                                <phase>package</phase>
                                <goals>
                                        <goal>shade</goal>
                                </goals>
                                <configuration>
                                        <release>13</release>
                                <transformers>
                                        <transformer
                                        implementation="org.apache.maven.plugins.shade.resourc
                                        e.ManifestResourceTransformer">
                                                <mainClass>org.argentinaprograma.validadorDe
                                        Correlativas.App</mainClass>
                                        </transformer>
                                </transformers>
```

```
                                    </configuration>
                                    </execution>
                                </executions>
                            </plugin>
                        </plugins>
</build>
```

i.

5. Videos

a. GitHub: Creación de un repositorio sobre un proyecto existente

https://utn.zoom.us/rec/play/duZWGyNAE0TQnQ5nPZcdVp140t4hPtdOQNJwabyX3Lp8HwcstHPj5D9rcxC6M_jUU9u75VD0qtNKI2BJ.bskKlPF_p_ODVwmZ?canPlayFromShare=true&from=my_recording&startTime=1679571311000&componentName=rec-play&originRequestUrl=https%3A%2F%2Futn.zoom.us%2Frec%2Fshare%2F4bcN8OeI7TIXM-8jCJI8-2duFHGSL9Z8-UGzd9qmWco-o0nZ573zQPGVdQtvxogR.6NC-tA-bAyi6yUbW%3FstartTime%3D1679571311000

b. Maven: Inicializar proyecto Maven

https://utn.zoom.us/rec/play/vQ953JLkzadPHU178ip_IO7PbV2lvoON3zgnczDA80um6IL5CN3RmY4VyN3dxbF3afVwo7-1UE0nTOjC.SYAZb5kdPxjR8O9n?canPlayFromShare=true&from=my_recording&startTime=1679573359000&componentName=rec-play&originRequestUrl=https%3A%2F%2Futn.zoom.us%2Frec%2Fshare%2F4bcN8OeI7TIXM-8jCJI8-2duFHGSL9Z8-UGzd9qmWco-o0nZ573zQPGVdQtvxogR.6NC-tA-bAyi6yUbW%3FstartTime%3D1679573359000

c. Importación de un proyecto Maven y su ejecución

https://utn.zoom.us/rec/play/o08N-5daKwQEg01VZolQ75DlRZMjFX9gseRoe9_BTOtfDQbR4DehxH7da5TyJCItUCMmx_wKrCGIxId7.vW7uYCeCTKfIozAy?autoplay=true&startTime=1678983904000

d. Polimorfismo: Video ejemplo

https://utn.zoom.us/rec/play/ES-oK5hm2pMieIlkEoE3dv2lQsuzQhsT3ZbnpVwf8wUrJJmvWnzgjOTq2zJKsdgr7Zvid3JbmwRsKitr.Ym0vR-7pjdHDy_CZ?canPlayFromShare=true&from=my_recording&startTime=1679574841000&componentName=rec-play&originRequestUrl=https%3A%2F%2Futn.zoom.us%2Frec%2Fshare%2F4bcN8OeI7TIXM-8jCJI8-2duFHGSL9Z8-U

Gzd9qmWco-o0nZ573zQPGVdQtvxogR.6NC-tA-bAyi6yUbW%3FstartTime%3D1679574841000

e. Colecciones: Comparación entre objetos

https://utn.zoom.us/rec/share/8GcSFHqsREtkwqCOKw5a963jU0jrSF10YReyrmucW4aeT4mGXPsWy24A6IBINkVs.JSiUJRZMVCKz5Co4?startTime=1679584553000

f. Mysql

i. Introducción a base de datos MYSQL

https://utn.zoom.us/rec/share/iUItToXbw-_Nz6FEgxSgkBKYnguTMwnNNDoP6INrOP57udDPLkiP4ZKHltTw7vvX.X10zZX6EE4pvdpTE?startTime=1680791003000

ii. Creación de base de datos utilizando comandos

https://utn.zoom.us/rec/share/iUItToXbw-_Nz6FEgxSgkBKYnguTMwnNNDoP6INrOP57udDPLkiP4ZKHltTw7vvX.X10zZX6EE4pvdpTE?startTime=1680793559000

iii. Creación de base de datos utilizando Interfaz gráfica

https://utn.zoom.us/rec/share/iUItToXbw-_Nz6FEgxSgkBKYnguTMwnNNDoP6INrOP57udDPLkiP4ZKHltTw7vvX.X10zZX6EE4pvdpTE?startTime=1680795422000

6. Repositorios

a. Repositorio de ejercicios

https://github.com/tachi123/ArgentinaPrograma4.0-Examples.git

b. Proyecto Compras

https://github.com/tachi123/ArgentinaPrograma4.0-Compras.git

c. Ejemplo Polimorfismo

https://github.com/tachi123/ArgentinaPrograma4.0-EjemploPolimorfismo.git

d. Ejemplo de Testing

El repositorio contiene código con el propósito de mostrar el funcionamiento básico de JUnit, framework que utilizamos para realizar Test Unitarios en Java Link: https://github.com/disilab-frba-utn-edu-ar/argentina-programa-java-ejemplo-testing

e. Ejemplo de Lectura e Interpretación de Archivos

El repositorio contiene código que pretender mostrar cómo se puede leer un archivo formato "csv" mediante una biblioteca de mercado (OpenCsv). Además, está pensado para que la ruta del archivo se envíe por argumento al programa Java Link: https://github.com/disilab-frba-utn-edu-ar/argentina-programa-java-ejemplo-archivos/tree/main/LectorCSV

f. Validador de Correlativas

https://github.com/tachi123/validadorDeCorrelativas.git