

# Enunciado Tarea 1

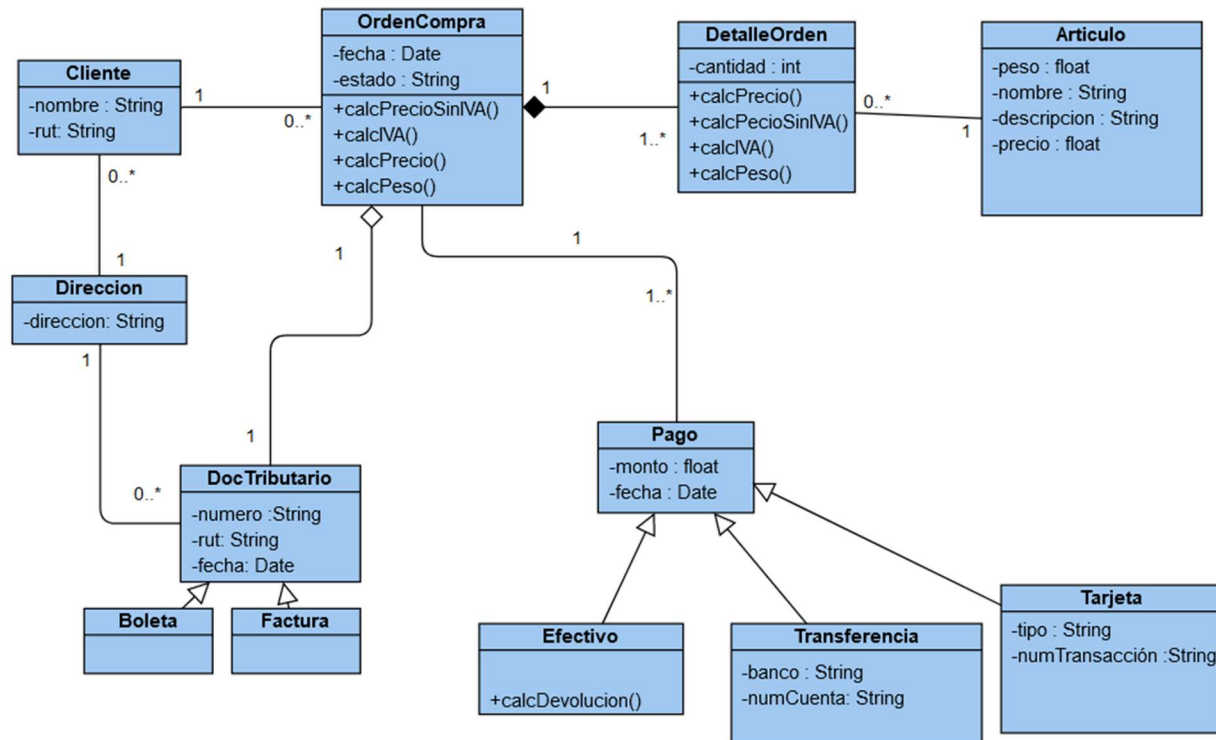
El objetivo de esta tarea es escribir el código del modelo UML visto en el último curso utilizando GIT. Esta tarea debe realizarse en grupos de **dos personas** (al ser la primera tarea en un grupo, excepcionalmente se puede hacer solo pero habrá que justificarlo durante la rendición).

A modo de recordatorio, aquí está el texto y el modelo UML del ejercicio realizado durante el curso.

En primer lugar, para cada pedido se crea una **orden de compra** en la **fecha** actual y con un **estado** que variará con el tiempo. Cada pedido se asocia a un **cliente** al que se le pide su **nombre**, su **RUT** y su **dirección**.

Una orden de compra se compone de varias líneas que detallan **una cantidad** de artículos que la persona quiere comprar. Tanto para el orden como para el **detalle**, queremos poder **saber el precio total**, los **precios sin y con IVA** y también el **peso total** para poder organizar la entrega (que gestionamos con otro sistema). Cada **artículo** tiene un **nombre**, una **descripción**, un **peso** y un **precio**.

Posteriormente, un orden de compra se asocia a un **documento tributario**, ya sea **boleta** o **factura**, con un **RUT**, una **fecha** y una **dirección**. También se registran los **pagos** asociados a cada orden. Un cliente puede pagar en varios plazos, y a veces en diferentes **fechas**. El pago puede hacerse por **transferencia** con **el nombre del banco** y el **número de cuenta**. O por **tarjeta**, con el **tipo de tarjeta** y **el número de transacción** que nos da el sistema bancario. El cliente también puede pagar en **efectivo**, en cuyo caso nos gustaría saber **cuánto tenemos que devolver** si no paga la cantidad exacta.



Deben escribir la implementación completa de cada una de las clases y las relaciones entre ellas. Recuerde que una OrdenCompra utilizará los métodos de los diferentes DetalleOrden que a su vez utilizarán los de Artículo.

Además de los métodos presentados aquí, debes implementar **los métodos getters y setters y el método toString de cada clase**. Este método toString debe mostrar una descripción relevante para la clase en cuestión (por ejemplo, "(nombre) RUT: (rut)" para la clase Cliente), tenga en cuenta que, para las clases compuestas, es posible hacer un loop y utilizar el toString de otras clases.

Pueden elegir los argumentos y los tipos de retorno de métodos, pero deben ser pertinentes y coherentes.

Pueden utilizar todas las clases estándar de Java para su implementación (String, ArrayList, [Date](#)...). No olviden utilizar la documentación de Java.

También deben escribir **un main que le permita probar todas sus clases para al menos 3 órdenes de venta, 2 compradores, 5 productos diferentes y 4 pagos**. Las instancias deben hacerse cuando sean relevantes, también puedes usar métodos para llamar desde el main para

organizar mejor el código. No olviden usar toString al mostrar la información en la salida estándar.

Puedes seguir el siguiente breve tutorial para iniciar tu proyecto con GIT. Es importante que ambos miembros del equipo contribuyan a la tarea en proporciones similares. Debe utilizar al **menos una rama además de la principal**. No se limiten a compartir el trabajo en las diferentes clases, es importante que a veces contribuyan en las mismas clases juntos (uno puede escribir algunos métodos o software y el otro termina el trabajo).

Su código debe ser ejecutable, y debe proporcionar un readme (en git) indicando los nombres completos de los estudiantes.

La fecha límite para la rendición de la tarea en Canvas (el enlace de su código en github) es el lunes 3 de octubre a las 23:59.

Rúbrica de evaluación					
Criterios	Calificaciones				Pts
Adecuación del código al modelo UML	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> El código se corresponde perfectamente con el modelo UML o se justifican las diferencias existentes	<b>4 para &gt;2.0 pts</b> <b>Bien</b> Una gran parte del código corresponde al modelo UML	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> Sólo algunos elementos corresponden al modelo UML	<b>0 pts</b> <b>Falta</b> El código no coincide con el modelo UML	6 pts
Ejecución del código	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> El código se ejecuta sin problemas y el main prueba todo el código como se requiere	<b>4 para &gt;2.0 pts</b> <b>Bien</b> El código se ejecuta sin problemas pero falta algunas pruebas en el main o hay errores menores	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> El código tiene problemas durante la ejecución o la implementación de main está incompleta	<b>0 pts</b> <b>Falta</b> El código no se compila o no se ejecuta	6 pts
Complejidad y calidad de la aplicación	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> La aplicación contiene todo el código necesario y la implementación es correcta	<b>4 para &gt;2.0 pts</b> <b>Bien</b> Faltan algunas partes menores del código o hay problemas menores de implementación	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> Faltan algunas partes importantes del código o hay problemas importantes con la implementación	<b>0 pts</b> <b>Falta</b> Falta gran parte del código	6 pts
Trabajo en grupo	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> El trabajo está bien repartido entre los dos miembros del equipo	<b>4 para &gt;2.0 pts</b> <b>Bien</b> El trabajo podría estar mejor distribuido o falta colaboración en las mismas clases	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> El trabajo está desequilibrado o no hay colaboración	<b>0 pts</b> <b>Falta</b> El trabajo está totalmente desequilibrado	6 pts
Uso de GIT	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> GIT se ha utilizado correctamente (uso de al menos una rama, los tamaños, las frecuencias y los mensajes de los commits son lógicos)	<b>4 para &gt;2.0 pts</b> <b>Bien</b> GIT fue bien utilizado (uso de al menos una rama, pero el tamaño, la frecuencia y los mensajes de los commits podrían mejorarse)	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> Falta el uso de una rama o el tamaño, la frecuencia y los mensajes de los commits podrían mejorarse mucho	<b>0 pts</b> <b>Falta</b> GIT ha sido poco o nada explotado	6 pts
Puntualidad (dependiendo del retraso, la tarea puede no ser evaluada)	<b>6 para &gt;4.0 pts</b> <b>Excelente</b> Entrega el trabajo en la fecha establecida.	<b>4 para &gt;2.0 pts</b> <b>Bien</b> Entrega el trabajo fuera de plazo, con justificación válida y oportuna.	<b>2 para &gt;0.0 pts</b> <b>Por mejorar</b> Entrega el trabajo fuera de plazo, pero con justificación inoportuna.	<b>0 pts</b> <b>Falta</b> Entrega el trabajo fuera del plazo y sin justificación	6 pts
Puntos totales: 36					

## Breve tutorial sobre Git y GitHub (ver curso o enlaces del curso para más detalles)

Git está instalado en las máquinas Linux de las salas de laboratorio,

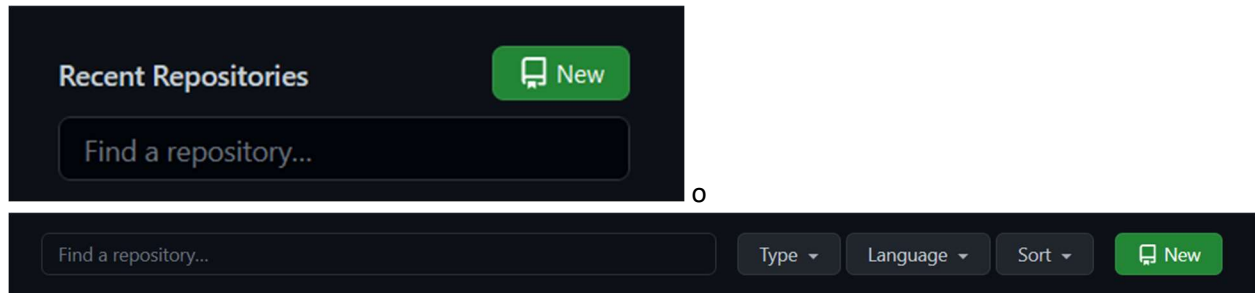
Si quieren instalarlo en su máquina pueden hacerlo desde aquí: <https://git-scm.com/downloads>

(debería ser posible utilizar la versión portátil en las ventanas del laboratorio también - pero no hay garantía)

También deben crear una cuenta en github : <https://github.com/>

Usando **Sign Up**. Respondan a la pregunta y asegúrense de seleccionar la oferta gratuita (normalmente deberían tener una cuenta de estudiante con el correo electrónico de la universidad, pero no es necesario)

Una vez conectado se debe crear un nuevo repositorio (uno por grupo)




o equivalente

Haga clic en new (nuvo)

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

 GeoffreyHecht ▾

Repository name \*

/ tarea1 ✓

Great repository names are short and memorable. Need inspiration? How about [automatic-eureka?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

### Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

### Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Java ▾

### Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

This will set  `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Create repository


Poner un nombre, un readme, un gitignore Java y hacer clic en create repository

Para poder trabajar juntos más fácilmente, hacer click en

Settings

Collaborators

Add people



Add a collaborator to tarea1

Select a collaborator above

Busque a su colega con su nombre de usuario o su nombre

Luego ambas personas pueden clonar el depósito (desde la pagina principal del proyecto)

Go to file

Add file ▾

Code ▾

Clone

?

HTTPS

SSH

GitHub CLI

https://github.com/GeoffreyHecht/tarea1.git

📋

Use Git or checkout with SVN using the web URL.

📂

Open with GitHub Desktop

📦

Download ZIP

Hacer clic en code y copiar la dirección dada que termina en .git

Ahora en el shell bash (o git bash bajo windows) y desde la carpeta donde quieren trabajar

Usar

```
git clone {{ dirección.git }}
```

```
$ git clone https://github.com/GeoffreyHecht/tarea1.git
Cloning into 'tarea1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

coldf@LAPTOP-0UF8SOG9 MINGW64 ~/OneDrive/Documents/NetBeansProjects
$ cd tarea1/
```

Ahora una de las dos personas puede crear un proyecto netbeans apuntando a la misma carpeta (aquí tarea1)

New Java Application

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back   Next >   **Finish**   Cancel   Help



Desde el shell ahora pueden ver y añadir sus primeros archivos con git status, git add y git commit

```
$ git add .

coldf@LAPTOP-0UF8SOG9 MINGW64 ~/OneDrive/Documents/NetBeansProjects/tarea1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore
    new file:   TareaProg/build.xml
    new file:   TareaProg/manifest.mf
    new file:   TareaProg/nbproject/build-impl.xml
    new file:   TareaProg/nbproject/genfiles.properties
    new file:   TareaProg/nbproject/project.properties
    new file:   TareaProg/nbproject/project.xml
    new file:   TareaProg/src/tareaprog/TareaProg.java

coldf@LAPTOP-0UF8SOG9 MINGW64 ~/OneDrive/Documents/NetBeansProjects/tarea1 (main)
$ git commit -m "Estructura del proyecto y primera clase main"
[main 49afacb] Estructura del proyecto y primera clase main
 8 files changed, 1987 insertions(+)
 create mode 100644 TareaProg/build.xml
 create mode 100644 TareaProg/manifest.mf
 create mode 100644 TareaProg/nbproject/build-impl.xml
 create mode 100644 TareaProg/nbproject/genfiles.properties
 create mode 100644 TareaProg/nbproject/project.properties
 create mode 100644 TareaProg/nbproject/project.xml
 create mode 100644 TareaProg/src/tareaprog/TareaProg.java
```

Luego puedes subirlos a github con git push

```
$ git push
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 15.73 KiB | 5.24 MiB/s, done.
Total 14 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/GeoffreyHecht/tarea1.git
 58146c8..49afacb  main -> main
```

Dependiendo de su instalación y de su sistema operativo, tendrá que iniciar la sesión a través del sitio web (Windows, Mac o algunas versiones de Linux) o con las credenciales en el Shell.

**En este último caso, consulte esta documentación :**

**<https://docs.github.com/es/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>**

Su colega puede entonces clonar el proyecto y trabajar desde su máquina (o hacer un pull si ya tiene un clon).

Ahora puedes trabajar con los comandos vistos en el curso (add, branch, checkout, merge, status...). **No olviden hacer commits/pushes/pulls regulares con mensajes claros y útiles.**

No dude en pedir ayuda a los ayudantes o a mí.

**No olviden dar la dirección de su repositorio** de GitHub (sin el .git, sólo la página) en Canvas

**No olviden poner su nombres completos en el readme de GitHub**