



CICLO: [DAM]
**MÓDULO DE [ENTORNOS DE
DESARROLLO]**

[Tarea N° 06]

Alumno:
[Juan Carlos Filter Martín]
[REDACTED]

Contenido

1. Documentos que se adjuntan a este informe.....	3
2. Crear proyecto.....	3
A) Proyecto tipo ant.....	3
B) Clase java Tarea6ED.....	4
La clase Tarea6ED contiene el siguiente código.....	4
C) Clase Main.....	5
La clase Main contiene el siguiente código.....	5
3. (RA04_c) Se ha revisado el código fuente usando un analizador de código.....	6
A) Instalar el plugin SonarLint en NetBeans.....	6
4. (RA04_e) Se han aplicado patrones de refactorización con las herramientas que proporciona el entorno de desarrollo.....	8
A) Realizar una refactorización de renombrado.....	8
B) Realizar una refactorización de extracción de método.....	9
C) Realizar una refactorización de extracción de constante para eliminar números mágicos.....	11
Constante para Descuento con mas de 3 productos.....	12
Constante para Descuento diferente a 0.....	13
Constante para Descuento igual a 0.....	14
5. Comentar la clase y documentación JavaDoc.....	15
A) Clase Java Tarea6ED.....	15
B) Clase Java MAIN.....	16
6. (RA04_h) Se han utilizado repositorios remotos para el desarrollo de código colaborativo.....	16
A) Crearemos un repositorio en github (https://github.com) para alojar este código en él.....	16
7. (RA04_f) Se ha realizado el control de versiones integrado en el entorno de desarrollo.....	20
A) Conectamos Netbeans al repositorio github creado a través del control de versiones git incorporado en Netbeans.....	20
Inicializar repositorio.....	20
Clonar repositorio.....	22
B) Sincronizamos Netbeans con github y mostramos el código fuente a través de la pagina de web de github.....	25
Realizar Commit.....	25
Mostrar el código fuente a través de la pagina de web de github.....	29

1. Documentos que se adjuntan a este informe.

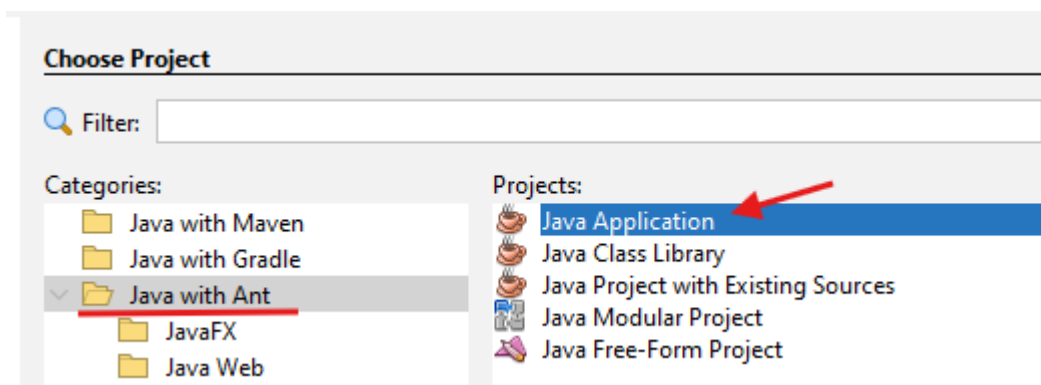
A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto java

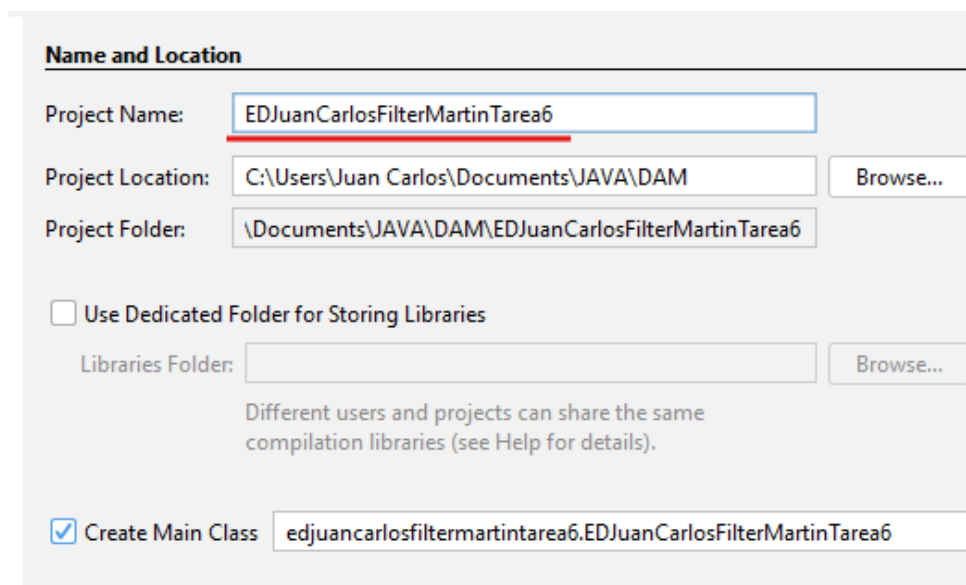
2. Crear proyecto

A) Proyecto tipo ant.

File > New Project > Java with Ant > **Java Application**



Indicamos un nombre al proyecto



B) Clase java Tarea6ED.

Se va a crear una clase aparte de la clase main

Botón derecho sobre el proyecto New > **Java Class**

Name and Location	
Class Name:	<input type="text" value="Tarea6ED"/>
Project:	<input type="text" value="EDJuanCarlosFilterMartinTarea6"/>
Location:	<input type="text" value="Source Packages"/>
Package:	<input type="text"/>
Created File:	<input type="text" value="Carlos\Documents\JAVA\DAM\EDJuanCarlosFilterMartinTarea6\src\Tarea6ED.java"/>

× La clase Tarea6ED contiene el siguiente código.

```
public class Tarea6ED {  
  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double Total;  
        if (numProductos > 3) {  
            precioProducto -= 5;  
        }  
        if (numProductos != 0) {  
            Total = precioProducto * 0.8;  
            System.out.println("El total a pagar es:" + Total);  
            System.out.println("Enviado");  
        } else {  
            Total = precioProducto * 0.95;  
            System.out.println("El total a pagar es:" + Total);  
            System.out.println("Enviado");  
        }  
    }  
}
```

C) Clase Main.

Posteriormente creamos una clase Main

Name and Location	
Class Name:	<input type="text" value="EDJuanCarlosFilterMartinTarea6"/>
Project:	<input type="text" value="EDJuanCarlosFilterMartinTarea6"/>
Location:	<input type="text" value="Source Packages"/>
Package:	<input type="text"/>
Created File:	<input type="text" value="VA\DAM\EDJuanCarlosFilterMartinTarea6\src\EDJuanCarlosFilterMartinTarea6.java"/>

✖ La clase Main contiene el siguiente código.

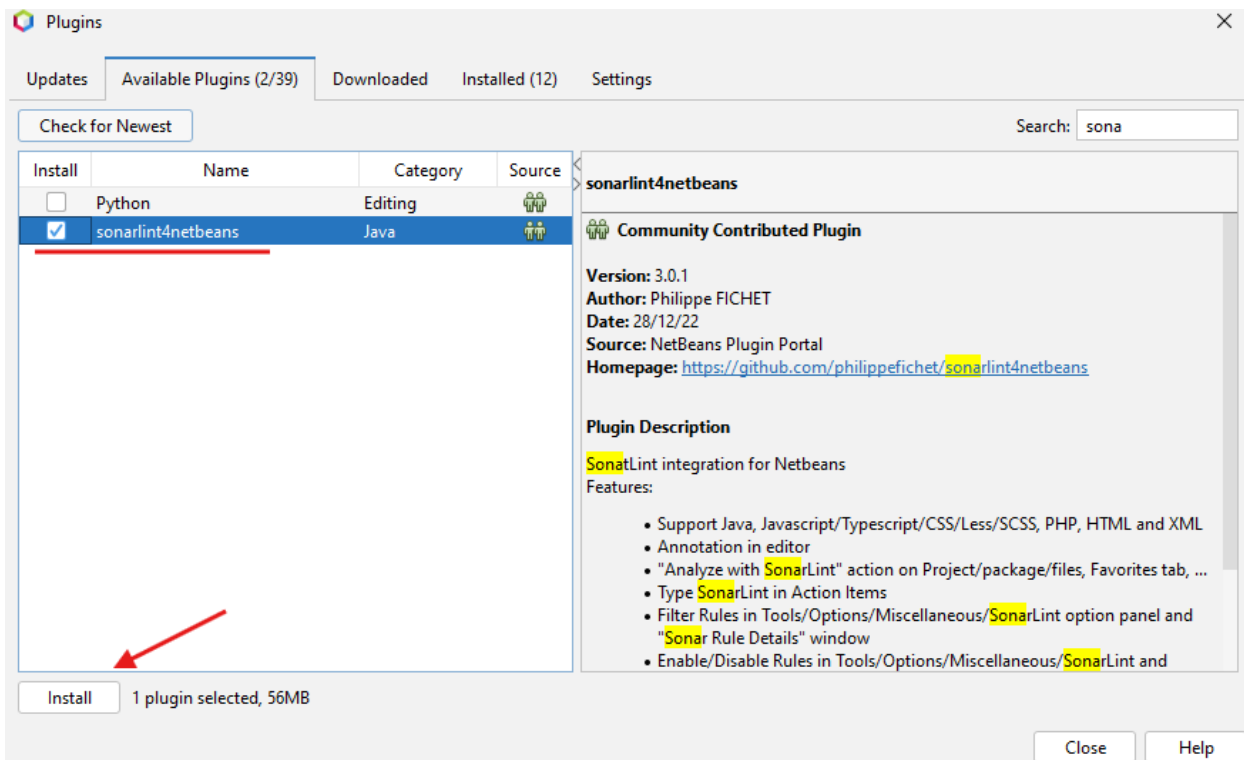
```
public class EDJuanCarlosFilterMartinTarea6 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        Tarea6ED miCarrito = new Tarea6ED();  
  
        miCarrito.aplicarDescuento(precioProducto: 100, numProductos: 5);  
    }  
}
```

3. (RA04_c) Se ha revisado el código fuente usando un analizador de código.

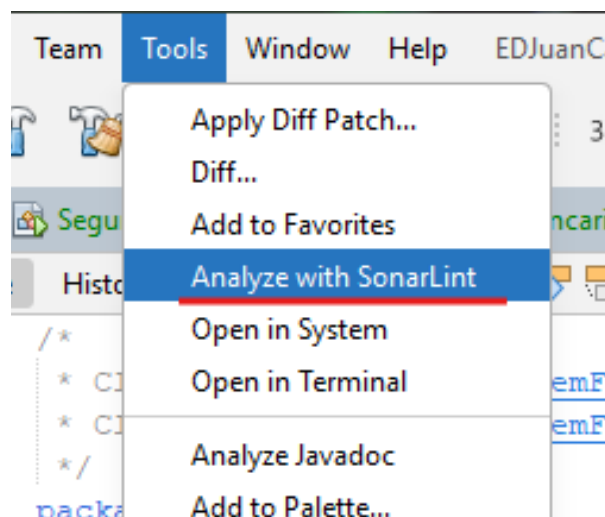
A) Instalar el plugin SonarLint en NetBeans.

Para instalarlo vamos a : Tools > **plugins**

Buscamos SonarLint4NetBeans y al pulsar “install” simplemente se abrirá un asistente para su instalación.



Una vez instalado solo tenemos que ir a **Tools /Analyze with SonarLint**



Este nos mostrará el código con los problemas encontrados.

Aparte este no dará una descripción para solucionar los errores.

```
11 public class Tarea6ED {
12
13     public void aplicarDescuento(double precioProducto, int numProductos) {
14         double Total;
15         if (numProductos > 3) {
16             precioProducto -= 5;
17         }
18         if (numProductos != 0) {
19             Total = precioProducto * 0.8;
20             System.out.println("El total a pagar es:" + Total);
21             System.out.println(x: "Enviado");
22         } else {
23             Total = precioProducto * 0.95;
24             System.out.println("El total a pagar es:" + Total);
25             System.out.println(x: "Enviado");
26         }
27     }
28 }
```

Output x SonarLint Analyzer Window x

Tarea6ED.java x EDJuanCarlosFilterMartinTarea6.java x

Nodes

⚠ Analyze done, 1 issue found

> 🟢 minor (1 issue)

Sonar Rule Details Window x

Global settings x

java:S106

Browse Source

Standard outputs should not be used directly to log anything

java:S106 ⚠ MAJOR ☹ CODE_SMELL 🔍 [bad-practice](#), [cert](#), [owasp-a3](#)

When logging a message there are several important requirements which must be fulfilled:

- The user must be able to easily retrieve the logs
- The format of all logged message must be uniform to allow the user to easily read the log
- Logged data must actually be recorded
- Sensitive data must only be logged securely

If a program directly writes to the standard outputs, there is absolutely no way to comply with those requirements. That's why defining and using a dedicated logger is highly recommended.

Noncompliant Code Example

```
System.out.println("My Message"); // Noncompliant
```

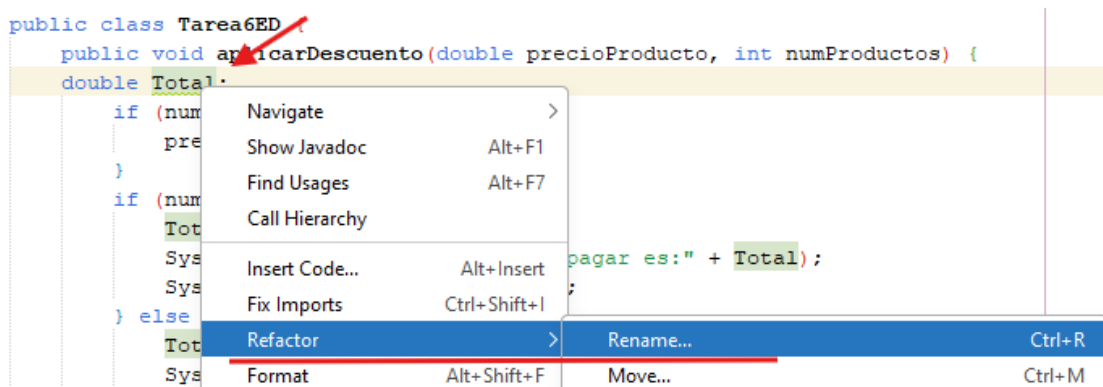
Si necesitamos más información sobre este error podemos hacer click sobre el título que aparece y nos mandará a una web donde queda explicado

4. (RA04_e) Se han aplicado patrones de refactorización con las herramientas que proporciona el entorno de desarrollo.

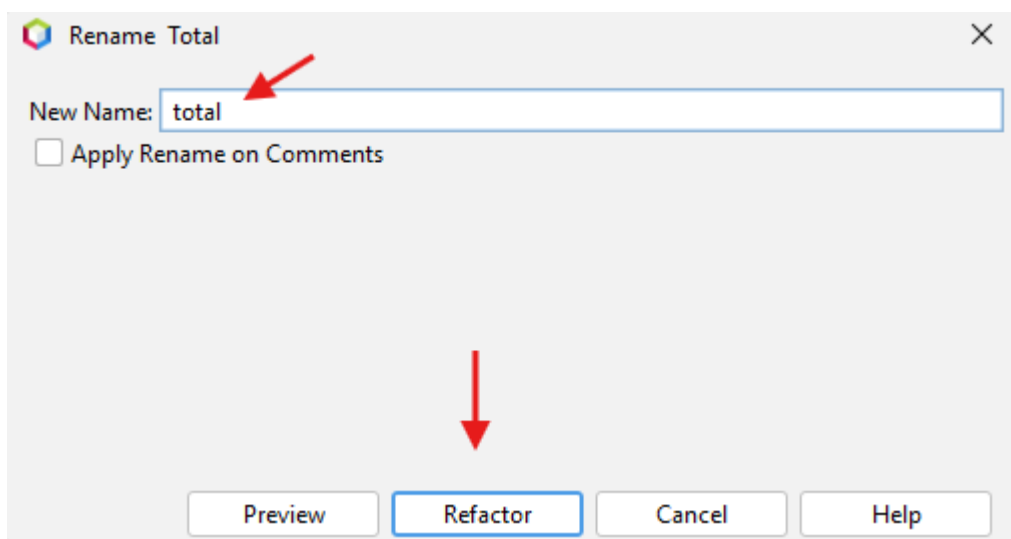
A) Realizar una refactorización de renombrado.

Refactorizamos el nombre de la variable **Total** a Formato loweCamelCase.

Sobre la variable: Pulsamos **botón derecho > Refactor > Rename**.



Modificamos el nombre y aplicamos Refactor.



Automáticamente se aplica el refactor a la variable en todos los lugares donde se encuentre.

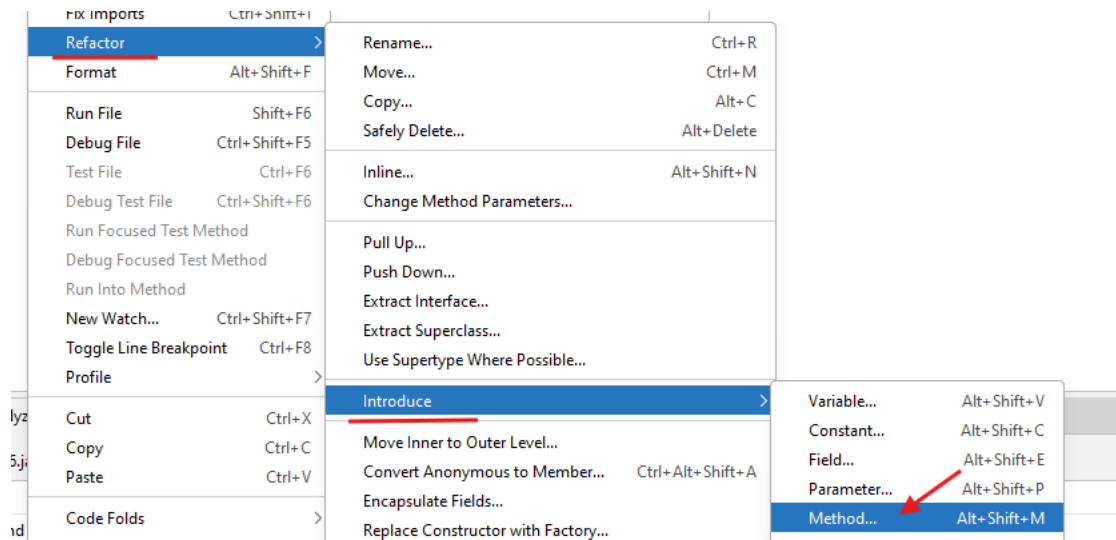
```
public class Tarea6ED {  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double total;  
        if (numProductos > 3) {  
            precioProducto -= 5;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * 0.8;  
            System.out.println("El total a pagar es:" + total);  
            System.out.println(x: "Enviado");  
        } else {  
            total = precioProducto * 0.95;  
            System.out.println("El total a pagar es:" + total);  
            System.out.println(x: "Enviado");  
        }  
    }  
}
```

B) Realizar una refactorización de extracción de método.

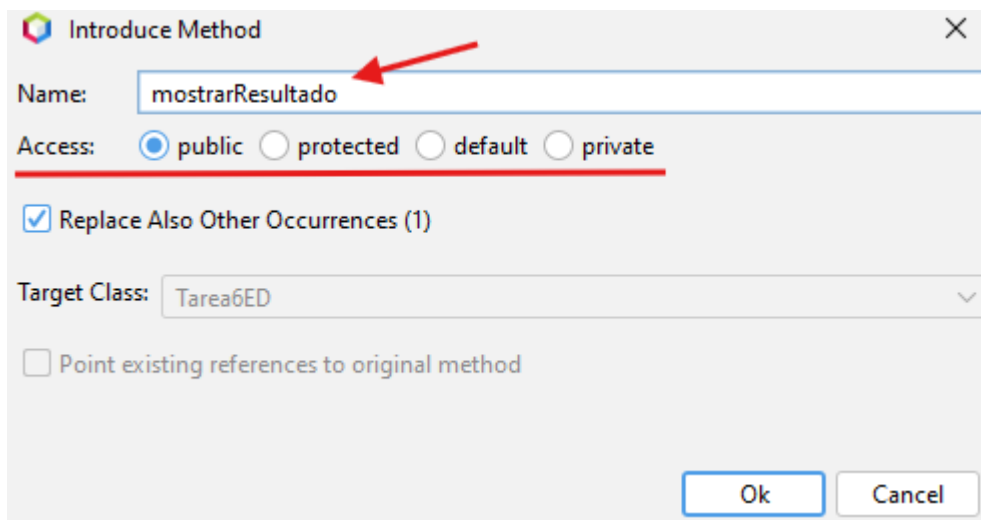
Seleccionamos la parte del código redundante que vamos a extraer en un nuevo método.

```
        if (numProductos > 3) {  
            precioProducto -= 5;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * 0.8;  
            System.out.println("El total a pagar es:" + total);  
            System.out.println(x: "Enviado");  
        } else {  
            total = precioProducto * 0.95;  
            System.out.println("El total a pagar es:" + total);  
            System.out.println(x: "Enviado");  
        }  
    }  
}
```

Pulsamos **botón derecho > Refactor > Introduce > Method**



Introducimos un nombre y aparte también tenemos la opción de elegir el tipo de acceso que tendrá el método.



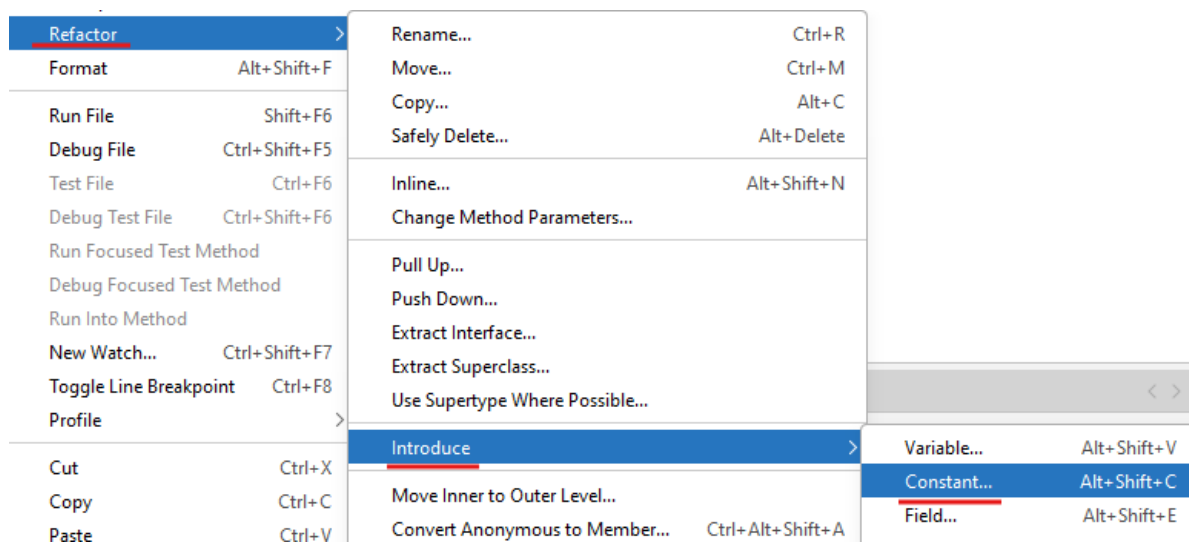
Se habrá creado el nuevo método y aplicado automáticamente donde antes estaba el código reutilizado.

```
public class Tarea6ED {  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double total;  
  
        if (numProductos > 3) {  
            precioProducto -= 5;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * 0.8;  
            mostrarResultado(total);  
        } else {  
            total = precioProducto * 0.95;  
            mostrarResultado(total);  
        }  
    }  
}  
  
public void mostrarResultado(double total) {  
    System.out.println("El total a pagar es:" + total);  
    System.out.println("Enviado");  
}
```

C) Realizar una refactorización de extracción de constante para eliminar números mágicos.

Para realizar la refactorización de extracción de constante la forma es la siguiente:

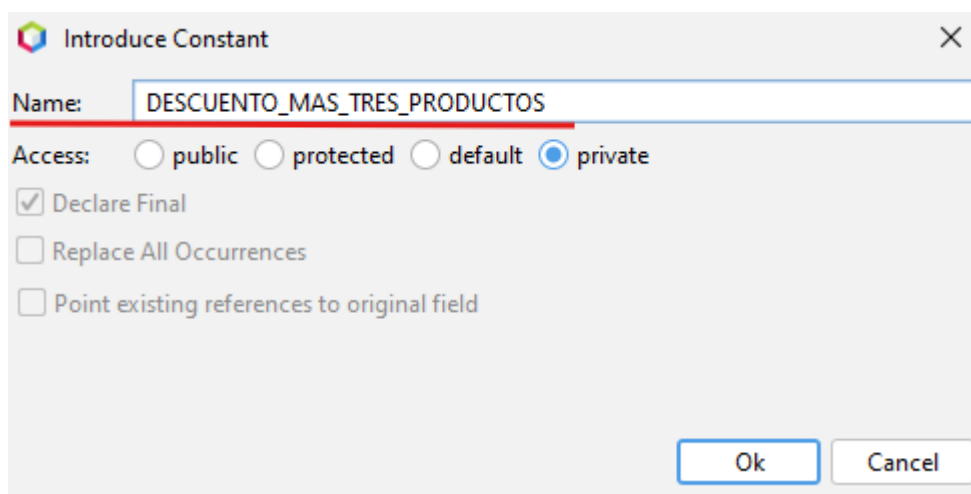
Pulsamos botón derecho sobre un número mágico > Refactor > Introduce > Constant.



× **Constante para Descuento con mas de 3 productos.**

```
if (numProductos > 3) {  
    precioProducto -= 5;  
}
```

Aplicamos la refactorización asignándole un nombre a este.



The dialog box titled "Introduce Constant" is shown. It has a close button (X) in the top right corner. The "Name:" field contains the text "DESCUENTO_MAS_TRES_PRODUCTOS". Below this, there are radio buttons for "Access": "public", "protected", "default", and "private", with "private" selected. There are three checkboxes: "Declare Final" (checked), "Replace All Occurrences" (unchecked), and "Point existing references to original field" (unchecked). At the bottom right, there are "Ok" and "Cancel" buttons.

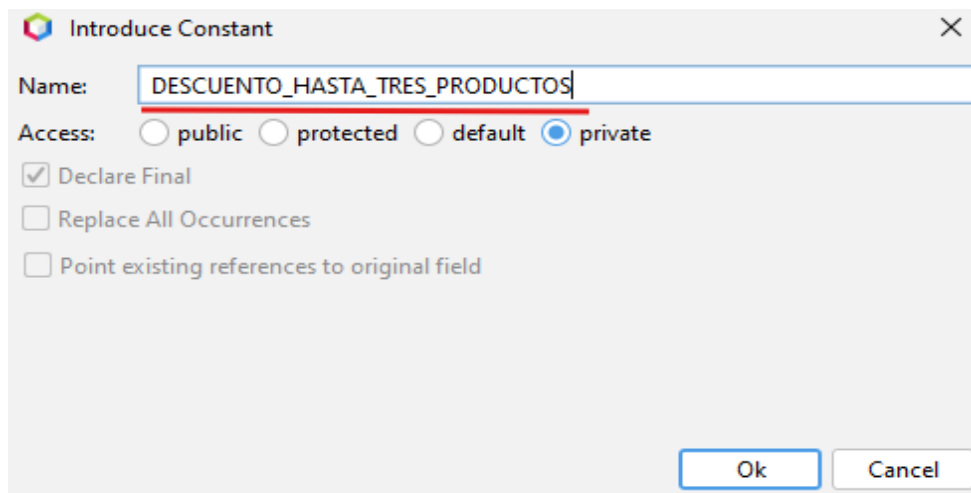
Tendríamos la Constante creada

```
public class Tarea6ED {  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double total;  
  
        if (numProductos > 3) {  
            precioProducto -= DESCUENTO_MAS_TRES_PRODUCTOS;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * 0.8;  
            mostrarResultado(total);  
        } else {  
            total = precioProducto * 0.95;  
            mostrarResultado(total);  
        }  
    }  
    private static final int DESCUENTO_MAS_TRES_PRODUCTOS = 5;  
  
    public void mostrarResultado(double total) {  
        System.out.println("El total a pagar es:" + total);  
        System.out.println("Enviado");  
    }  
}
```

× **Constante para Descuento diferente a 0.**

```
if (numProductos != 0) {  
    total = precioProducto * 0.8;  
    mostrarResultado(total);  
} else {
```

Aplicamos la refactorización asignándole un nombre a este.



The dialog box titled "Introduce Constant" is shown. It has a close button (X) in the top right corner. The "Name:" field contains the text "DESCUENTO_HASTA_TRES_PRODUCTOS". Below this, the "Access:" section has four radio buttons: "public", "protected", "default", and "private", with "private" selected. There are three checkboxes: "Declare Final" (checked), "Replace All Occurrences" (unchecked), and "Point existing references to original field" (unchecked). At the bottom right, there are "Ok" and "Cancel" buttons.

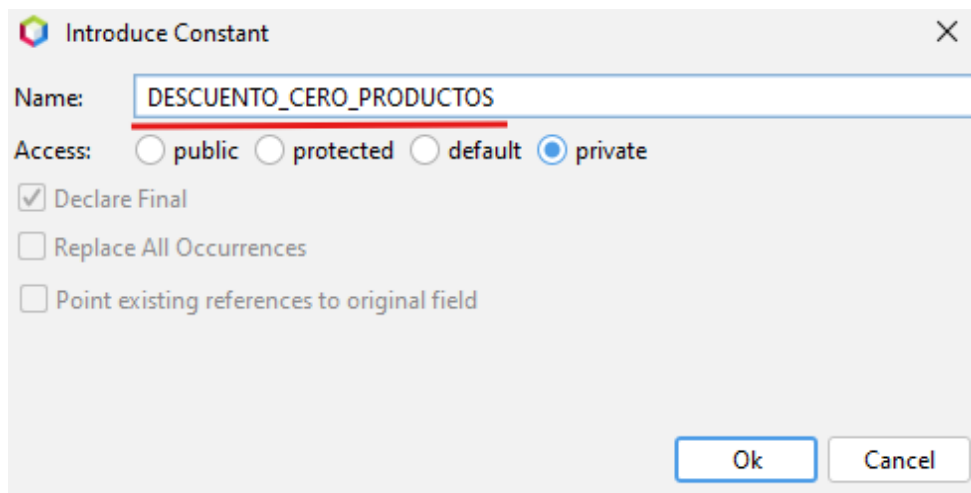
Tendríamos la Constante creada

```
public class Tarea6ED {  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double total;  
  
        if (numProductos > 3) {  
            precioProducto -= DESCUENTO_MAS_TRES_PRODUCTOS;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * DESCUENTO_HASTA_TRES_PRODUCTOS;  
            mostrarResultado(total);  
        } else {  
            total = precioProducto * 0.95;  
            mostrarResultado(total);  
        }  
    }  
  
    private static final double DESCUENTO_HASTA_TRES_PRODUCTOS = 0.8;  
    private static final int DESCUENTO_MAS_TRES_PRODUCTOS = 5;  
  
    public void mostrarResultado(double total) {  
        System.out.println("El total a pagar es:" + total);  
        System.out.println("x: " + "Enviado");  
    }  
}
```

× Constante para Descuento igual a 0.

```
    } else {  
        total = precioProducto * 0.95;  
        mostrarResultado(total);  
    }  
}
```

Aplicamos la refactorización asignándole un nombre a este.



The dialog box titled "Introduce Constant" is shown. It has a close button (X) in the top right corner. The "Name:" field contains the text "DESCUENTO_CERO_PRODUCTOS". Below the name field, there are radio buttons for "Access": "public", "protected", "default", and "private". The "private" option is selected. There are three checkboxes: "Declare Final" (checked), "Replace All Occurrences" (unchecked), and "Point existing references to original field" (unchecked). At the bottom right, there are "Ok" and "Cancel" buttons.

Tendríamos la Constante creada

```
public class Tarea6ED {  
    public void aplicarDescuento(double precioProducto, int numProductos) {  
        double total;  
  
        if (numProductos > 3) {  
            precioProducto -= DESCUENTO_MAS_TRES_PRODUCTOS;  
        }  
        if (numProductos != 0) {  
            total = precioProducto * DESCUENTO_HASTA_TRES_PRODUCTOS;  
            mostrarResultado(total);  
        } else {  
            total = precioProducto * DESCUENTO_CERO_PRODUCTOS;  
            mostrarResultado(total);  
        }  
    }  
  
    private static final double DESCUENTO_CERO_PRODUCTOS = 0.95;  
    private static final double DESCUENTO_HASTA_TRES_PRODUCTOS = 0.8;  
    private static final int DESCUENTO_MAS_TRES_PRODUCTOS = 5;  
  
    public void mostrarResultado(double total) {  
        System.out.println("El total a pagar es:" + total);  
        System.out.println("x: " + "Enviado");  
    }  
}
```

5. Comentar la clase y documentación JavaDoc

Se va a comentar el código de la siguiente forma:

A) Clase Java Tarea6ED.

En la clase Tarea6ED se ha comentado y se ha aplicado documentación JavaDoc a los método explicando su función.

```
/**
 *Clase para aplicar descuentos a los productos.
 *
 * @author Juan Carlos
 */
public class Tarea6ED {

    //Constantes
    private static final double DESCUENTO_CERO_PRODUCTOS = 0.95;
    private static final double DESCUENTO_HASTA_TRES_PRODUCTOS = 0.8;
    private static final int DESCUENTO_MAS_TRES_PRODUCTOS = 5;

    /**
     * Aplicamos un descuento al producto segun el numero de productos.
     *
     * @param precioProducto Precio del producto.
     * @param numProductos Numeros de productos.
     */
    public void aplicarDescuento(double precioProducto, int numProductos) {
        double total;

        if (numProductos > 3) {
            precioProducto -= DESCUENTO_MAS_TRES_PRODUCTOS;
        }
        if (numProductos != 0) {
            total = precioProducto * DESCUENTO_HASTA_TRES_PRODUCTOS;
            mostrarResultado(total);
        } else {
            total = precioProducto * DESCUENTO_CERO_PRODUCTOS;
            mostrarResultado(total);
        }
    }

    /**
     * Muestra el resultado de precio a pagar
     *
     * @param total Precio total a pagar
     */
    public void mostrarResultado(double total) {
        System.out.println("El total a pagar es: " + total);
        System.out.println("x: " + "Enviado");
    }
}
```

B) Clase Java MAIN.

En la clase main se ha comentado cuando se crea el objeto de la clase y la llamada al método.

```
/**
 *
 * @author Juan Carlos
 */
public class EDJuanCarlosFilterMartinTarea6 {

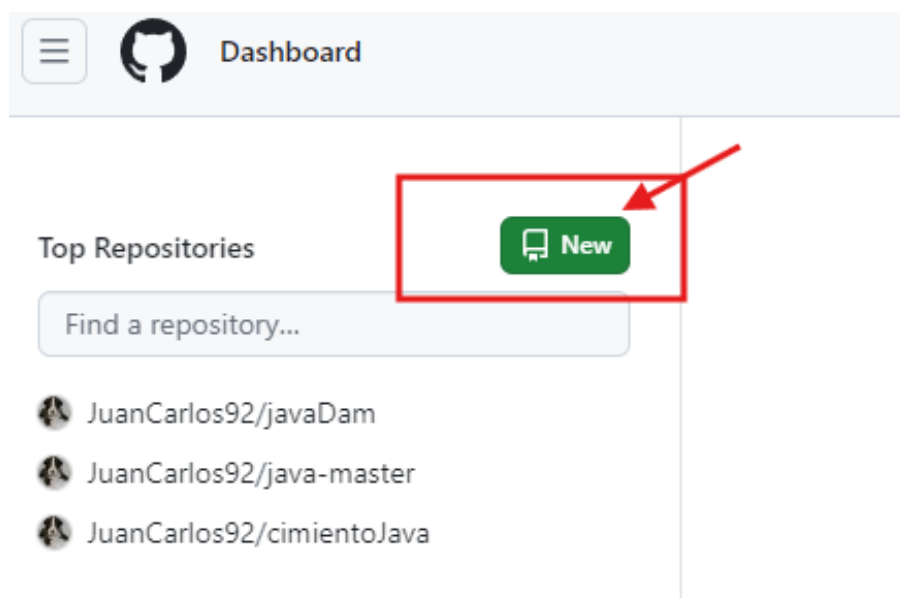
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Objeto de la clase Tarea6ED
        Tarea6ED miCarrito = new Tarea6ED();

        //Llamamos al método para aplicar el descuento
        miCarrito.aplicarDescuento(precioProducto: 100, numProductos: 5);
    }
}
```

6. (RA04_h) Se han utilizado repositorios remotos para el desarrollo de código colaborativo.

A) Crearemos un repositorio en github (<https://github.com>) para alojar este código en él.

En github.com creamos un nuevo repositorio



Asignamos un nombre

También tenemos la opción de elegir que el repositorio sea publico o privado


Y por ultimo pulsamos en **crear repositorio**.

Create a new repository

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

Required fields are marked with an asterisk (*).



Owner * Repository name *

 JuanCarlos92 /

✓ DamRepository is available.

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

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:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None** ▾

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

Choose a license

License: **None** ▾

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

 You are creating a public repository in your personal account.

Create repository

El repositorio ya estaría creado

Popular repositories

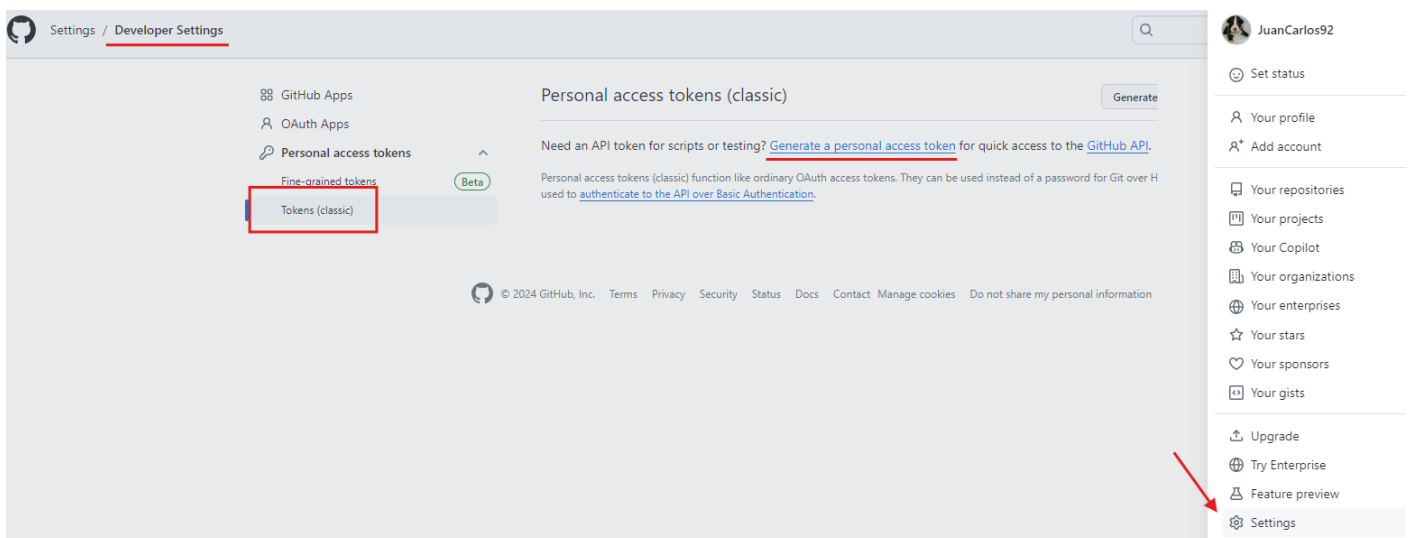
[Customize your pins](#)



Ahora se va a **generar una Personal Access Token** ya que es muy probable que se pida al conectar git con este repositorio.

Para esto vamos a Setting > Developer Setting > **Token (classic)**

Generate a personal access token



Indicamos una nota (opcional), podemos indicarle días para su expiración y **marcamos la opción “repo”**

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens (Beta)
Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note
Conectar Git

What's this token for?

Expiration *
30 days The token will expire on Tue, May 21 2024

Select scopes
Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry

Y tendríamos el token generado

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens (Beta)
Tokens (classic)

Personal access tokens (classic)

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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 Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

7. (RA04_f) Se ha realizado el control de versiones integrado en el entorno de desarrollo.

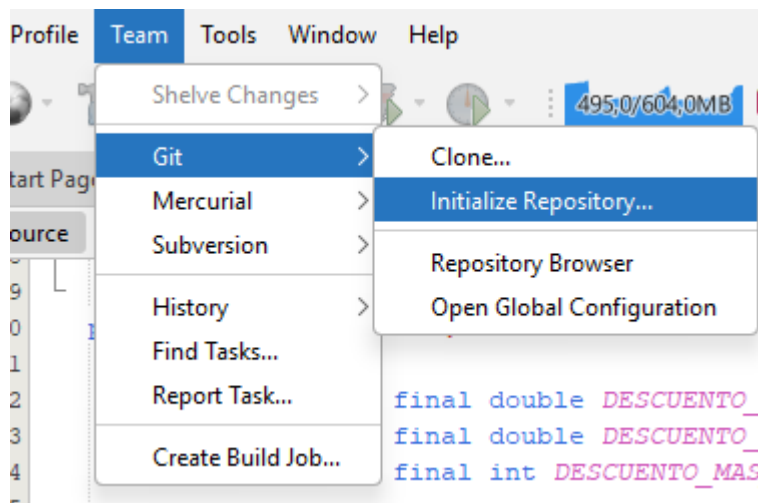
A) Conectamos Netbeans al repositorio github creado a través del control de versiones git incorporado en Netbeans.

× **Inicializar repositorio.**

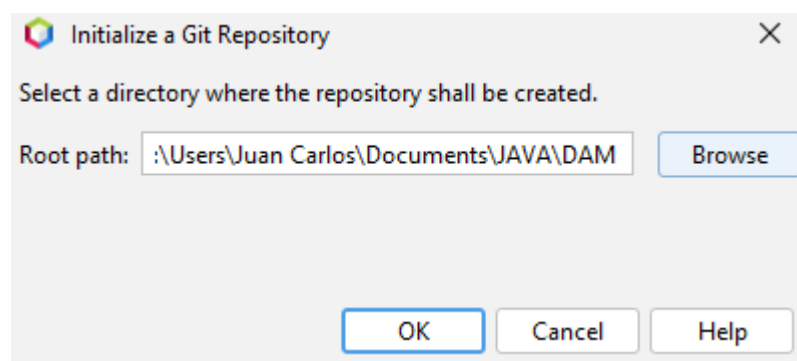
En primer lugar hay que **inicializar el repositorio**.

Los pasos a seguir son:

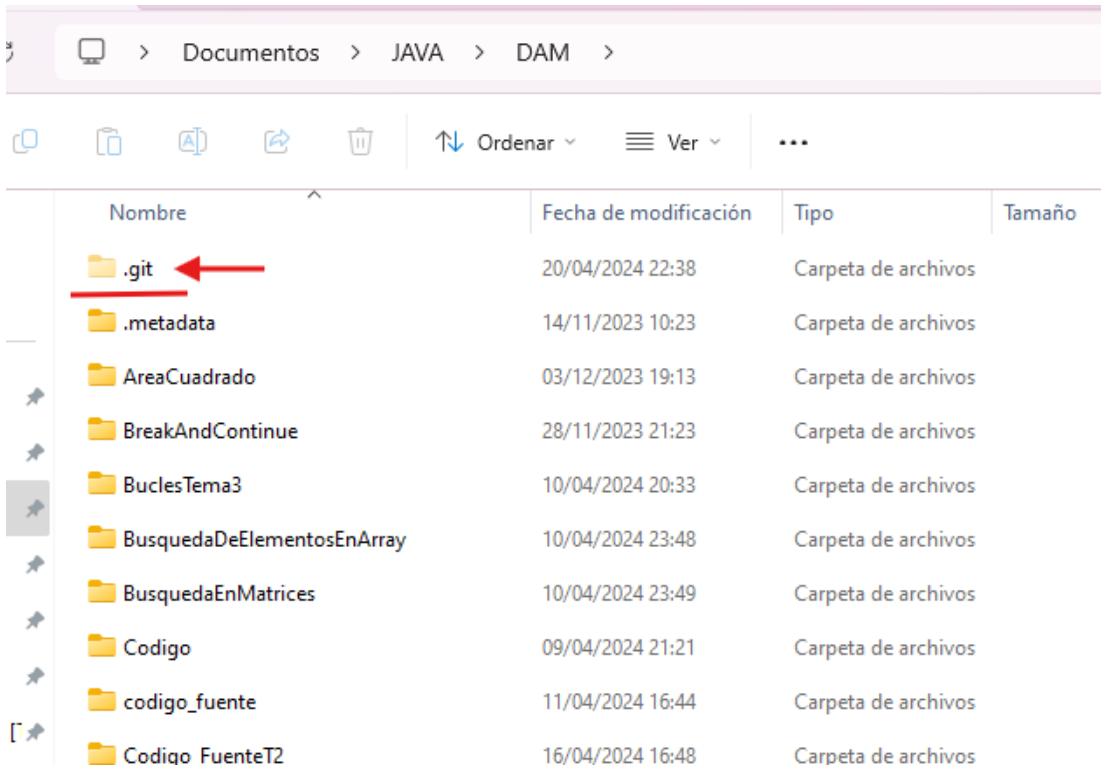
Team > Git > **Inicializar Repositorio**



Indicamos la ruta del directorio donde se encuentra el proyecto



Si vamos al directorio donde se encuentra el proyecto podemos ver una **capeta en oculto** llamada **.git**



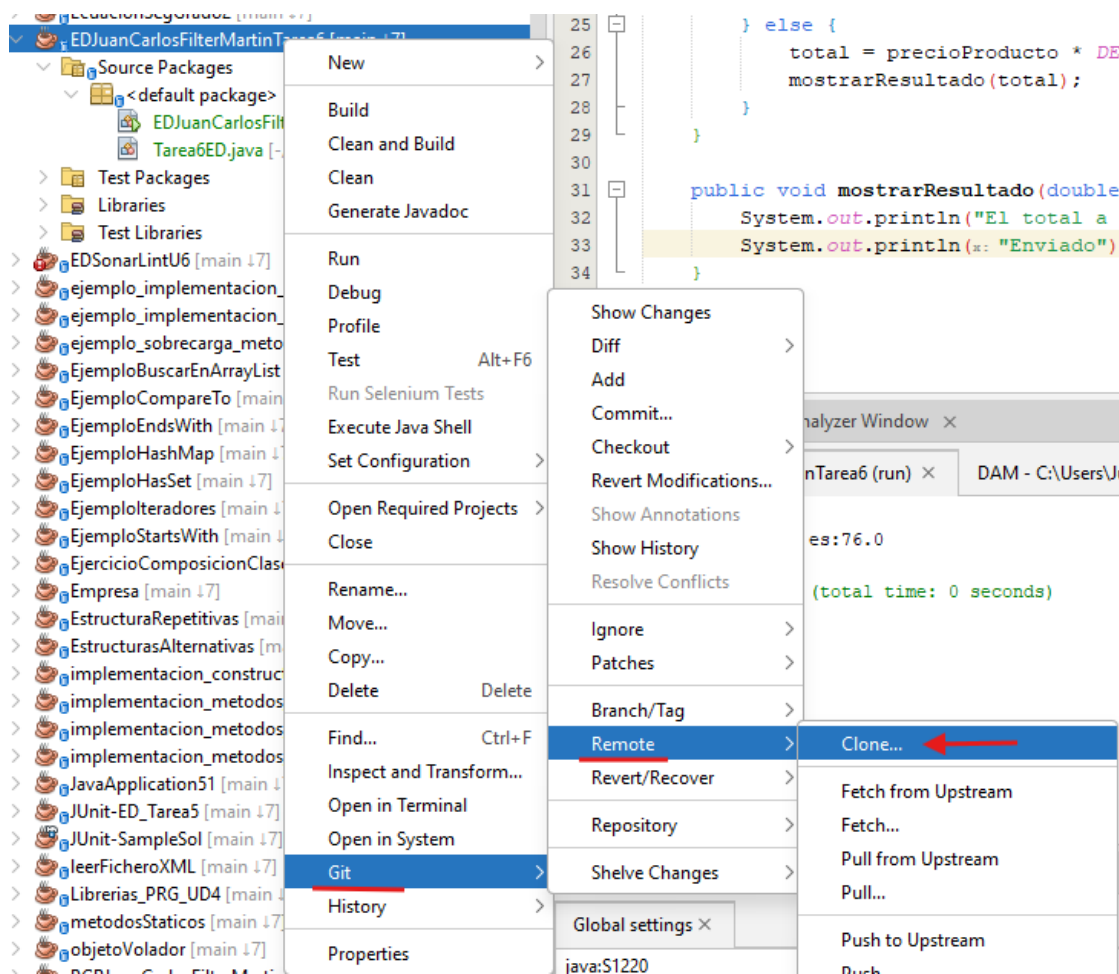
Documentos > JAVA > DAM >				
Ordenar Ver ...				
Nombre Fecha de modificación Tipo Tamaño				
.git	20/04/2024 22:38	Carpeta de archivos		
.metadata	14/11/2023 10:23	Carpeta de archivos		
AreaCuadrado	03/12/2023 19:13	Carpeta de archivos		
BreakAndContinue	28/11/2023 21:23	Carpeta de archivos		
BuclesTema3	10/04/2024 20:33	Carpeta de archivos		
BusquedaDeElementosEnArray	10/04/2024 23:48	Carpeta de archivos		
BusquedaEnMatrices	10/04/2024 23:49	Carpeta de archivos		
Codigo	09/04/2024 21:21	Carpeta de archivos		
codigo_fuente	11/04/2024 16:44	Carpeta de archivos		
Codigo FuenteT2	16/04/2024 16:48	Carpeta de archivos		

✖ Clonar repositorio.

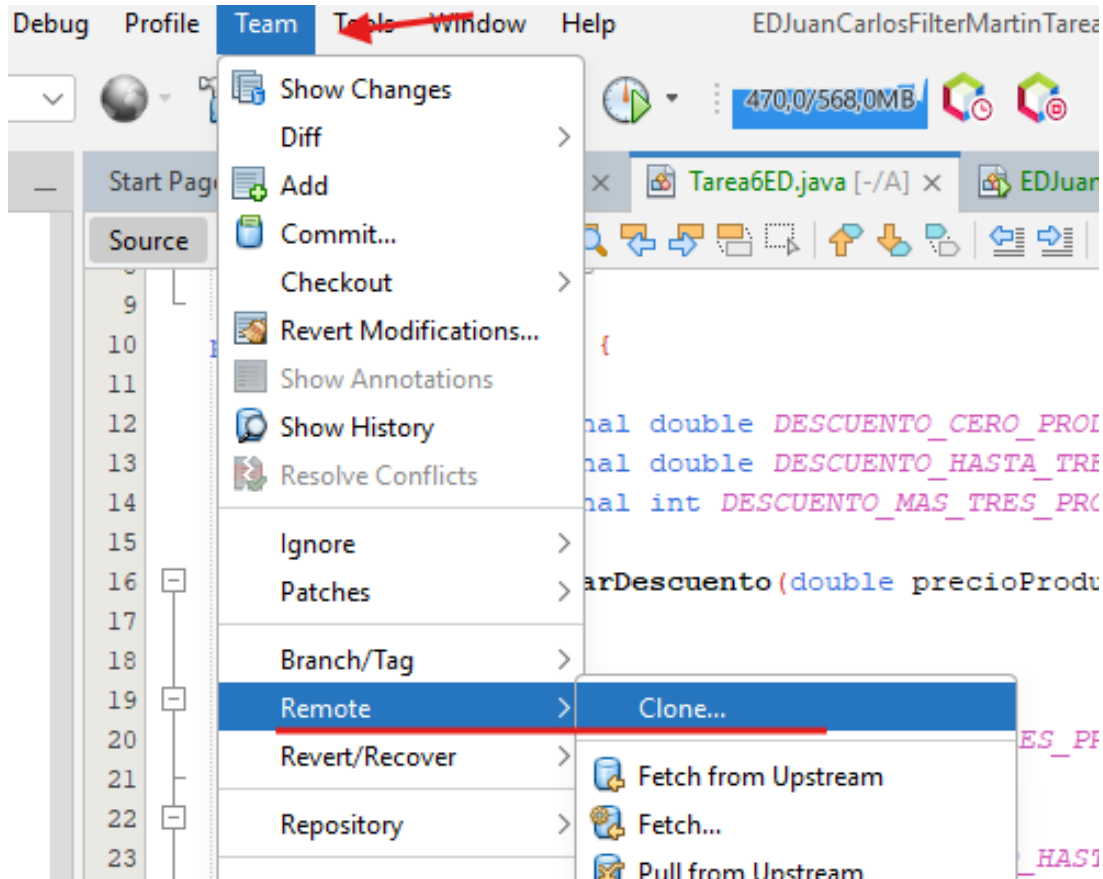
Ahora si queremos clonar el repositorio remoto en GitHub

Se puede hacer de dos formas. Los pasos son los siguientes:

1. Botón derecho en el proyecto > Git > Remote > **Clone**

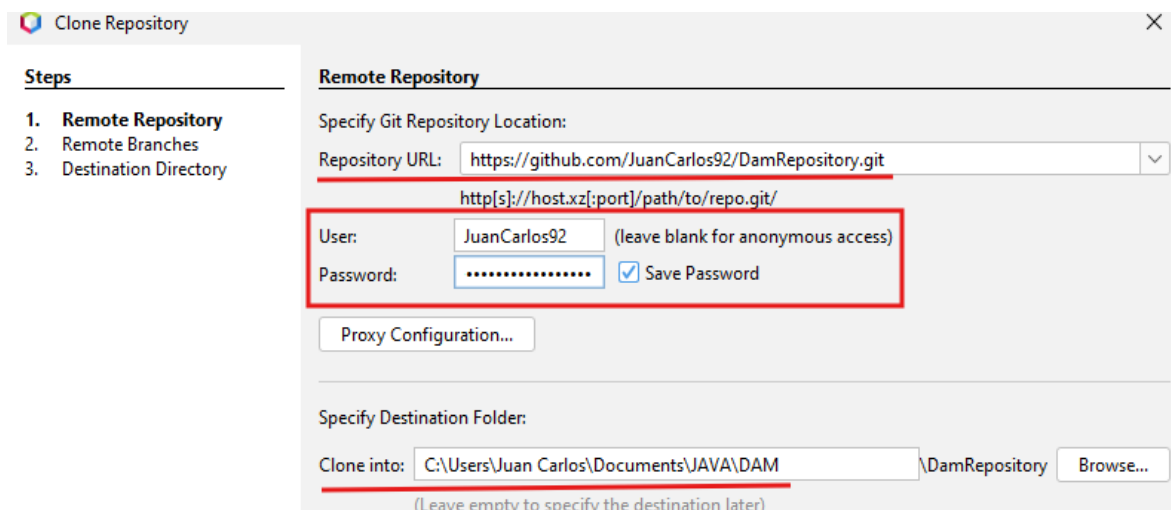


2. Team > Remote > **Clone**

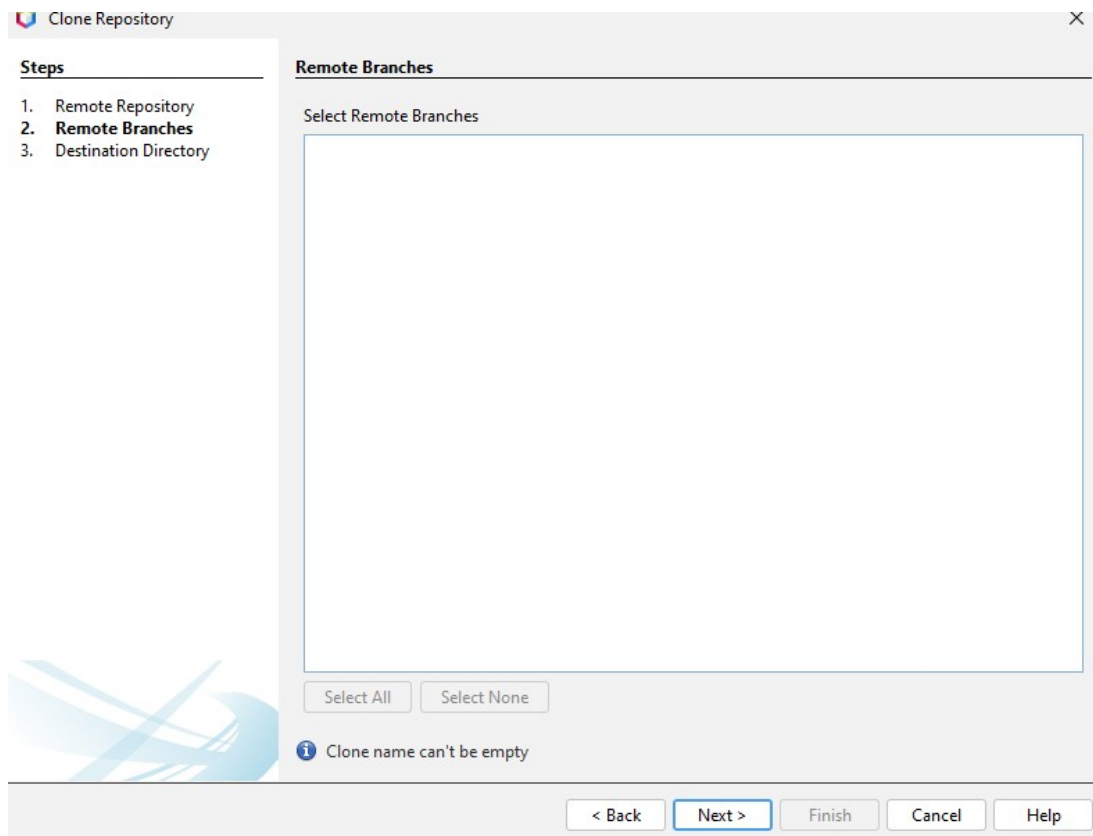


En esta ventana es para **indicar el repositorio remoto**:

Tenemos que indicar la URL del repositorio, el usuario, la contraseña (token generado)

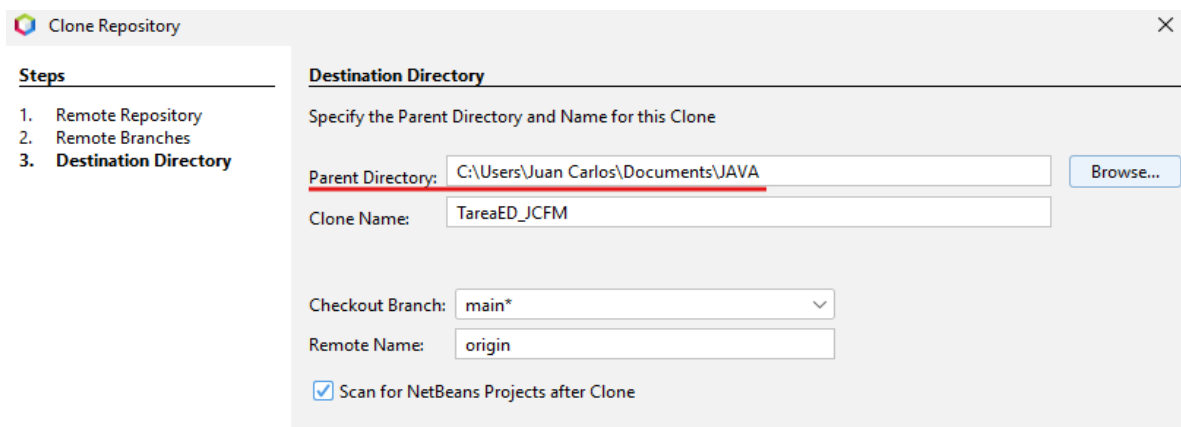


Indicamos la rama (Branch) que queremos copiar del repositorio remoto pero como aun no se ha subido nada entonces no aparece ninguna opción para marcar



En esta ventana es para **indicar el directorio de destino**

Nos pedirá el destino del directorio donde se encuentra en nuestro PC el proyecto

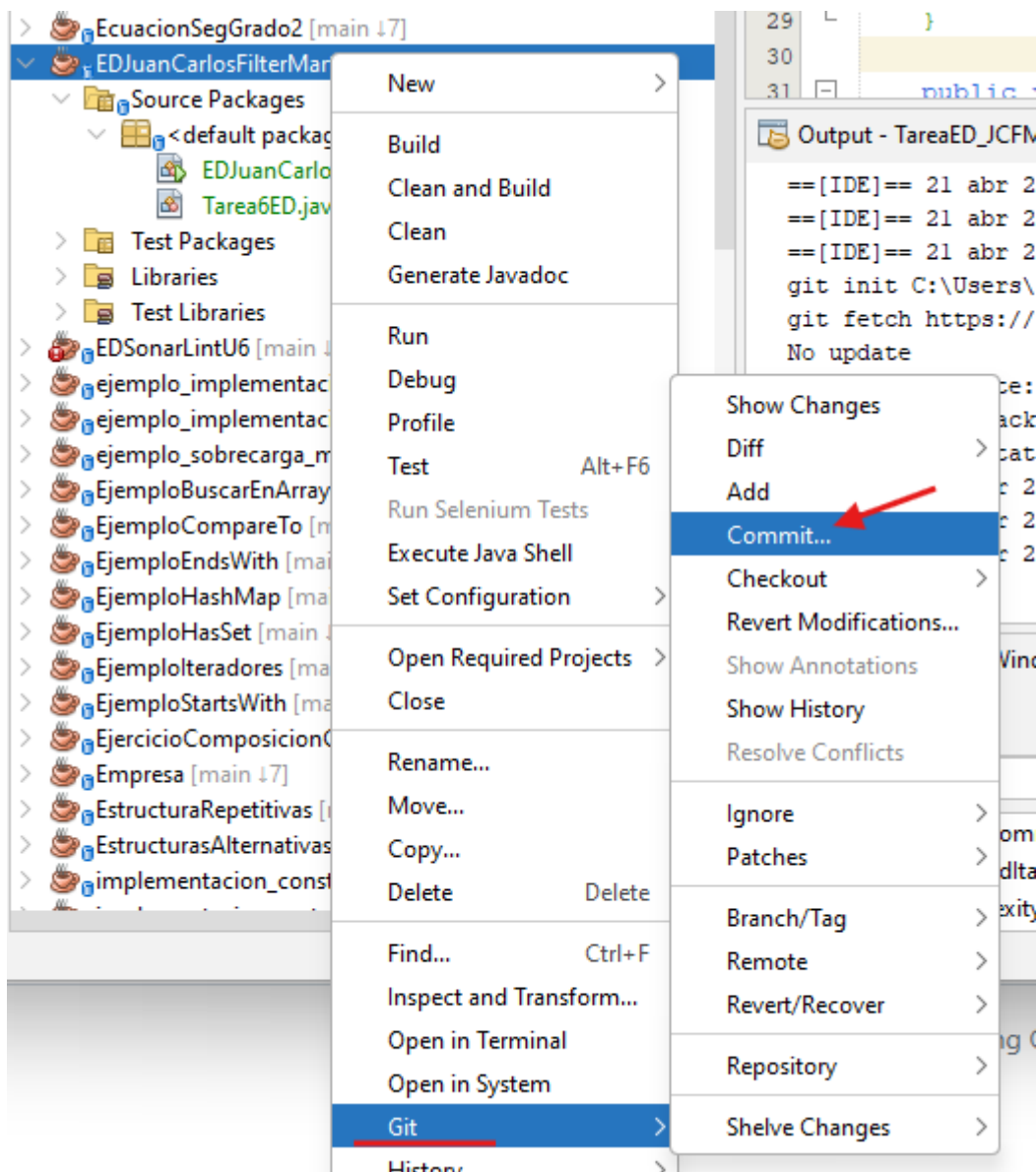


B) Sincronizamos Netbeans con github y mostramos el código fuente a través de la pagina de web de github.

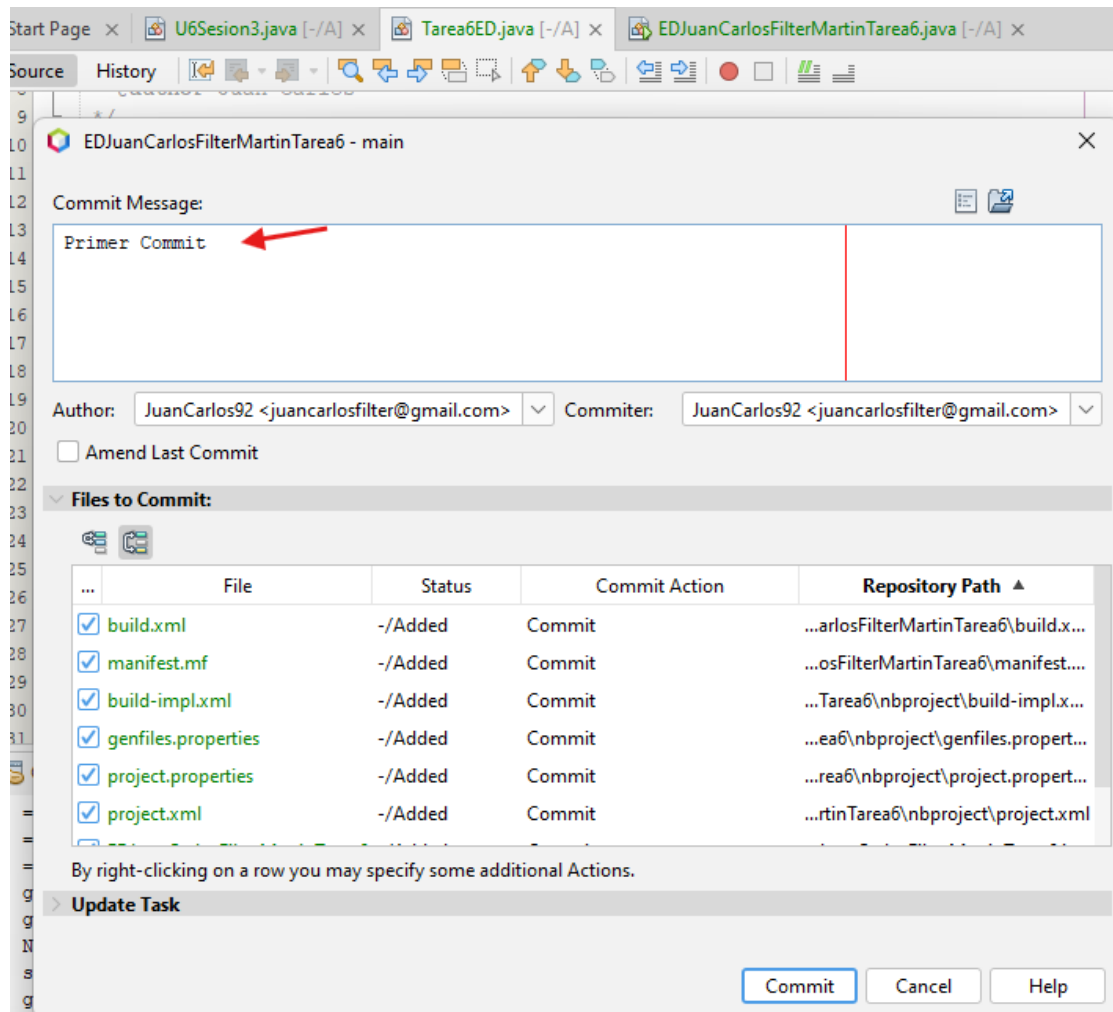
× Realizar Commit.

Antes de llevar el proyecto al repositorio con push previamente hay que hacer un commit (Volcado de los archivos del proyecto al repositorio git)

Botón derecho en el proyecto > Git > **Commit**

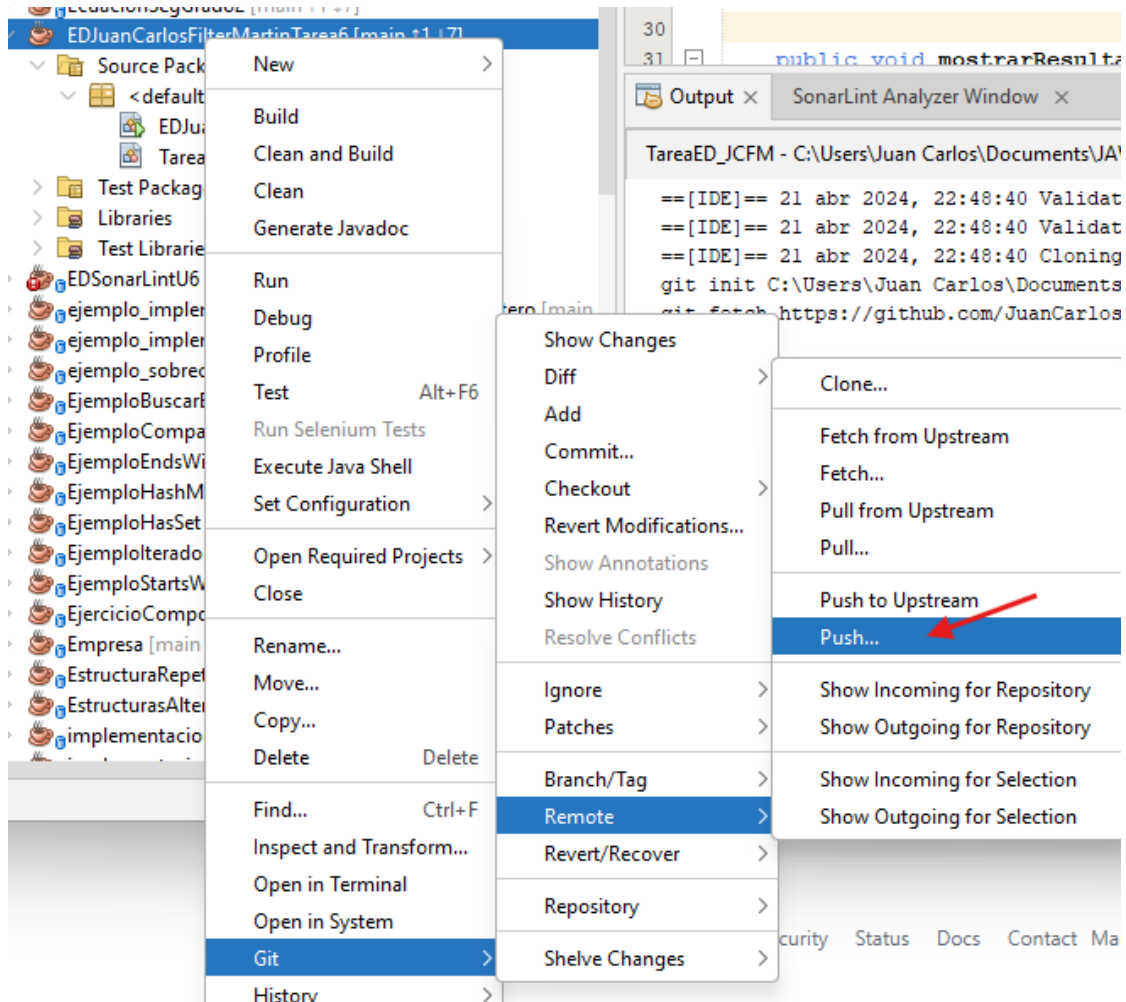


Indicamos un mensaje (Obligatorio) para hacer referencia a la versión que se va a almacenar en el repositorio local y los archivos que se van a almacenar en él.

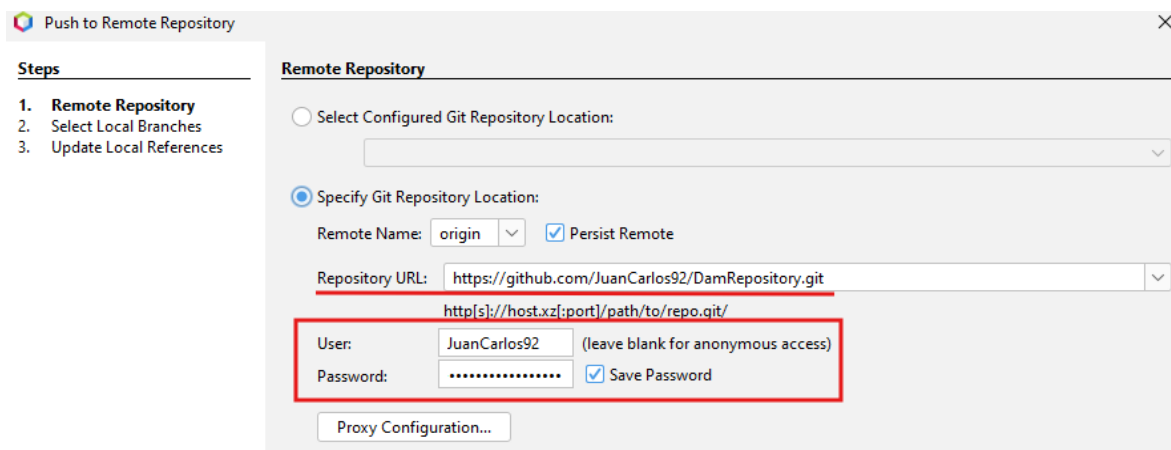


Una vez realizado el commit se va a hacer un push del proyecto, que consiste en mandarlo (subirlo) al repositorio.

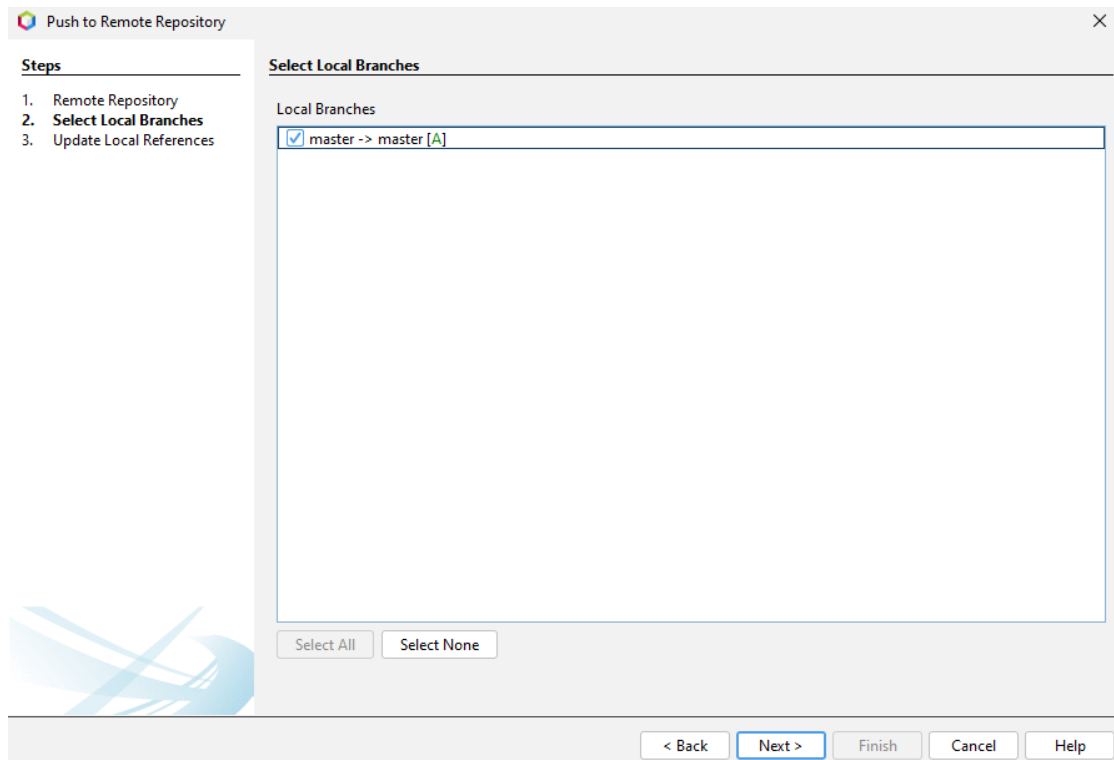
Botón derecho sobre el proyecto > Git > Remote > **Push**



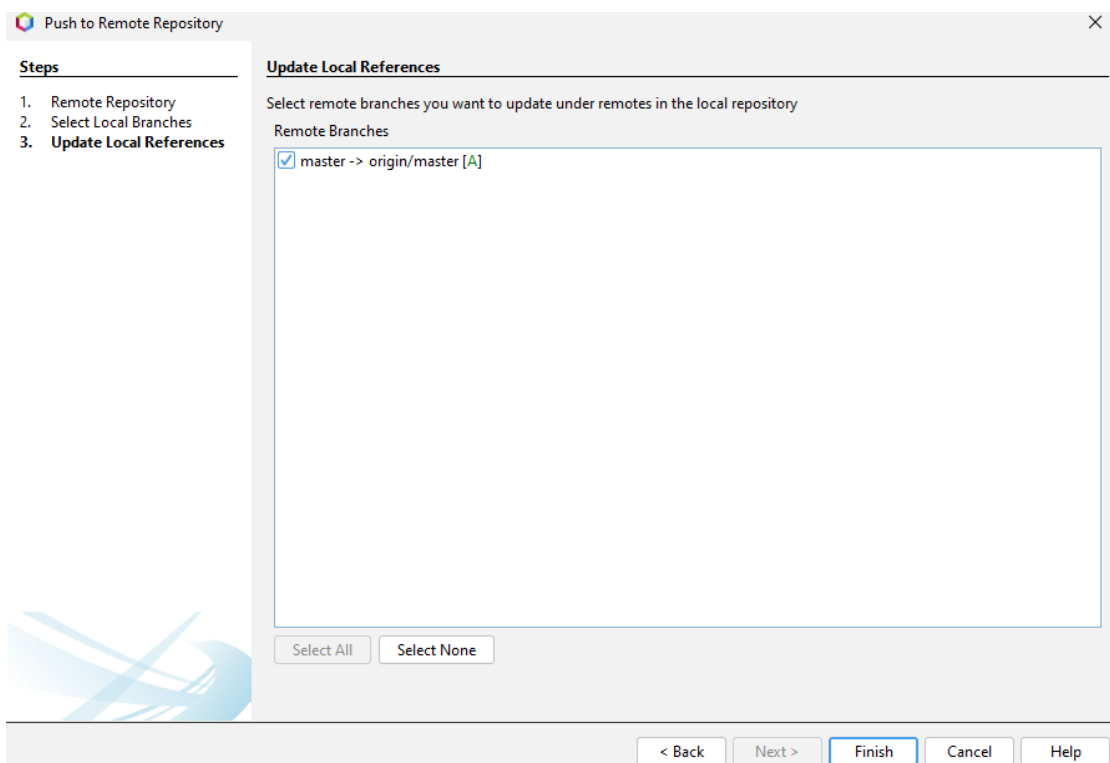
Indicamos el enlace de github donde se encuentra el repositorio, el usuario y la contraseña que es el token generado



También Indicamos la rama (Branch) que queremos enviar al repositorio remoto.



Y por último como se va a actualizar las referencias locales

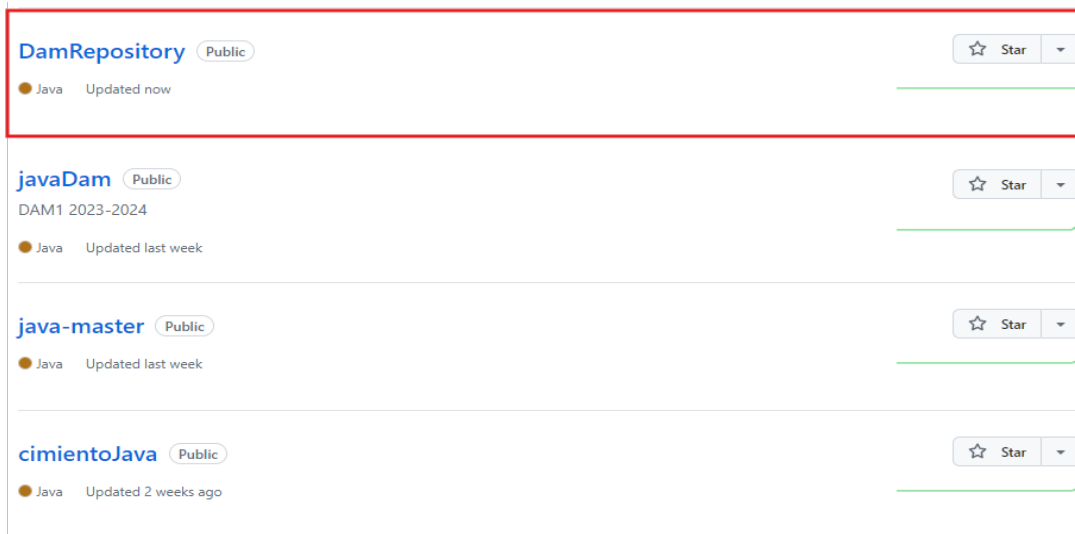


Con esto tendríamos el proyecto en el repositorio de GitHub

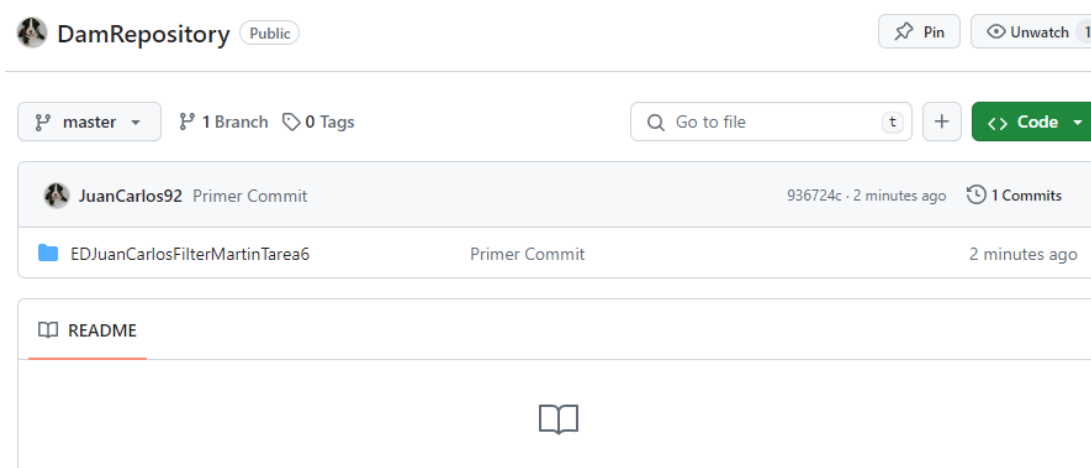
(En caso de que quisiéramos traer el proyecto del repositorio git a nuestro directorio local tendríamos que hacer un pull)

× Mostrar el código fuente a través de la pagina de web de github.

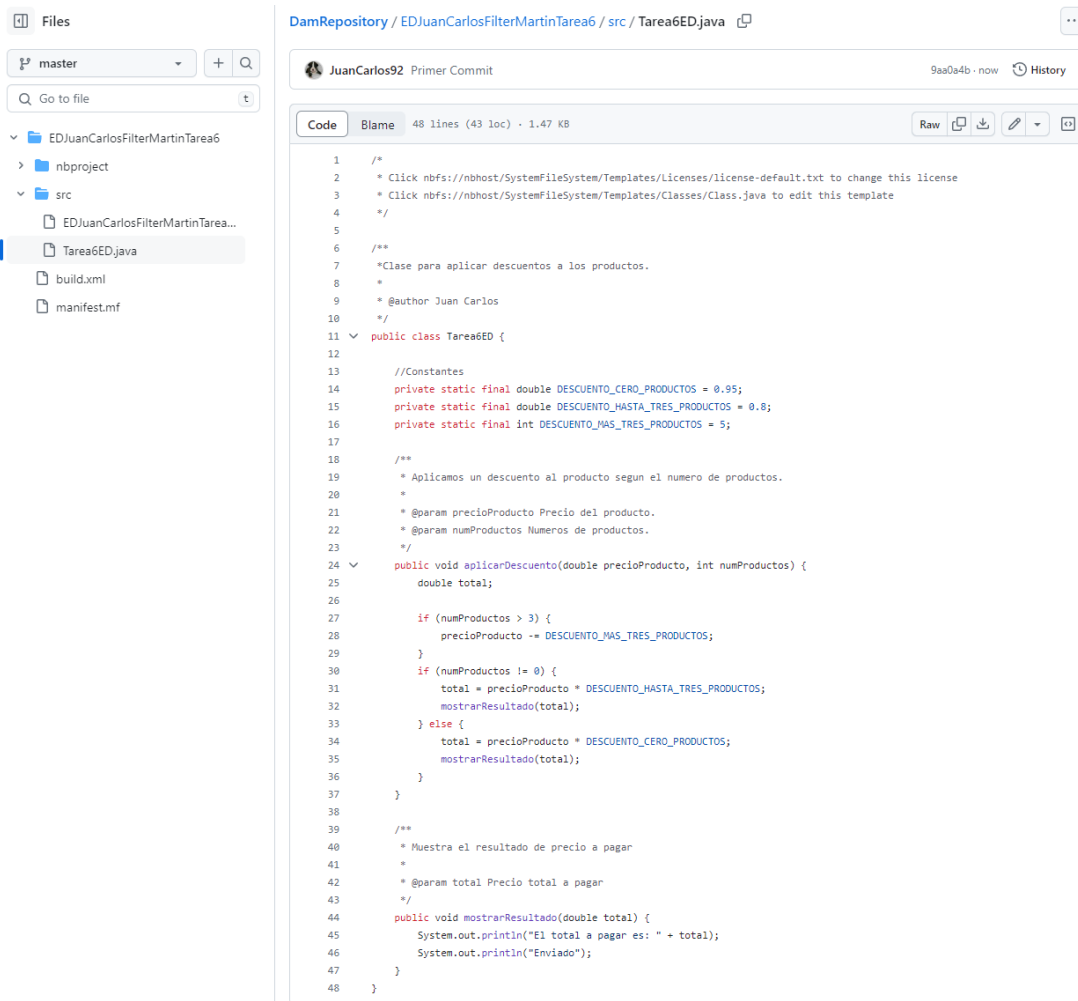
Nos dirigimos a la página web de GitHub y en repositorio entramos en el repositorio donde se encuentre el proyecto.



Si entramos en el podemos ver como ahora aparece el proyecto



Entrando en el se puede comprobar que tenemos las clases con sus respectivos códigos.



The image shows a file explorer on the left and a code editor on the right. The file explorer displays the directory structure of a repository, including files like build.xml, manifest.mf, and the Tarea6ED.java file. The code editor shows the content of Tarea6ED.java, which is a Java class for applying discounts to products. The code includes comments in Spanish and defines constants for discount rates and thresholds. It also contains methods for applying the discount and displaying the result.

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5
6  /**
7   * Clase para aplicar descuentos a los productos.
8   *
9   * @author Juan Carlos
10  */
11  public class Tarea6ED {
12
13      //Constantes
14      private static final double DESCUENTO_CERO_PRODUCTOS = 0.95;
15      private static final double DESCUENTO_HASTA_TRES_PRODUCTOS = 0.8;
16      private static final int DESCUENTO_MAS_TRES_PRODUCTOS = 5;
17
18      /**
19       * Aplicamos un descuento al producto segun el numero de productos.
20       *
21       * @param precioProducto Precio del producto.
22       * @param numProductos Numeros de productos.
23       */
24      public void aplicarDescuento(double precioProducto, int numProductos) {
25          double total;
26
27          if (numProductos > 3) {
28              precioProducto -= DESCUENTO_MAS_TRES_PRODUCTOS;
29          }
30          if (numProductos != 0) {
31              total = precioProducto * DESCUENTO_HASTA_TRES_PRODUCTOS;
32              mostrarResultado(total);
33          } else {
34              total = precioProducto * DESCUENTO_CERO_PRODUCTOS;
35              mostrarResultado(total);
36          }
37      }
38
39      /**
40       * Muestra el resultado de precio a pagar
41       *
42       * @param total Precio total a pagar
43       */
44      public void mostrarResultado(double total) {
45          System.out.println("El total a pagar es: " + total);
46          System.out.println("Enviado");
47      }
48  }
```

Cabe comentar que si queremos obtener este código desde la propia web de GitHub podemos obtenerlo mediante un ZIP o abrirlo mediante la GitHub Desktop

