

# Manejo básico de Git:

## Comandos y Netbeans. Práctica IV

### Tareas

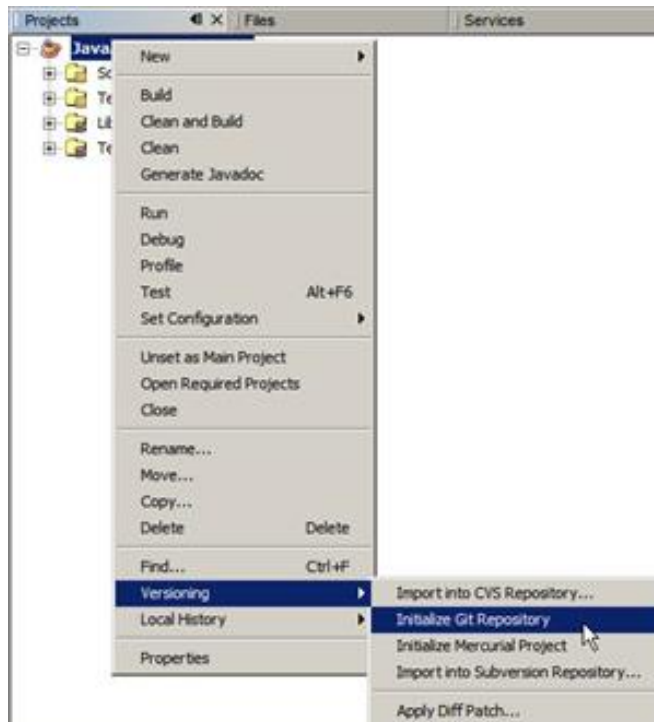
#### 1. Iniciar Repositorio

**Cmd:**

>git init

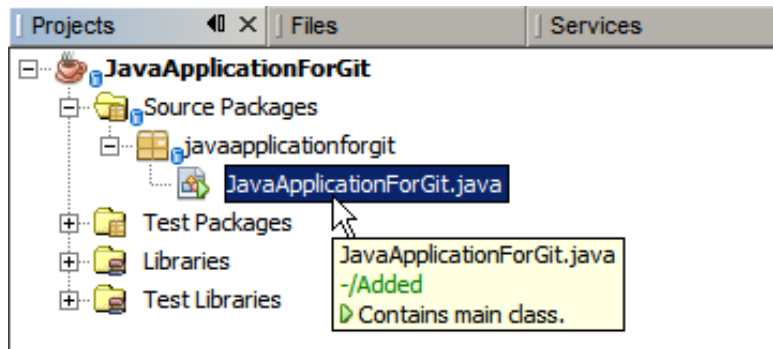
**Netbeans:**

Sobre mi proyecto: <Botón Derecho> Versioning > Initialize Git Repository  
(alternativamente , en el Menú: Team > Git > Initialize)



Aparecerá una pequeña ventana indicando el lugar donde se creará el repositorio, por defecto en la dirección del proyecto actual (y en la carpeta oculta .git). La dejamos como está y presionamos OK.

- Una vez iniciado el repositorio GIT aparece un pequeño **cilindro de color azul** en el proyecto.
- Todos los archivos de proyecto están marcados como agregados (add)
- Para ver el estado de un archivo, coloca el cursor sobre el nombre del archivo en la ventana Proyectos, como se muestra en la siguiente imagen:



## 2. Desconectar de Git

### Cmd:

>ninguna orden en particular

### Netbeans:

Menu: Team>Disconnect

Atención: si no desconecto, sigue activo aunque cierre NetBeans y lo vuelva a abrir

## 3. Indicar a Git que archivos queremos añadir al *commit*

### Cmd

>git add lista.txt

### Netbeans:

Sobre mi proyecto: <Botón Derecho> Git > add

(Alternativamente Menu: Team>Add)

## 4. Realizar un *commit*

### Cmd:

>git commit -m "comentario"

> git commit -a -m "comentarios diversos"

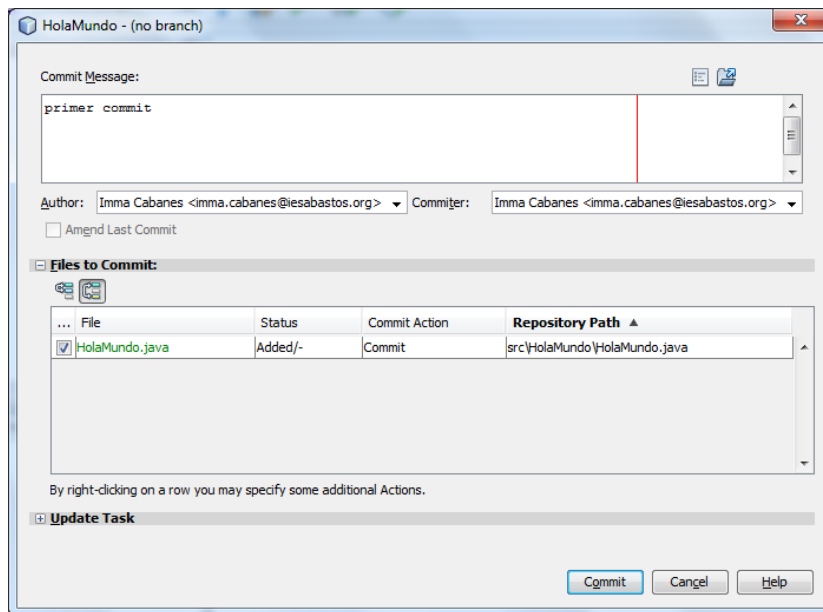
(La opción -a permite hacer el commit sin pasar por el área de preparación o index)

### Netbeans:



Sobre mi proyecto (o bien puedo situarme sobre el paquete o fichero que quiera commit) <Botón Derecho> Git > Commit

(Alternativamente Menu: Team > Commit)

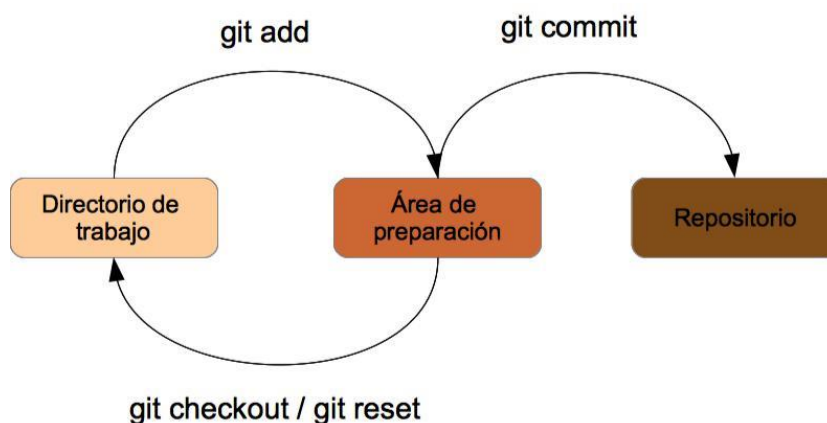
- Y se abre la ventana:



Atención!!! **Opciones importantes:**

- Changes between HEAD and Index (  )
- Changes between HEAD and Working Tree (  )

>> Observa las diferencias entre seleccionar una u otra opción



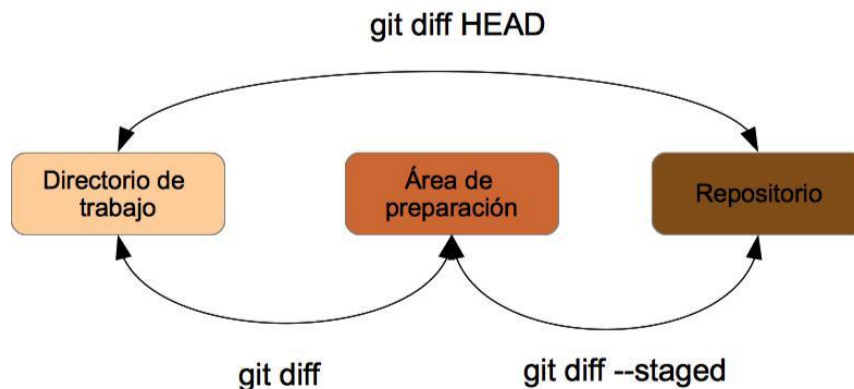
Recuerda: Un fichero en Git puede tener uno de estos 3 estados:

- **confirmado:** fichero cuyos datos ya están almacenados en el repositorio
- **preparado (staged):** fichero que ha sido marcado para que en el próximo commit pase al repositorio.
- **modificado:** fichero que ha sido modificado en el directorio de trabajo, pero que no será incluido en el próximo commit.

## 5.- Diferencias entre versiones:

### Cmd:

>git diff



### Netbeans:

- a) Cuando editemos nuestros archivos podremos observar que aparecen líneas de colores en el lado izquierdo del editor de código:

**Azul**  ) Indica líneas que han sido cambiadas desde la revisión anterior.

**Verde**  ) Indica las líneas que se han añadido desde la revisión anterior.

**Rojo**  ) Indica líneas que se han eliminado desde la revisión anterior.

El margen izquierdo del Editor muestra los cambios que ocurren línea por línea. Al modificar una línea dada, los cambios se muestran inmediatamente en el margen izquierdo:



Si acercamos el puntero del mouse a estas líneas podremos obtener más información acerca de los cambios

- b) Si hacemos clic en estas líneas, aparece un pequeño menú con unos comandos para navegar más fácilmente por el editor, permite también eliminar los cambios realizados y abrir la ventana DIFF, en donde se puede observar mucho mejor el antes y el después de los cambios realizados.

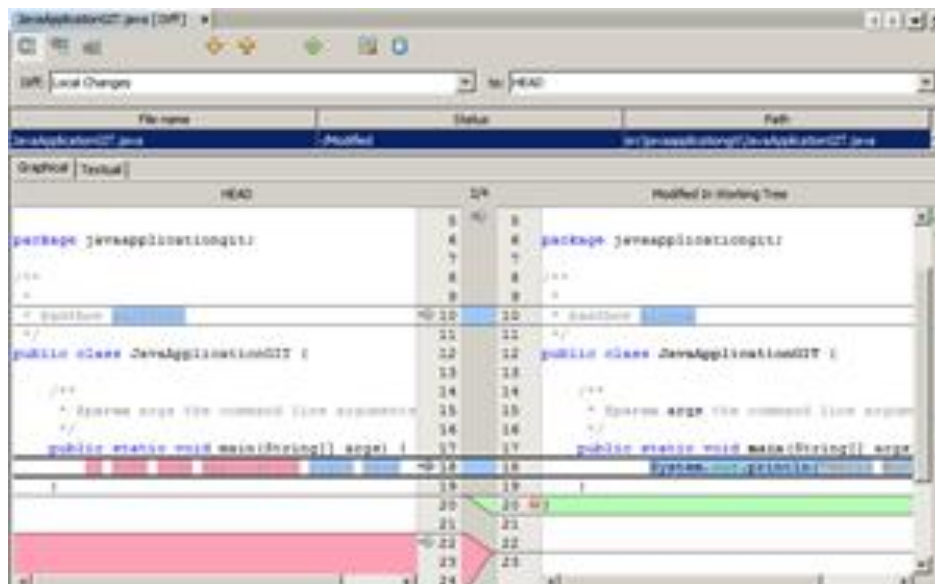


- c) El margen derecho del editor de código proporciona una vista general que muestra los cambios realizados en su archivo como un todo, de arriba a abajo. La codificación de color se genera inmediatamente cuando se realizan cambios en el archivo.

```
public String getDescription() {
    // This description will be displayed in the dialog,
    // hard-coded = ugly, should be done via I18N
    return "Text documents (*.txt)";
}
```



- d) Otra forma de abrir la ventana DIFF, es clic derecho sobre el archivo que queremos explorar, GIT -> DIF



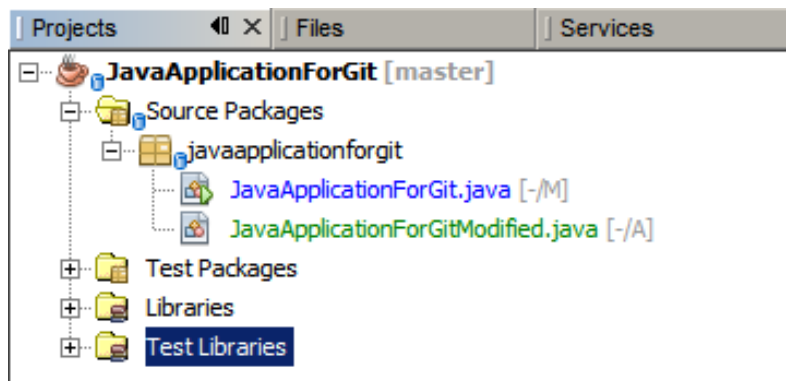
## 5. Ver el estado en el que se encuentra Git

### Cmd:



> git status

### Netbeans:






Muestra de forma visual de la información del estado del archivo



Codificación de colores utilizado para los elementos del proyecto:

Componente	Descripción
<b>cilindro azul</b> 	Indica la presencia de archivos que se han modificado, agregado o eliminado en su árbol de trabajo
<b>cilindro rojo</b> 	Marca proyectos, carpetas o paquetes que contienen <i>archivos</i> conflictivos.

Codificación de colores aplicada a los nombres de archivo para indicar su estado actual en el repositorio:

Color	Ejemplo	Descripción
<b>Sin color específico (negro)</b>	 Main.java	Indica que el archivo no tiene cambios.
<b>Azul</b>	 Main.java	Indica que el archivo se ha modificado localmente.
<b>Verde</b>	 Main.java	Indica que el archivo se ha añadido localmente.
<b>rojo</b>	 Main.java	Indica que el archivo está en un conflicto de mezcla (merge).
<b>Gris</b>	 Main.java	Indica que el archivo es ignorado por Git

Etiquetas de estado del archivo:

Se muestran dos valores de estado para un archivo:




- Un estado que describe las diferencias del archivo entre el área Índice y el commit HEAD actual.
- Un estado que describe las diferencias del archivo entre el directorio de trabajo y el área índice

Las posibles etiquetas son:

Etiqueta de estado	Significado
-	Sin modificar
A	Añadido
U	Actualizado pero no mezclado
M	Modificado
D	Eliminado
I	Ignorado
R	Renombrado

Por ejemplo:

```

 JavaApplicationForGit.java [M/-]
 JavaApplicationForGitModified.java [-/A]
 JavaApplicationToDelete.java
  
```

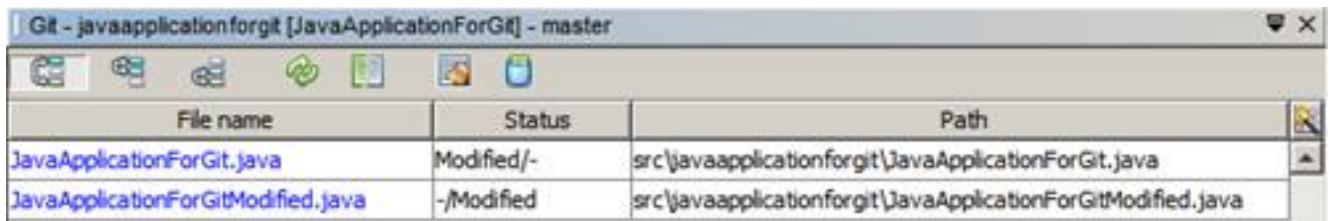
Se muestra en el formato [1erEstado/2doEstado]

>>Observa y discute las diferencias entre [-/Modified] y [Modified/-]

### Netbeans: Show Changes. Otra herramienta interesante








Sobre mi proyecto: <Botón Derecho> Git > Show Changes  
(alternativamente Menu: Team > Show changes)

Abre una lista en tiempo real de todos los cambios realizados en los archivos dentro de una carpeta seleccionada del árbol de trabajo local.



File name	Status	Path
JavaApplicationForGit.java	Modified/-	src\javaapplicationforgit\JavaApplicationForGit.java
JavaApplicationForGitModified.java	-/Modified	src\javaapplicationforgit\JavaApplicationForGitModified.java

Incluye botones que le permiten invocar las tareas más comunes de Git en todos los archivos mostrados en la lista. En la tabla siguiente se enumeran los comandos Git disponibles en la barra de herramientas de Show Changes:

Icono	Nombre	Función
	<b>Changes between HEAD and Working Tree</b>	Muestra una lista de archivos que ya están preparados (área índice) o solo se han modificado / creado (área de trabajo).
	<b>Changes between HEAD and Index</b>	Muestra una lista de archivos que están en preparados.
	<b>Cambios entre el índice y el Área de trabajo</b>	Muestra los archivos que tienen diferencias entre el área índice y el área de trabajo.
	<b>Refresh Statuses</b>	Actualiza el estado de los archivos y carpetas seleccionados. (Los archivos que se muestran en la vista se pueden actualizar para reflejar cualquier cambio que se haya realizado externamente)
	<b>Open Diff</b>	Abre el Visor de Diff proporcionándole una comparativa entre las copias locales y las versiones mantenidas en el repositorio.
	<b>Revert Modifications</b>	Muestra el cuadro de diálogo para deshacer modificaciones.
	<b>Commit Changes</b>	Muestra el cuadro de diálogo Commit.

## 6. Crear etiquetas

### Cmd:

>git tag -a v1 -m "mi primer tag" ab029

### Netbeans:

Sobre mi proyecto: <Botón Derecho> Git > Branch/Tag >Create Tag Repository (alternativamente , en el Menú: Team > > Branch/Tag >Create Tag)

## 7. Deshacer cambios

Para desechar los cambios locales realizados en los archivos seleccionados en su árbol de trabajo y reemplazarlos con los del índice o HEAD:

### Cmd:

> git checkout HEAD (Sustituir por el último commit )

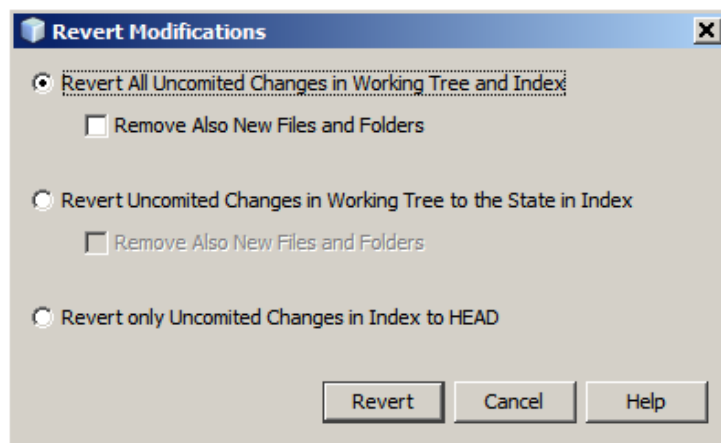
>git checkout -- lista.txt (Sustituir el fichero lista.txt que hay en el directorio de trabajo por el que hay en el área de preparación)

>git reset HEAD lista.txt (Sustituir el fichero lista.txt del área de preparación por el del último commit)

### Netbeans:

Sobre mi proyecto: <Botón Derecho> Git > Revert/Recover (Alternativamente, en el Menú: Team > Revert/Recover)





## 8. ¿Cómo puedo volver a un commit anterior

Tener un histórico de versiones y no poder volver a ellas si fuera necesario no tendría mucho sentido. Para conseguir cargar en el directorio de trabajo un *commit* determinado

### Cmd:

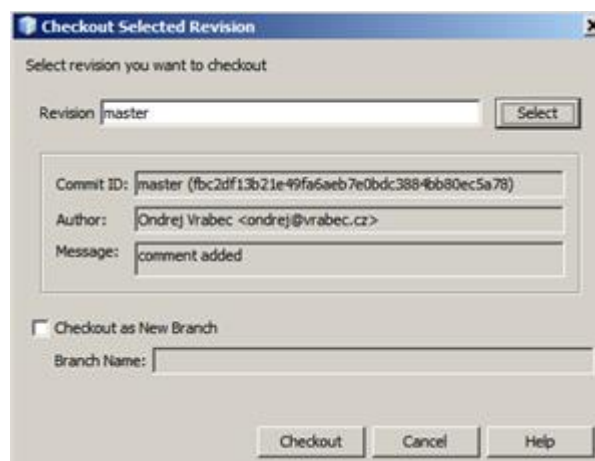
- > git checkout HEAD~1 (volver al penúltimo commit)
- > git checkout afd342a1 (volver a un determinado commit)
- > git checkout master (Volver al último commit)

### NetBeans:

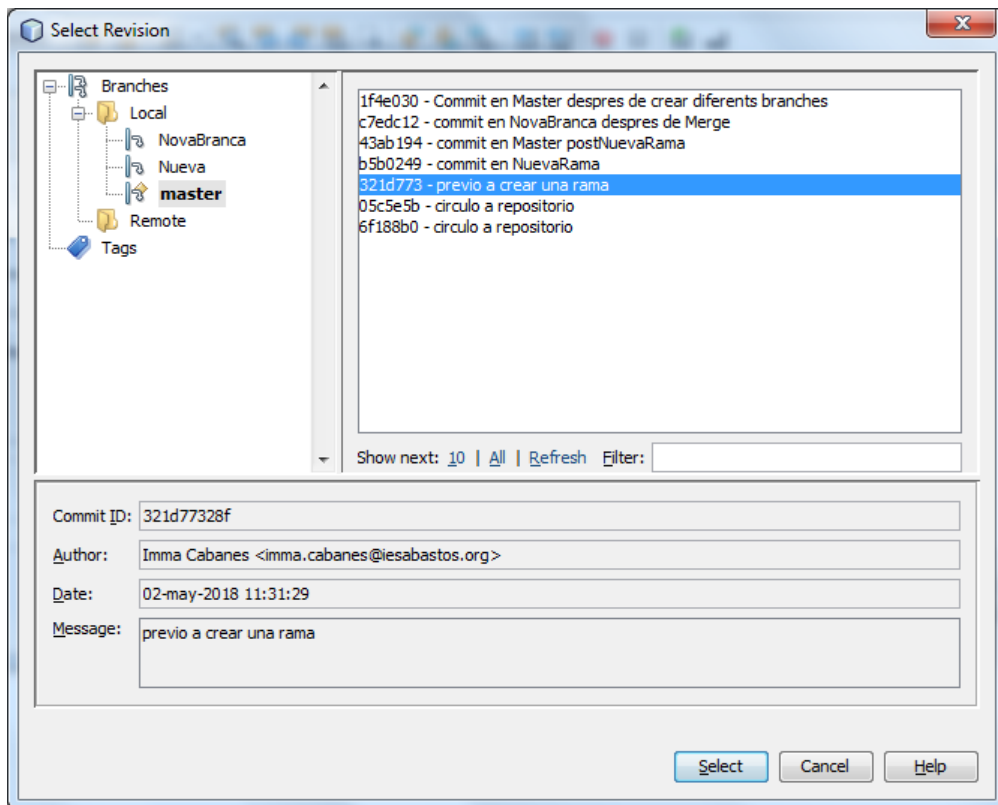
Llama Revision a cada una de los commits

En el menú: Team > Checkout > Checkout Revision

Aparecerá el cuadro de diálogo Checkout Selected Revision :

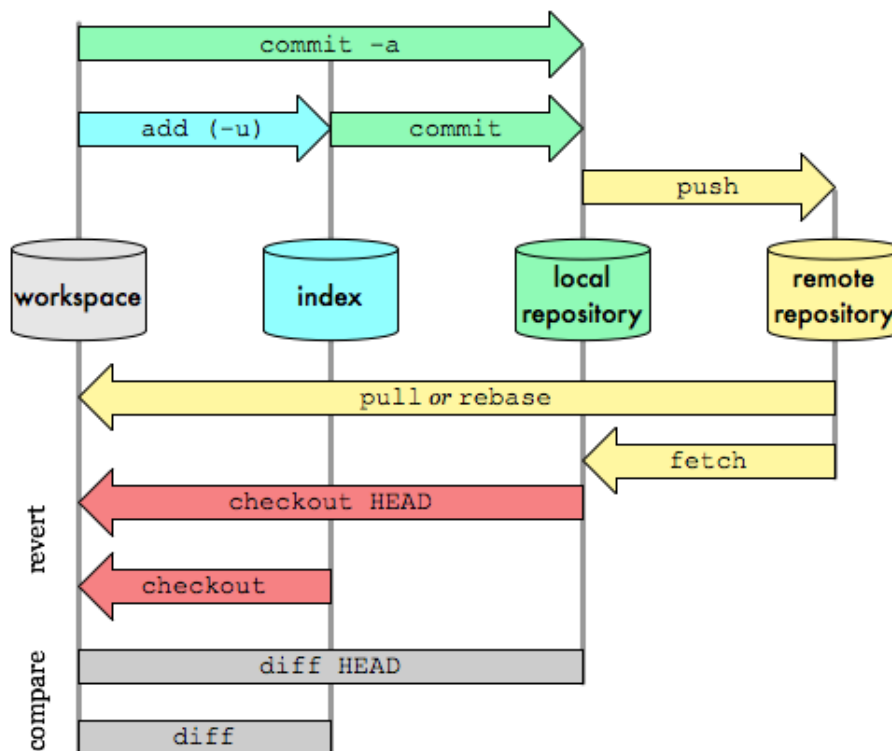


Pulsando Select podremos elegir la Revisión:



## Git Data Transport Commands

<http://osteele.com>



## 9. Trabajando con Ramas

Git nos permite mantener diferentes versiones de un proyecto usando ramas. Esto puede ser interesante, por ejemplo, si deseamos trabajar en una versión diferenciada del sistema de archivos para estabilizar o experimentar sin alterar la rama principal

### a) Crear una rama local

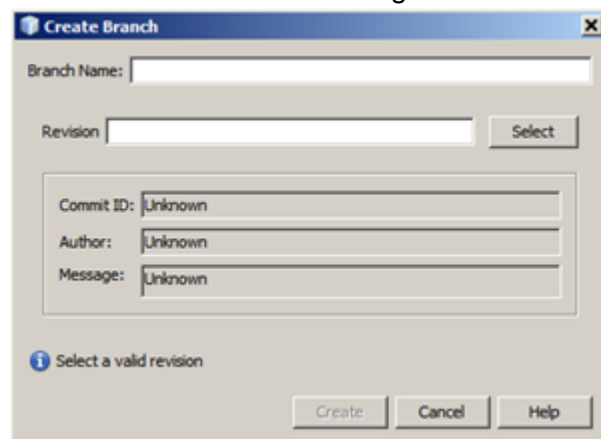
#### Cmd:

>git branch nombreDeLaRama

#### Netbeans:

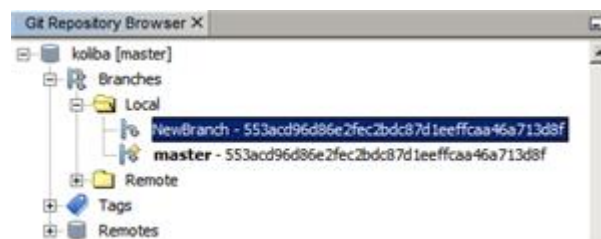
Sobre mi proyecto: <Botón Derecho> Git > Branch/Tag >Create Branch  
(alternativamente , en el Menú: Team > > Branch/Tag >Create Branch)

Se muestra el cuadro de diálogo Crear rama.



Indicaremos:

- El nombre de la rama que se está creando.
- Revisión: El commit o etiqueta donde se inicia la rama. Presionando Select vemos una lista de revisiones mantenidas en el repositorio:



(Opcional, puedo ponerme a trabajar –Checkout- en la rama al crearla)

## b) Checkout

### Cmd:

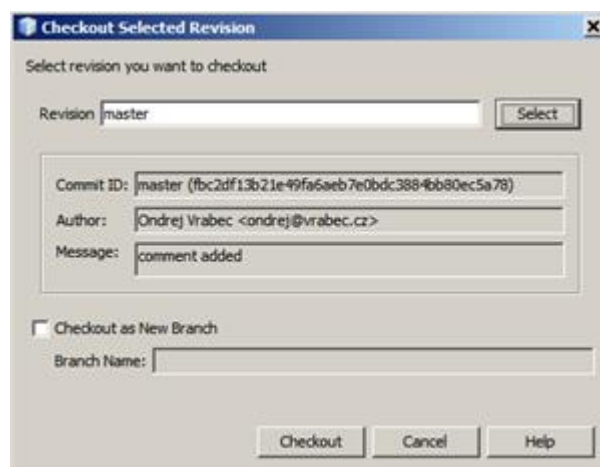
>git checkout nombreDeLaRama

### Netbeans:

Para cambiar de rama y seguir trabajando (editar, crear, etc. sus archivos) en una rama existente, podemos volcar los archivos de la rama al Área de Trabajo (Checkout).

En el menú: Team > Checkout > Checkout Revision

Aparecerá el cuadro de diálogo Checkout Selected Revision :



Como antes, pulsando Select podremos elegir la Revisión

## c) Merge o Fusión

### Cmd:

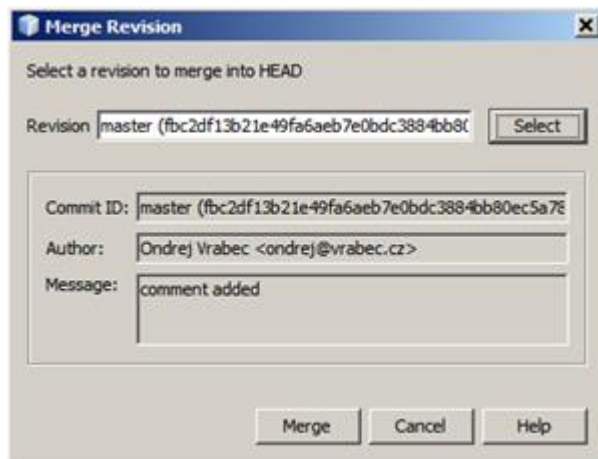
desdeLaRamaDondeFusiono> git merge nombreDeLaRamaAFusiar

### Netbeans:

Para fusionar en el Área de Trabajo una revisión del repositorio. Es decir, fusionar dos ramas

En el Menú principal Team> Branch / Tag> Merge Revision.

Aparecerá el cuadro de diálogo

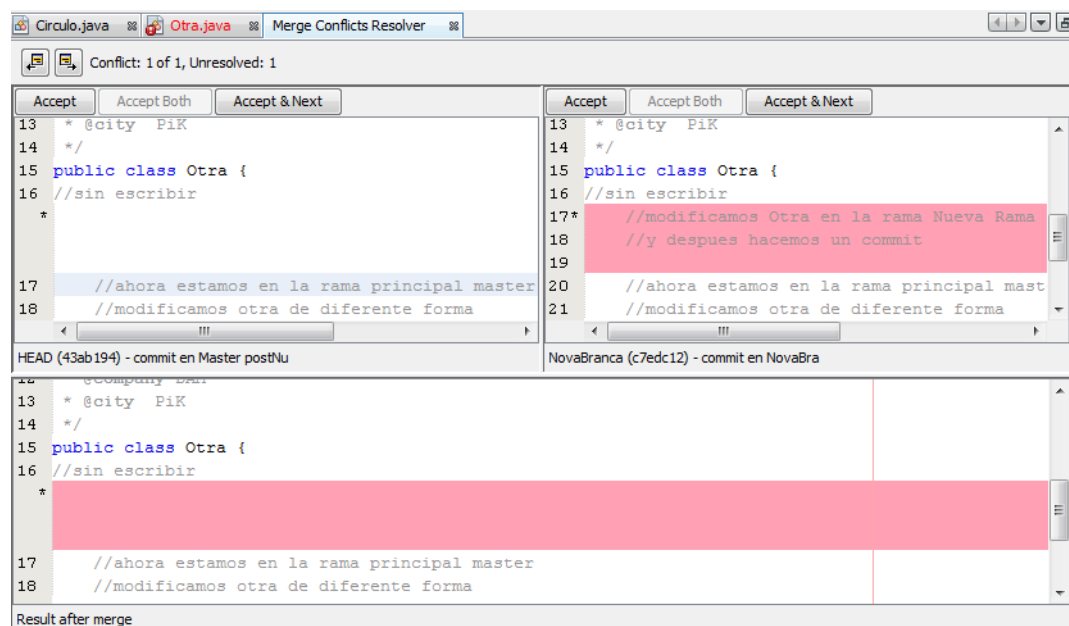
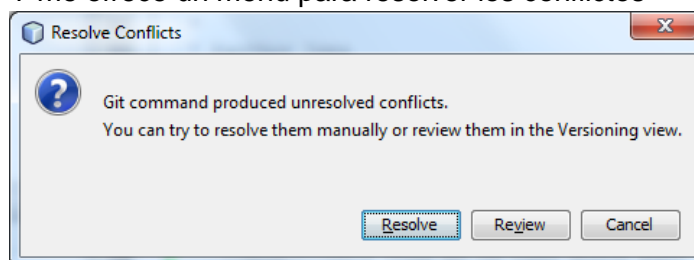


Especificamos la revisión requerida presionando Seleccionar para ver la lista de revisiones mantenidas en el repositorio (o introducimos la ID de un commit, o una rama existente o un nombre de etiqueta)

Se realiza una combinación a tres bandas entre la rama actual, el contenido del Árbol de Trabajo y la rama especificada.

- Si se produce un conflicto de fusión, el archivo en conflicto se marcará con una insignia roja

Y me ofrece un menú para resolver los conflictos



- Después de la fusión, aún debe confirmar los cambios para que se puedan agregar al HEAD.

#### **d) Eliminar una rama**

##### **Cmd:**

```
>git branch -d nombreDeLaRama  
>git branch -D nombreDeLaRama
```

##### **Netbeans:**

Para eliminar una rama local innecesaria:

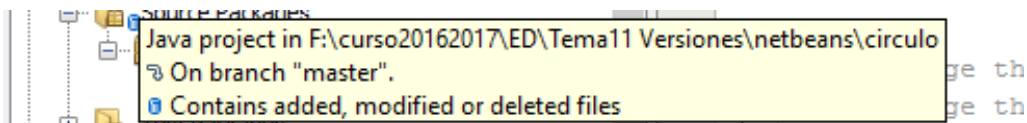
En el menu principal Team > Repository >Repository Browser

## Ejercicios guiados Pasos a Pasos:

1. Crea un nuevo proyecto o reutiliza el que tenías con la clase Circulo.
2. Inicializa git para este proyecto
3. Añade todos los archivos (observa los indicadores gráficos del estado de git)
4. Haz un commit (observa el estado)
5. Modifica la clase Circulo añadiéndole la documentación de la clase:

```
/**
 * @author xxxx
 * @version 10/5/2017
 */
```

- Observa cómo se marcan en verde las líneas añadidas
- Sitúate sobre el proyecto y mira la información del git.

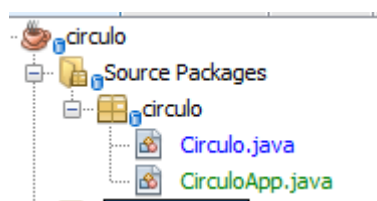


6. Ahora vamos a añadirlo al index: sitúate sobre el archivo y en el menú contextual selecciona git->add
7. Modifica la fecha de la versión

Observa las diferencias entre:

	HEAD y working tree
	HEAD e Index
	Index y working tree

8. Añade una clase CirculoApp.java

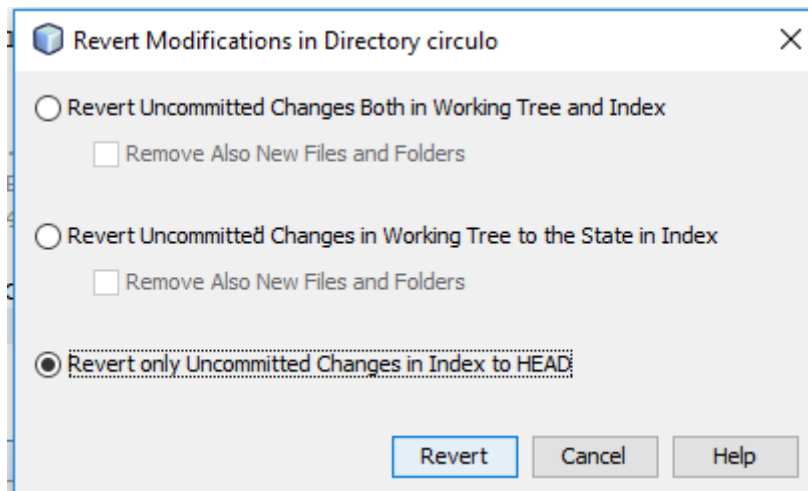


¿Qué indica cada color?

9. Muestra los cambios con git→show changes

File name	Status	Path
Circulo.java	Modified/Modified	src\circulo\Circulo.java
CirculoApp.java	-/Added	...irculo\CirculoApp.java

10. Si seleccionamos la opción de revertir cambios, nos muestra la siguiente ventana:



Qué opción sería equivalente a:

git reset HEAD circulo.java: .....

git checkout circulo.java: .....

11. Añade CirculoApp al index

12. Realiza un commit (Mensaje: Añadida aplicación)

**Ahora SIGUE TÚ**

13. Crea una rama “rectangulo”

14. Cambia a esta rama

15. Añade una clase Rectangulo (sólo el constructor)



```
public class Rectangulo {  
    float alto, ancho;  
    public Rectangulo(float l1, float l2)    {  
        alto=l1;  
        ancho=l2;  
    }  
}
```

16. Añádela al index y realiza un commit
17. Cambia a la rama master y observa su contenido
18. Muévete entre las 2 ramas y mira cómo cambia el contenido del proyecto.
19. Sitúate en la rama master y haz un fast-forward para fusionar la rama rectángulo