Práctica GIT 2

1.-Introducción

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

2.-Enunciado

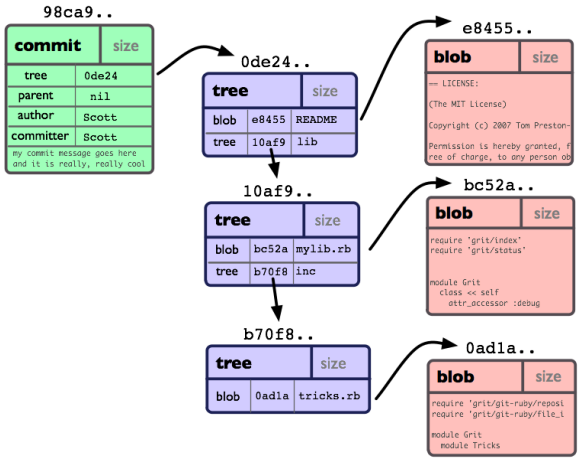
Para realizar estos ejercicios es necesario que hagas uso de la cuenta de GitHub que has creado en clase. Por cada ejercicio es necesario que crees un repositorio en tu cuenta.

2.1.- Individual

2.1.1.- Ejercicio 1

Para entender cómo funciona GIT es necesario saber cómo lo hace por dentro. Para ello se necesita conocer estos términos: **SHA, blob, tree, commit, tag**.

Define estos términos conociendo la información que alberga cada uno y como se relacionan entre ellos para hacer funcionar un repositorio. Posteriormente explica lo que ve en esta imagen detallando información.



Para facilitar la búsqueda de información se sugiere utilizar el siguiente enlace: http://shafiul.github.io/gitbook/1_the_git_object_model.html

2.1.2.- Repositorio remoto

Lo ideal para realizar este ejercicio, es jugar con dos usuarios para cómo se notifican los cambios. Podéis decirle a un compañero que os cree un repositorio remoto y os pase el enlace, paso 1).

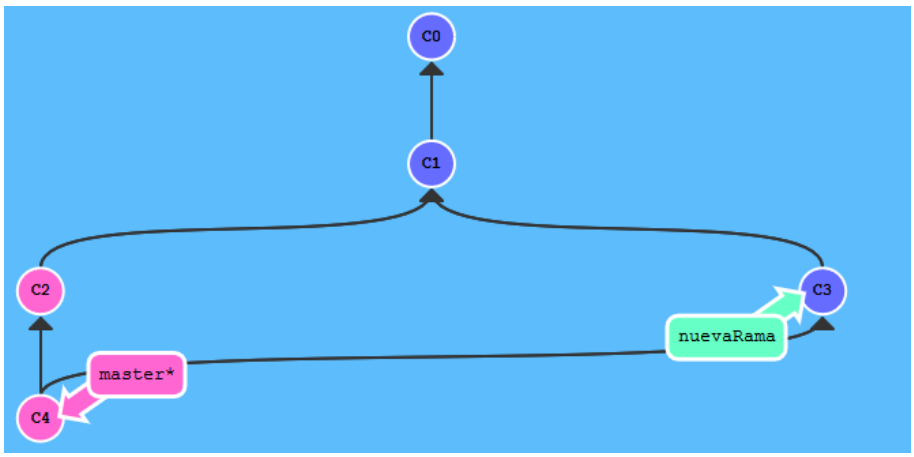
1. Colaborar en el repositorio remoto, crea uno para ello.
2. Clónalo su repositorio.
3. Añadir varios ficheros con algún contenido.
4. Añadir los cambios a la zona de intercambio temporal.
5. Hacer un commit con el mensaje "subida ejercicio avanzado."
6. Subir los cambios al repositorio remoto.
7. Haz una bifurcación del repositorio remoto anterior.
8. Clonar el repositorio de nuevo.
9. Crear una nueva rama y activarla.
10. Añade varios ficheros con contenido.
11. Añadir los cambios a la zona de intercambio temporal.
12. Hacer un commit con el mensaje "cambios en la nueva rama."
13. Subir los cambios de la rama creada en el punto 9 al repositorio remoto.
14. Hacer un *Pull Request* de los cambios en la rama.

2.1.3.- Programación en GIT

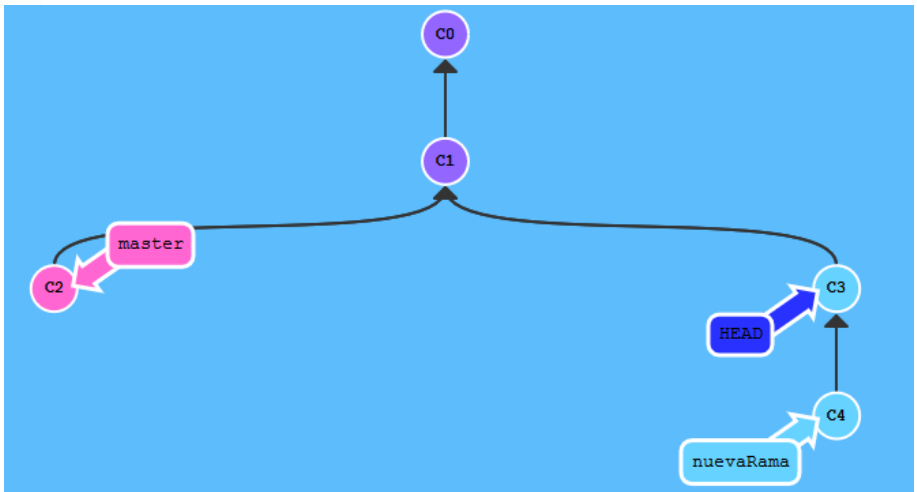
Existen múltiples herramientas de aprendizaje de GIT, en este ejercicio, vamos a utilizar learninggit. Esta herramienta nos va a permitir profundizar en los comandos que hemos visto en clase: clone (para hacer commit en repositorio remoto tienes que hacer git fakeTeamwork cantidad_commits), merge, fetch, push, pull, branch,rebase , reset, revert, …

En este ejercicio tienes que indicar la serie de pasos a seguir para obtener los resultados de la imagen en cuestión:

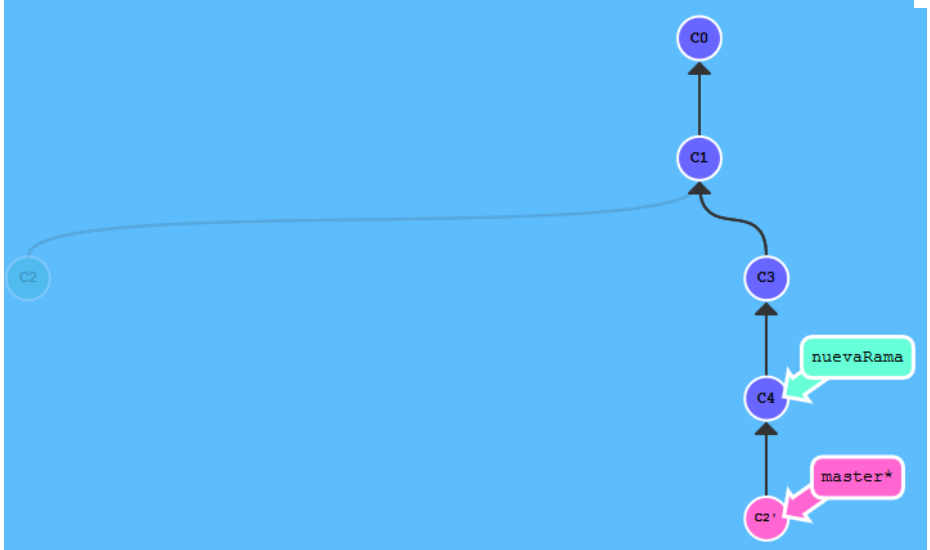
1)



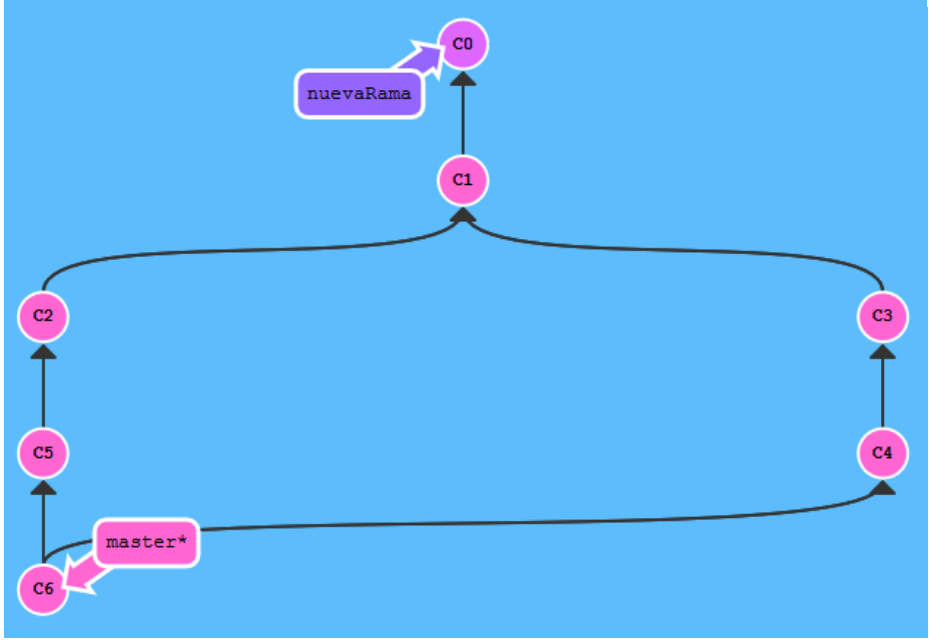
2)


C0
C1
master
C2
C3
HEAD
C4
nuevaRama

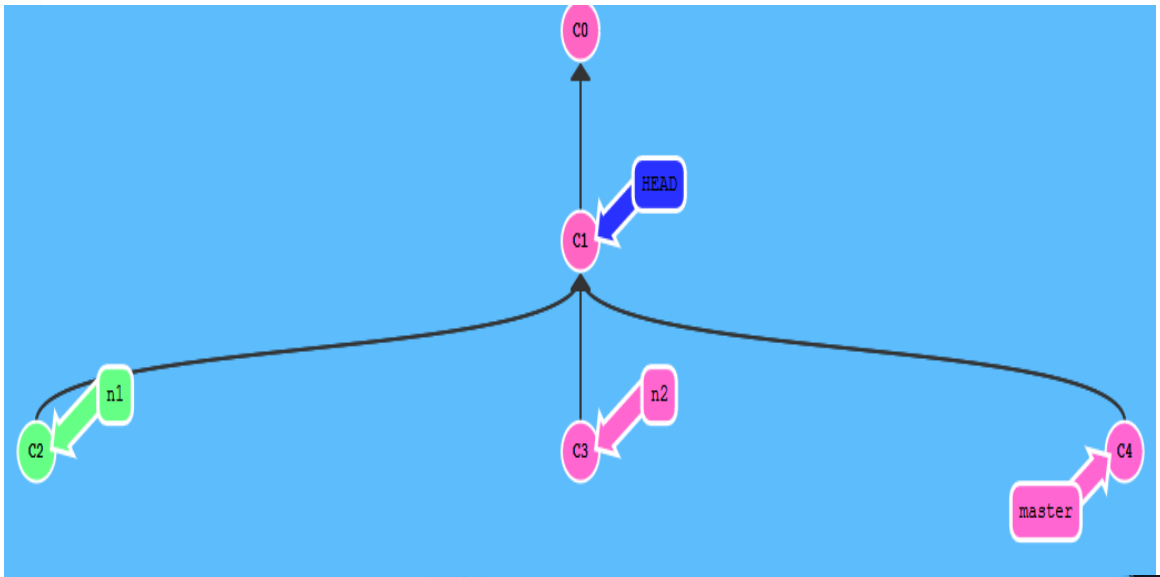
3)


C0
C1
C2
C3
nuevaRama
C4
master*
C2'

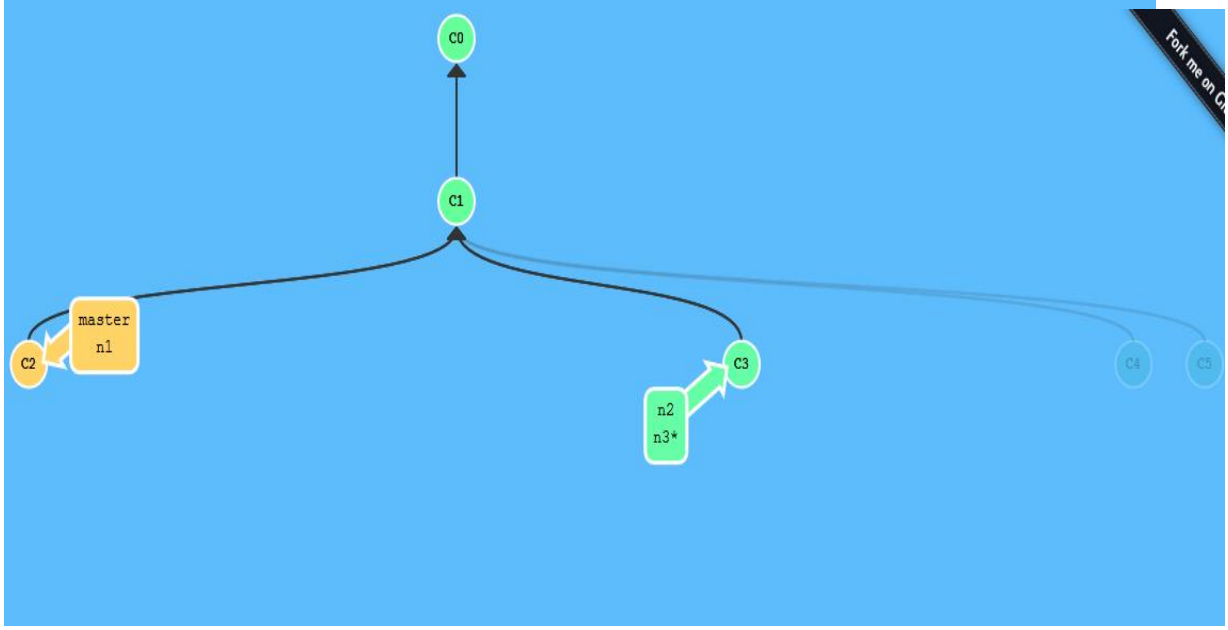
4)


C0
nuevaRama
C1
C2
C3
C5
C4
master*
C6

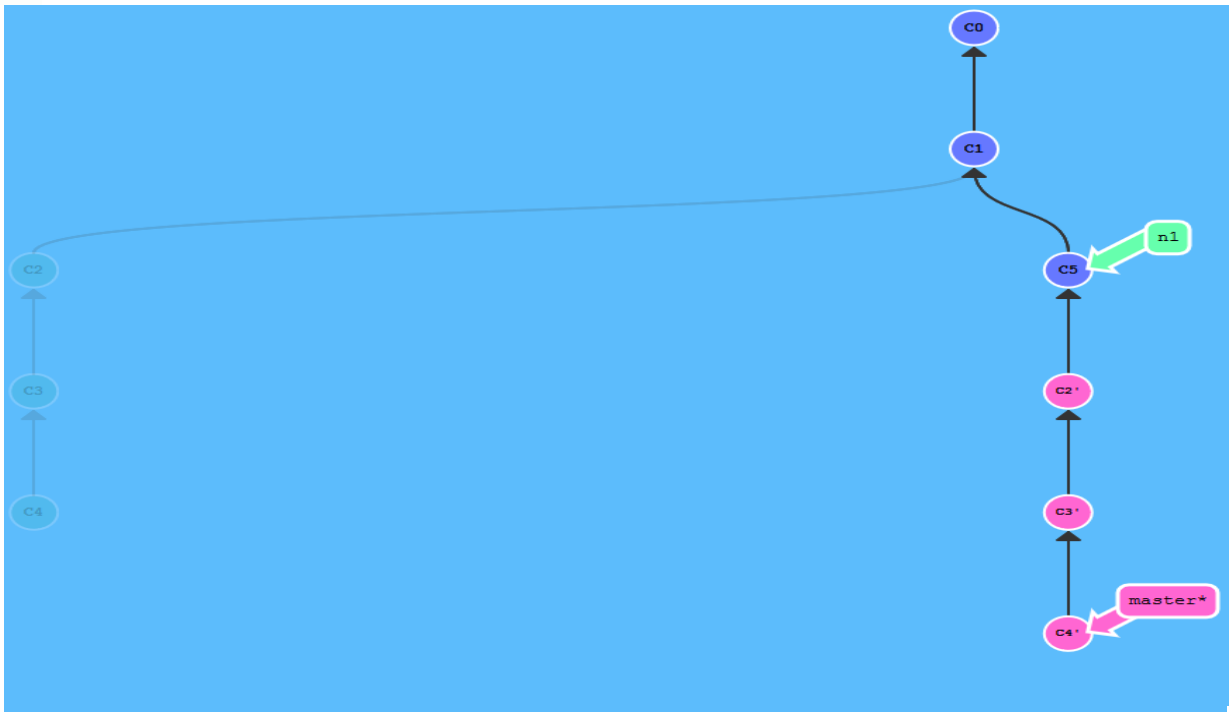
5)



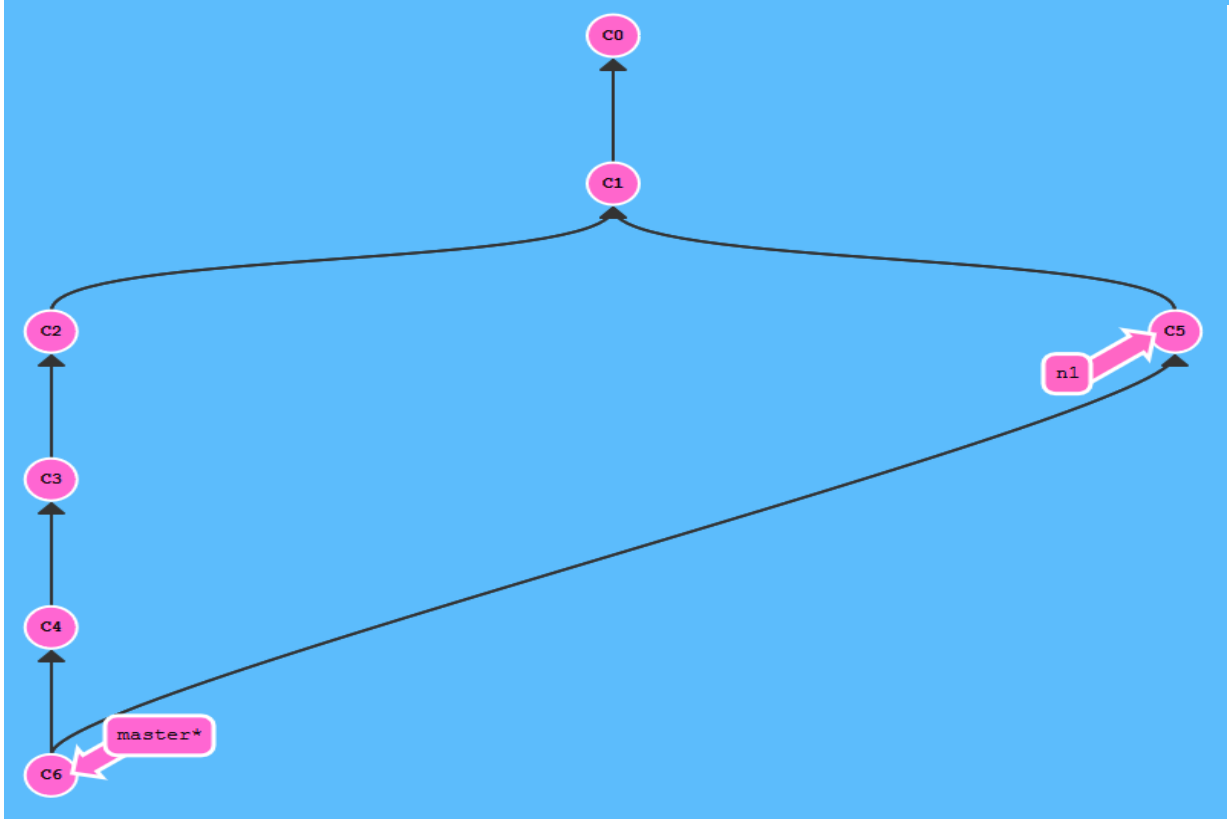
6)



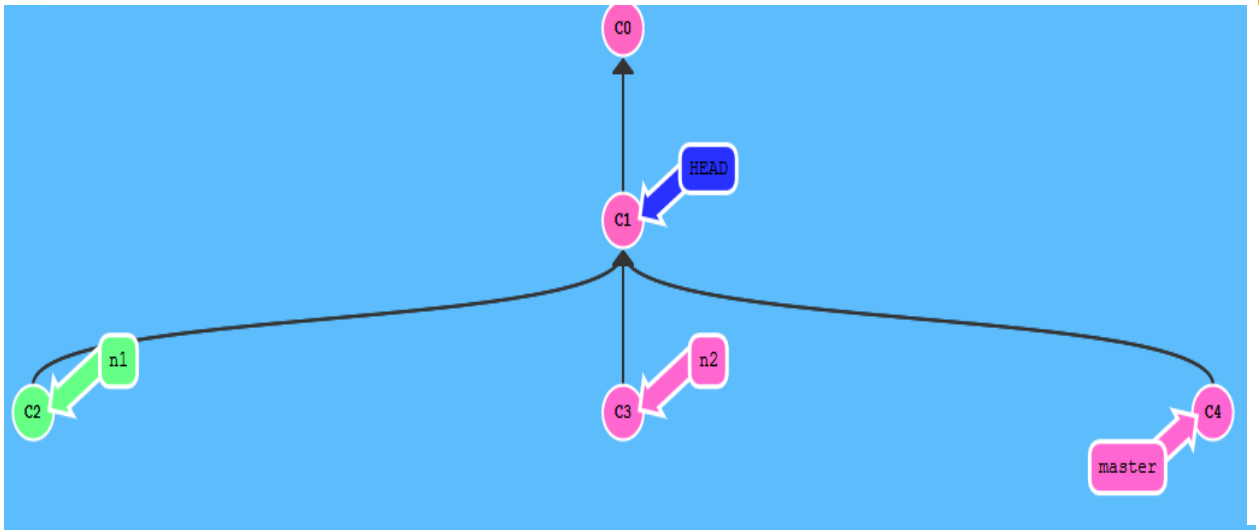
7)



8)



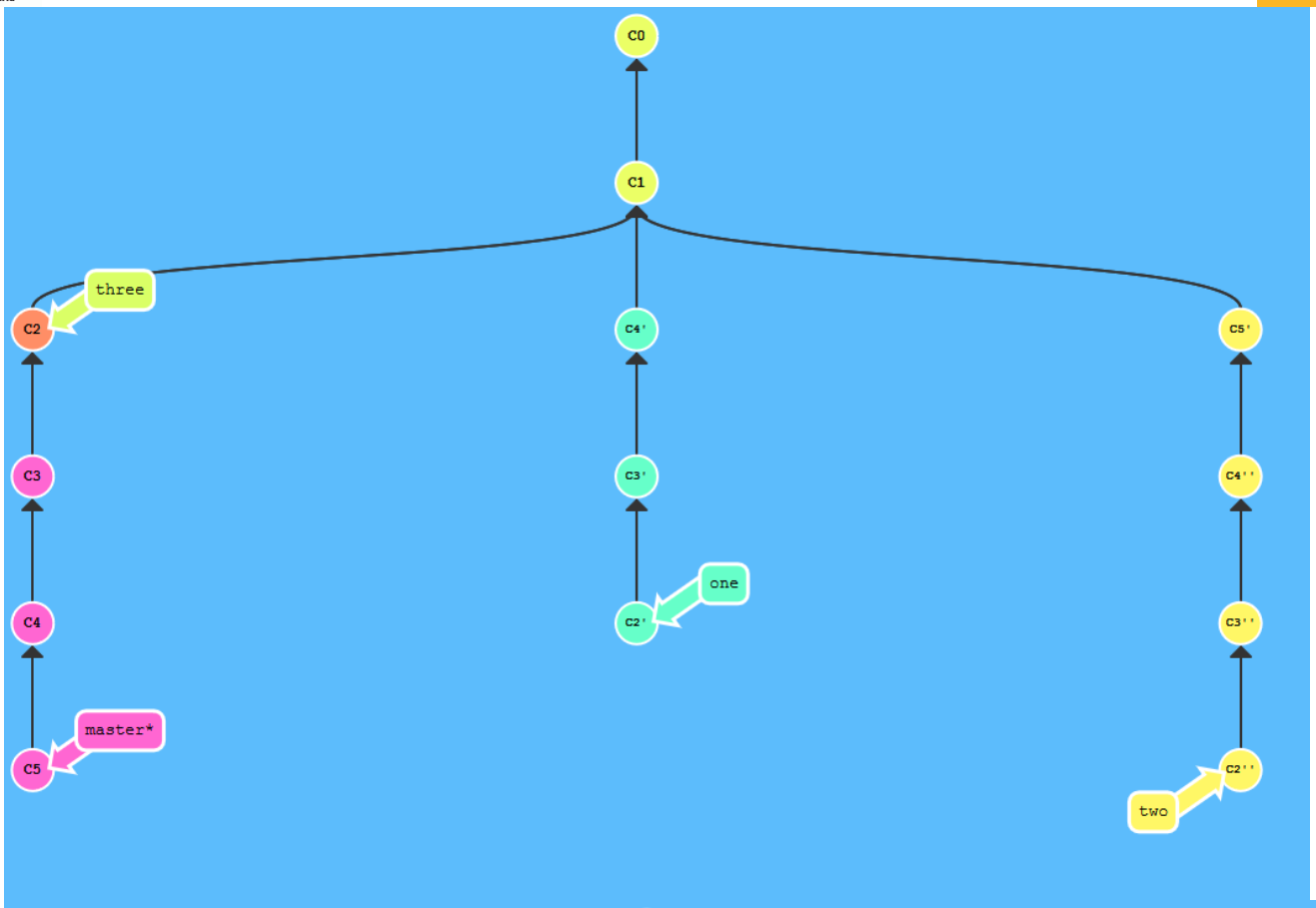
9)


C0
HEAD
C1
n1
C2
n2
C3
C4
master

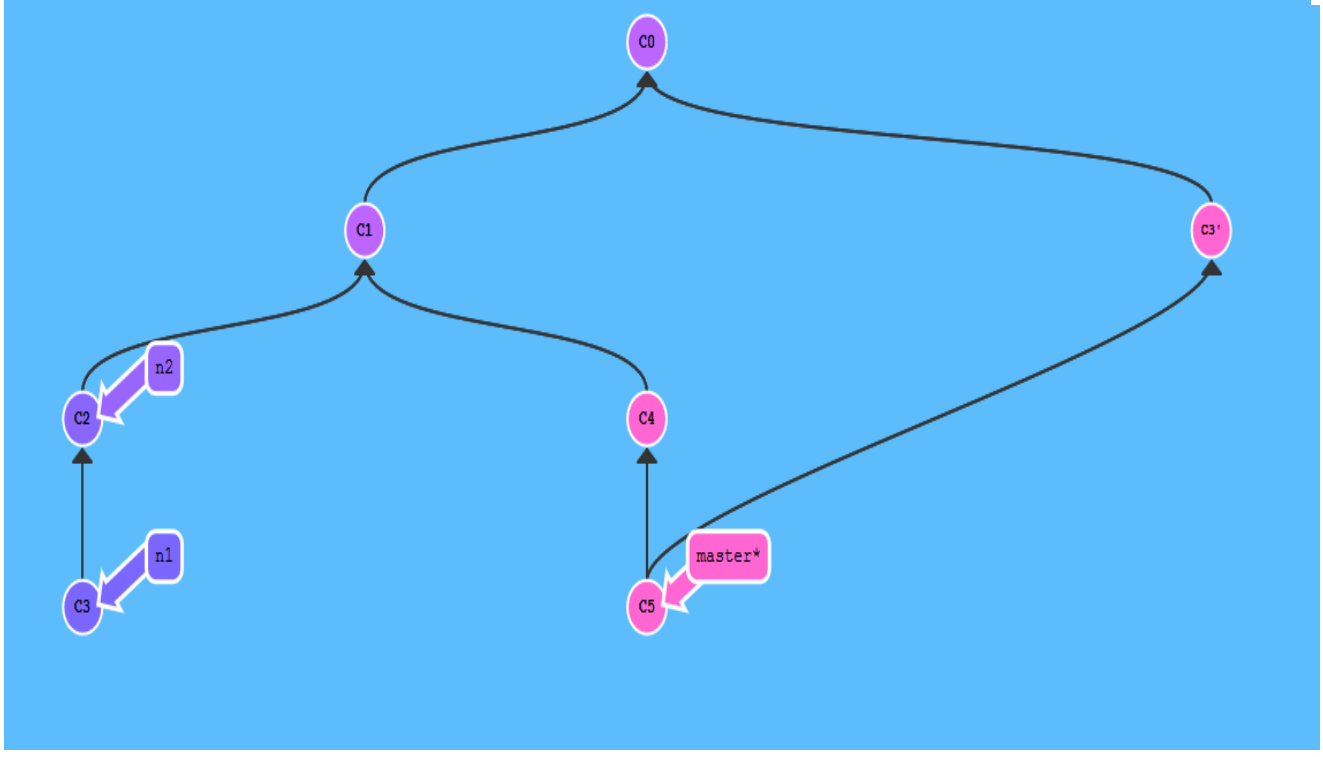
10)


C0
master*
o/master
C1
C0
master
C1

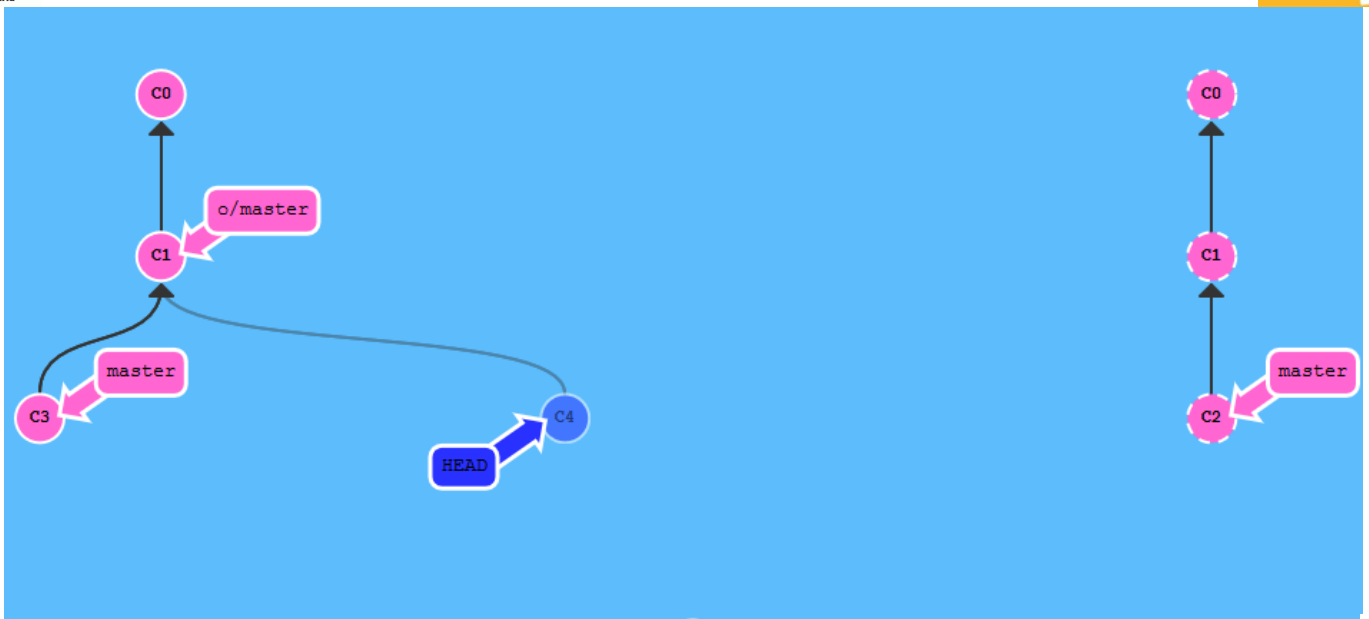
11)


C0
C1
three
C2
C4'
C5'
C3
C3'
C4''
one
C4
C2'
C3''
master*
C5
C2''
two

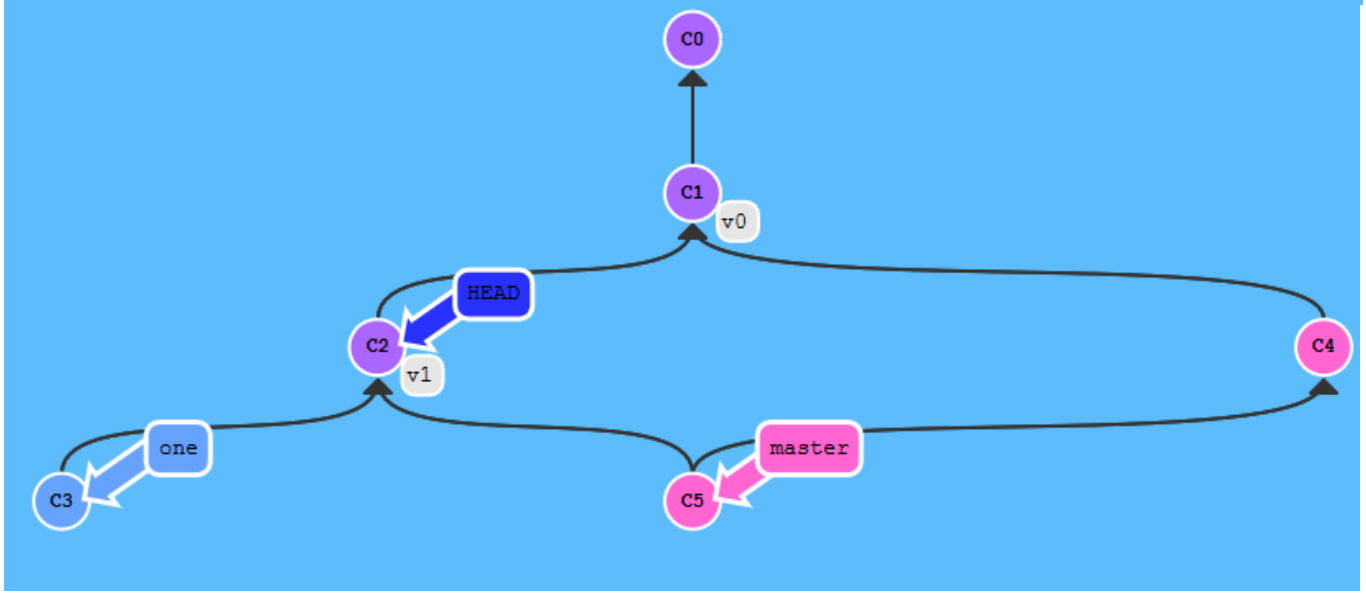
12)


C0
C1
C3'
n2
C2
C4
n1
C3
master*
C5

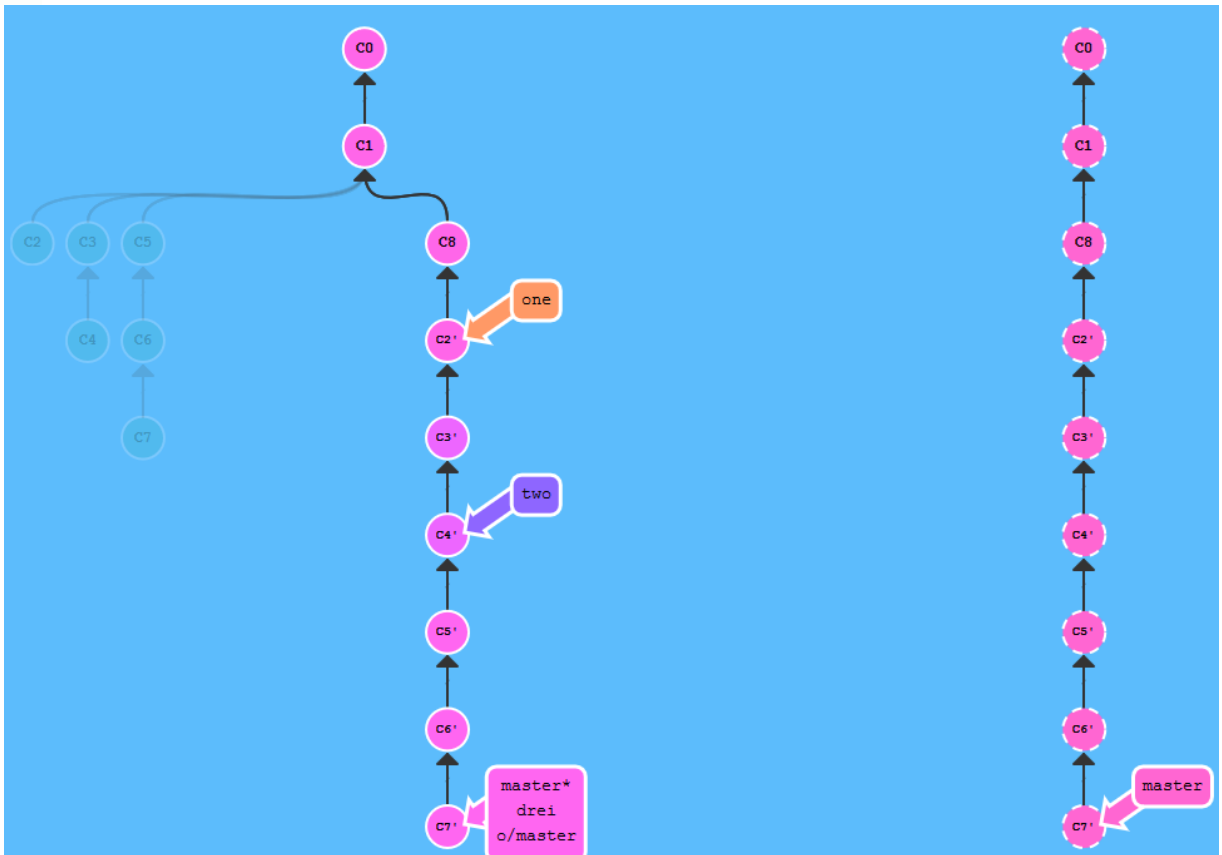
13)


C0
o/master
C1
master
C3
C4
HEAD
C0
C1
master
C2

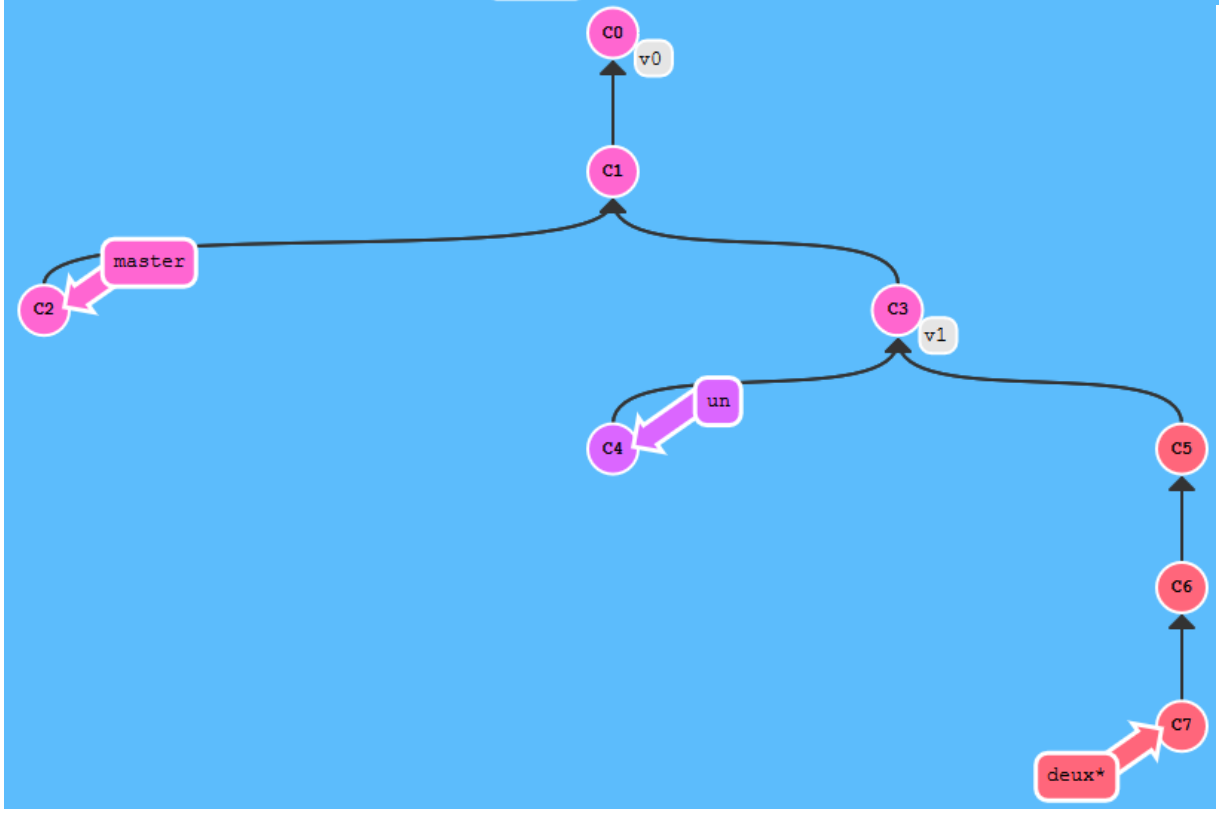
14)


C0
C1
v0
HEAD
C2
v1
C4
one
C3
master
C5

15)


C0
C1
C2
C3
C5
C8
C4
C6
C7
one
C2'
C3'
two
C4'
C5'
C6'
master*
drei
o/master
C7'
C0
C1
C8
C2'
C3'
C4'
C5'
C6'
master
C7'

16)


C0
v0
C1
master
C2
C3
v1
un
C4
C5
C6
C7
deux*

2.2.- Grupal

2.2.1.- Ejercicio 1

Para este ejercicio es necesario montar un repositorio conjunto. Es necesario hacerlo en grupos de 3. Lo primero es por ello en GitHub montar un equipo de trabajo de 3 personas y crear un repositorio. Todas ellas tienes que tener todos los poderes sobre el repositorio.

Cada integrante deberá seguir los pasos de los ficheros que se dejan a continuación. Finalmente detalla todos los pasos realizados por cada miembro.

Alumno A:

1. Crea un proyecto Java con nombre facturas y este súbelo al repositorio remoto a la rama Master.
2. Crea una rama “alumno A” para trabajar tu.
3. Se te ha encargado crear una interfaz que será implementada por todas las facturas.
4. Crea un interfaz llamado IFactura.java, con tres métodos:

   void cambiarEstado(Estados estado);
   float getTotal();
   float getCantidadIva();

5. Crea una clase enumerado llamado Estados.java, que se llame Estados y que tenga como valores Aprobada, Pendiente y Cobrada.
6. Crea un commit para añadir el enum.
7. Crea un commit para añadir la interfaz.
8. Vuelca el trabajo a la rama Master y borra la rama que has usado.
9. Cuando tengas en tu repositorio la clase Factura que ha implementado tu compañero, crea una rama para corregir un bug referente a la propiedad estado de la clase Factura.
10. Corrige el bug: cambia la visibilidad de public a private. Haz un commit con esta corrección.
11. Vuelca la modificación a la rama Master y borra la rama que has usado.
12. Cuando esté lista la primera puesta en producción mediante la rama master, bájatela a tu repositorio local.

Alumno B

1. Se te ha encargado empezar el esqueleto de la clase Factura. Crea una rama para ello llamada “Alumno B” a partir de la versión actualizada de Master.
2. Crea un archivo llamado Factura.java para la clase Factura, con los siguientes miembros además del método main():

   public int num;
   public float base;
   public float tipoIva;
   public Estados estado;
3. Comenta la línea de la propiedad "estado", porque aún no está implementado el enum Estados.

4. Crea un commit con la clase.
5. Vuelca el trabajo a la rama Master y borra la rama que has usado.
6. Cuando tengas la interfaz y el enum (desarrollada por otro compañero) en tu repositorio local, crea una rama para aplicar los siguientes cambios.
7. Descomenta la línea que comentaste en el punto 5 y haz un commit con esta modificación.
8. Implementa la interfaz en tu clase Factura (no hace falta que implementes los métodos, déjalos en blanco).
9. Crea un commit con este último cambio.
10. Corrige un bug: cambia la visibilidad de la propiedad estado a protected. Haz un commit con esta corrección. Si más adelante hay un conflicto con algún compañero del equipo, ésta es la visibilidad que tiene que quedarse definitivamente.
11. Vuelca el trabajo a la rama Master y borra la rama que has usado.
12. Eres el encargado de preparar la primera versión que irá a producción:
13. Añade un tag al último commit de master con la versión 0.1.0.

Alumno C

1. Se te ha encargado implementar algunos métodos de la clase Factura y un enum de estados. Empieza creando una rama llamada “Alumno C” para el enum a partir de la versión actualizada de Master.
2. Crea un archivo llamado Estados.java para un enum, que se llame Estados y que tenga como valores Aprobada, Pendiente, Cobrada y Anulada.
3. Si más adelante aparece este mismo enum en la rama de algún compañero del equipo y hay conflictos, éstos son los cuatro valores definitivos que tienen que quedar.
4. Crea un commit para añadir el enum.
5. Vuelca el trabajo a la rama Master y borra la rama que has usado.
6. Cuando tus compañeros hayan añadido estos dos métodos a la clase Factura, impleméntalos:
   getTotal();
   getCantidadIva();
7. Crea un commit con los cambios.
8. Vuelca el trabajo a la rama Master y borra la rama que has usado.
9. Sube los cambios de develop al repositorio remoto.
10. Cuando esté lista la primera puesta en producción mediante la rama master, bájatela a tu repositorio local.