UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Seminario de tecnología aplicada al diseño

Cátedra Diseño de sistemas

Agenda

- Git
- Maven
- Java
- Coffe Break
- Ejercicio de modelado de objetos (hasta el final)

GIT

GIT

Git es un herramienta de **control de versiones** (o sistema de versionado).

Una herramienta de control de versiones lleva adelante la gestión de los diversos cambios que se realizan sobre los elementos de algún código fuente.

GIT

¿Para qué me sirve?

- Compartir código con otras personas.
- Tener un historial de los cambios realizados en el código.

GIT

¿Por qué lo usamos en diseño?

GIT

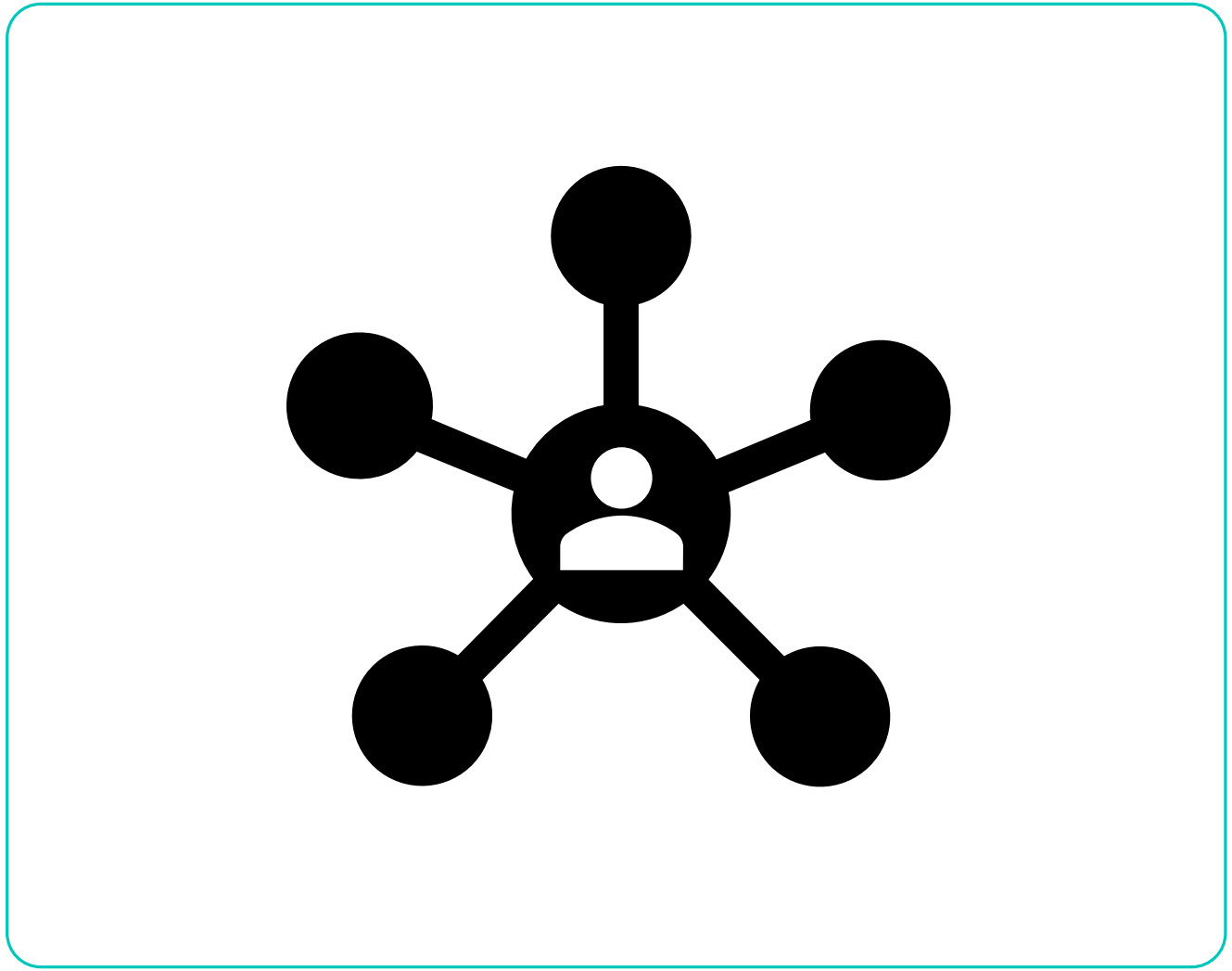
Vamos a estar trabajando con otras personas

Git nos permite:

- tener un historial de cambios
- saber quién los hizo y cuándo
- resolver los conflictos que surjan cuando dos o más personas modifiquen el mismo archivo (merge).

GIT

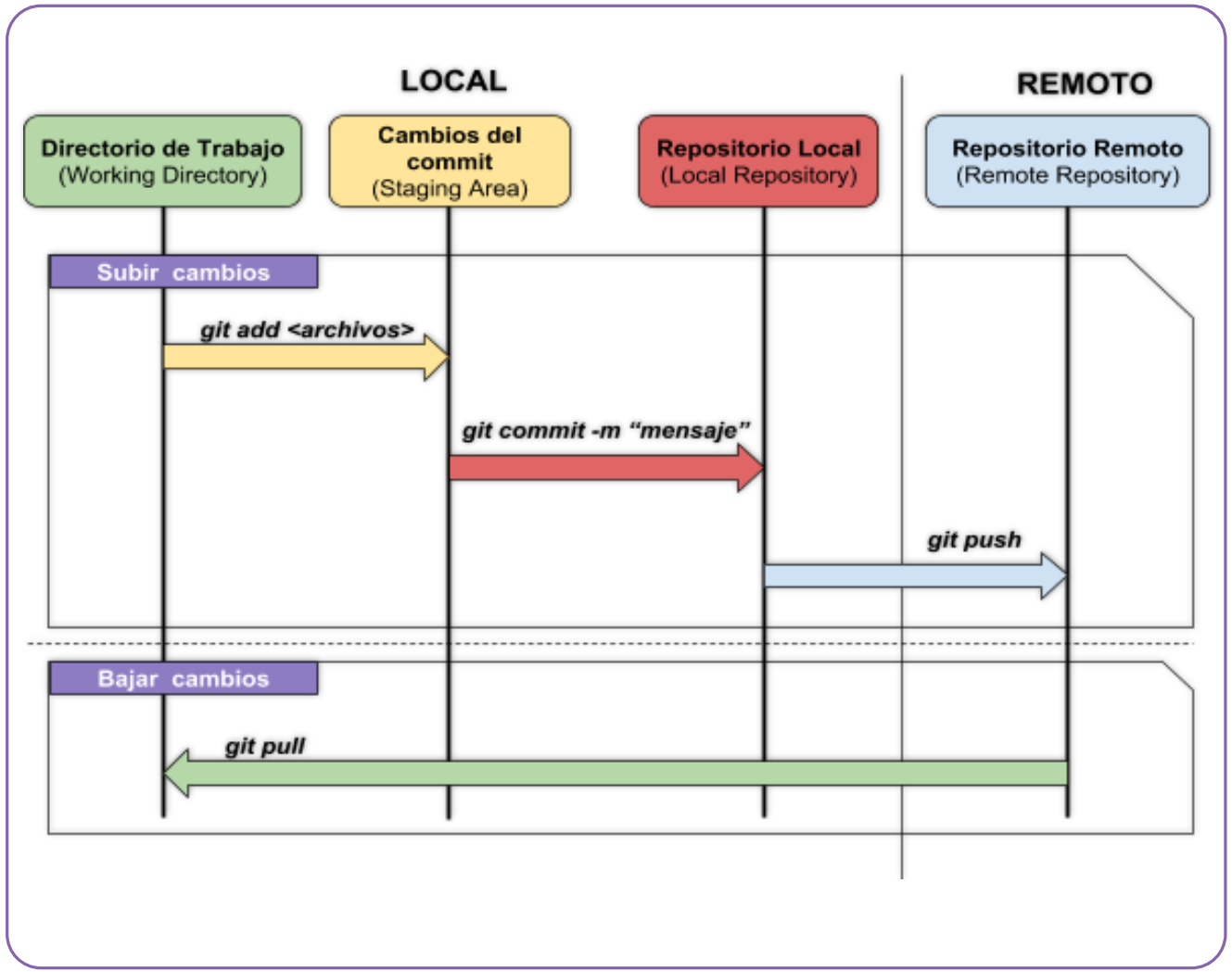
¿Cómo funciona?



GIT

- Creas/borras/modificas archivos en una carpeta asociada a un repositorio.
- Seleccionas los archivos que van a ser parte de un ***commit***.
- Confirmas el commit para agregarlo al repositorio.
- Subis los commits de tu repositorio local al repositorio remoto.

GIT


LOCAL
REMOTO
Directorio de Trabajo
(Working Directory)
Cambios del commit
(Staging Area)
Repositorio Local
(Local Repository)
Repositorio Remoto
(Remote Repository)
Subir cambios
git add <archivos>
git commit -m "mensaje"
git push
Bajar cambios
git pull

GIT

Commit

Un Commit es un conjunto de cambios en tus archivos que decidiste que tenían que ser versionados juntos, además le podes poner una descripción para que sepas que cambiaste.

GIT

Repositorio

Un Repositorio es el lugar donde van a estar todos los commits que forman parte de tu historial.

GIT

Repositorio Local vs Repositorio Remoto

GIT trabaja con un repositorio local que está en tu equipo, donde vas a ir agregando tus commits y uno remoto del cual podes subir tus commits o bajarte commits que haya subido alguien.

GIT

Comandos básicos

GIT

git config --global user.name
"TU NOMBRE"

git config --global user.email
"TU DIRECCION DE EMAIL"

GIT

git clone <url_repositorio_remoto>

Sirve para clonarnos un repositorio remoto

GIT

GIT STATUS

SIRVE PARA SABER SI HAY CAMBIOS QUE AÚN NO COMMITEASTE

GIT

- **git add <path>**

- Sirve para decirle a Git que los cambios hechos hasta el momento de un archivo/path debería ser commiteados

GIT

git commit -m “MENSAJE”

Sirve para realizar el commit de los archivos a los que previamente le hiciste git add

GIT

git reset <path>

Sirve para de-seleccionar los archivos que querías commitear

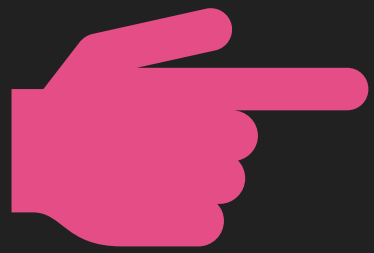
GIT

git checkout <path>

Volver a la versión vieja del path, perdiendo los cambios hechos hasta el momento

GIT

git push origin HEAD



Sirve para subir los commits de tu repositorio local

GIT

git pull

Sirve para bajarnos los cambios desde el repositorio remoto


Maven™

Maven

¿Qué nos resuelve?

- Creación automática de los proyectos para poder importarlos desde un IDE.
- Automatización de tareas de compilación/ejecución de test/cobertura, etc.
- Manejo de las dependencias de nuestro proyecto.

Maven

¿Cómo funciona?

En el directorio raíz del proyecto se deberá crear un archivo con el nombre pom.xml

En su versión más básica se parecerá a esto:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>nombre-proyecto</groupId>
  <artifactId>nombre-proyecto</artifactId>
  <version>1</version>
</project>
```

Maven

¿Cómo uso Maven para importar mi proyecto desde Eclipse o IntelliJ?

Maven

Para Eclipse: **mvn eclipse:clean**

Para IntelliJ: **mvn idea:clean**

Sirve para borrar cualquier configuración del proyecto anterior.

Maven

Para Eclipse: **mvn eclipse:eclipse**

Para IntelliJ: **mvn idea:idea**

Crea la configuración del proyecto.

Maven

- Para Eclipse: **mvn eclipse:eclipse -DdownloadSources=true -DdownloadJavadocs=true**
- Para IntelliJ: **mvn idea:idea -DdownloadSources=true -DdownloadJavadocs=true**

Crea la configuración del proyecto y baja el código fuente de las bibliotecas utilizadas.

Java 8

Maven

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>proyecto</groupId>
  <artifactId>proyecto</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Maven

mvn clean : Borra todos los archivos compilados que existan.

mvn compile : Compila todo el código fuente.

Maven

mvn test : Primero compila todo el código fuente si no estuviera compilado y luego ejecuta los tests.

mvn install : Ejecuta todos los pasos anteriores (compile, test, etc) y además junta el código compilado en un archivo .jar y deja la biblioteca disponible en mi sistema para que la pueda utilizar algún otro proyecto.

Maven

¿Qué es una ***dependencia***?

Podríamos decir que una dependencia en Maven es una *biblioteca* hecha por terceros que voy a tener disponible para utilizar en mi proyecto.

Si la biblioteca de terceros utiliza otra bibliotecas, entonces Maven se va a dar cuenta de eso y va a bajar esas dependencias también.

Maven

```
<project>
 ……….
 <!-- Dependencias -->
 <dependencies>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
    </dependency>

  </dependencies>
 ……….
</project>
```

Maven

Si ya habías importado el proyecto en Eclipse, acordate de hacer ***mvn eclipse:eclipse*** y refrescar el proyecto para que tome los cambios.

Si ya habías importado el proyecto en IntelliJ, acordate de hacer ***mvn idea:idea*** y refrescar el proyecto para que tome los cambios.

Java

Java

¿Cómo funciona?

Cuando programamos en Java, escribimos el código fuente en archivos con extensión ***.java***, eso se compila a bytecode (para esto necesitamos instalar la JDK que es la que trae el compilador) en unos archivos ***.class*** que puede correr la Java Virtual Machine (***JVM***).

Java

Es fuertemente tipado y estático.

Es compilado.

Java

¿THIS?

THIS ES UNA REFERENCIA AL OBJETO ACTUAL.

ENTONCES, POR EJEMPLO, CUANDO SE HACE ***THIS.VELOCIDAD*** SE ESTÁ HACIENDO REFERENCIA AL ATRIBUTO VELOCIDAD DEL OBJETO ACTUAL.

Java

Existen modificadores que indican quién puede tener acceso a las clases/atributos/métodos

Java

	Misma clase	**Clase y subclases**	**Otros**
private	Accede	No accede	No accede
protected	Accede	Accede	No accede
public	Accede	Accede	Accede

Java

¿Cómo consigo un objeto?

new

Java

Variable de clase

static

```
private static int comprasMaximas;
```

Java

Método de clase

static

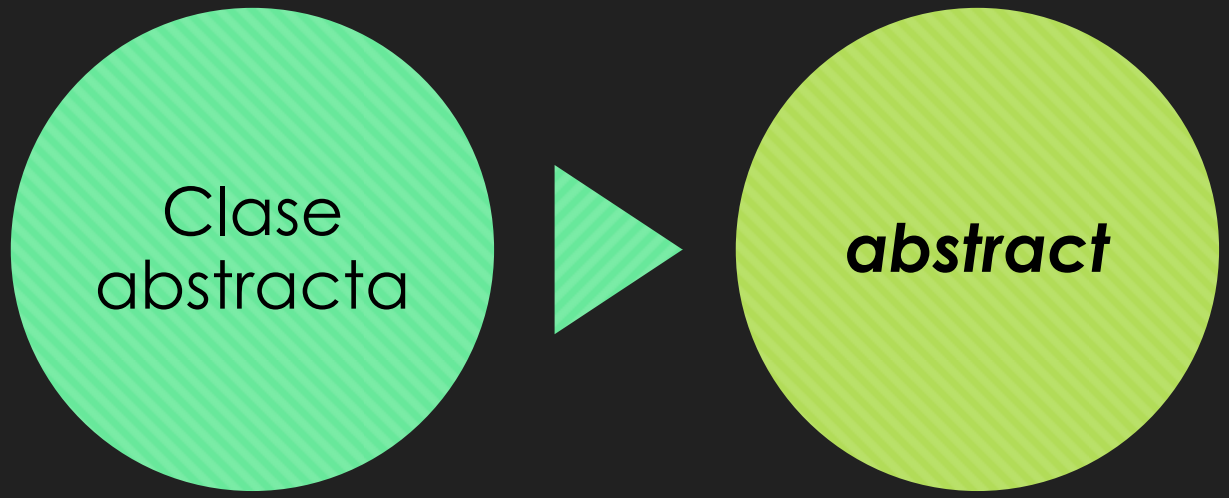
public static Auto getAuto(Motor unMotor){…}

Java

HERENCIA ***EXTENDS***

public class Auto extends Vehiculo { ….}

Java



public abstract class Vehiculo {…}

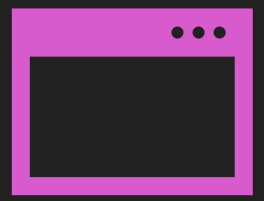
Java

Redefinición

super

```
public void encendete(){
    super.encendete();
    ....
}
```

Java



Interfaz ***interface***

public interface Manejable {…}

public class Auto implements Manejable{…}

Java

Colecciones

Java

- List
- Set
- Map

Java

LIST

Es una interfaz que extiende la interfaz Collection, agregando todos los mensajes que tienen que ver a colecciones que tienen un ***orden*** (indexado), por ejemplo: get(int index), remove(int index), sublist(int fromIndex, int toIndex), etc.

SET

Esta interfaz no agrega nuevos mensajes que deberían cumplir las colecciones que la implementen, pero sirve para indicar que las implementaciones de dicha interfaz ***no van a tener repetidos***. La decisión de si va a poner los elementos en la lista la va a hacer en base a la equivalencia de objetos.

Java

Java

MAP

Esta interfaz no extiende de Collection, porque lo que modela son los Dictionary/Map/Array-Asociativo, lo llaman de muchos nombres.

Básicamente es una estructura que guarda ***claves asociadas a valores***.

Por ejemplo le podemos decir: instanciaDeMap.put(key, value) y obtener el valor en base a la clave, haciendo: instanciaDeMap.get(key)

Java

List	Set	Map
ArrayList	HashSet	HasMap
LinkedList	LinkedHashSet	LinkedHashMap
Vector	TreeSet	Hashtable
		TreeMap

Java

	List	Set	Map
Duplicados	Si	No	No (las claves son únicas)
Orden	Ordenado	No; pero depende la implementación.	No; pero depende la implementación.
Accesso por índice	Si	No	Si

Java

¿Filter?

¿Map?

stream!

Java

Desde Java8, las colecciones ya soportan mensajes como “map” o “filter”, pero por una decisión de diseño, decidieron agregarlos a otro objeto que se llama Stream y se lo podes pedir a las colecciones mandándole el mensaje stream()

Java

```
personas.stream()
.filter(persona ->
persona.esMayorDeEdad())
```

Tené en cuenta cuenta que lo que devuelve el filter(..) sigue siendo un Stream, entonces para obtener de nuevo una lista tenes que usar el mensaje ***collect(..)***.

Java

```
personas
    .stream()
    .filter(persona -> persona.esMayorDeEdad())
    .map(persona -> persona.getNombre())
    .collect(Collectors.toList())
```

Java

Annotations

Sirven para agregar metadata a los elementos de una clase (definición de la clase/atributo/métodos)

Java

@Override

Es la annotation para sobrescribir un método

Java

ENUMS

```
public enum Dia {   LUNES, MARTES, MIERCOLES,
JUEVES, VIERNES, SABADO, DOMINGO };
```

Java

Excepciones

Chequeadas

No chequeadas

Java

Las excepciones chequeadas heredan de ***Exception*** y en las firmas de los métodos que van a dejar propagar la excepción (sea porque la lanzan o no la atrapan) deben indicar qué puede lanzar una excepción de ese tipo.

Java

```
public void comprar() throws SaldoInsuficienteException
```

Java

Las no chequeadas heredan de ***RuntimeException*** y no es obligatorio definir en la firma que el método las puede lanzar.

Java

```
public class SaldoInsuficienteException extends RuntimeException {

 public SaldoInsuficienteException(String message) {
   super(message);
 }

}
```

Java

```
public void comprar() {
 try {
   // código que puede lanzar una SaldoInsuficienteException
 } catch (SaldoInsuficienteException e) {
   // Hago algo, por eso la atrape ;)
 } finally {
   // Se ejecuta siempre
 }
}
```

¡Gracias!

Seminario de tecnología aplicada al diseño

UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS