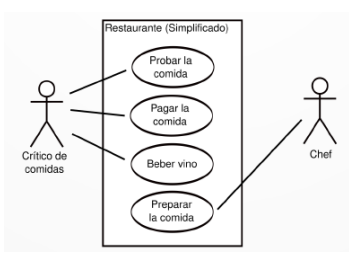
**UML**

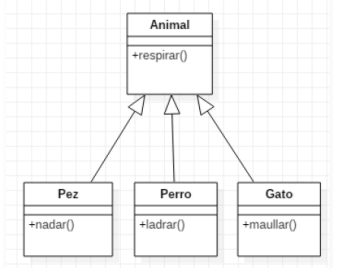
El lenguaje unificado de modelado (UML) es un lenguaje para modelar, construir, y documentar un sistema software orientado a objetos. Es un estándar e ingeniería del software. Aunque tiene 13 diagramas, nosotros solo vemos 3: casos de uso, clases, secuencia.

Los diagramas de casos de uso son una herramienta para determinar los requisitos del sistema por medio de un análisis de las funciones que tiene que realizar. El sistema se representa por un rectángulo que engloba los casos de uso representadas como elipses y los roles de quienes usan estos casos de uso.

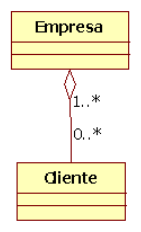


En los diagramas de casos de estudio, los casos de uso, a su vez pueden tener otros casos de uso que dependan de ellos (subfunciones), esto se muestra mediante una relación de <<include>>. Sin embargo, si una subfunción no se usa siempre, se usa una relación <<extend>>.

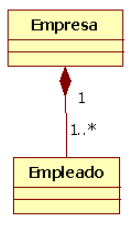
Los diagramas de clase modelan el sistema explicado en el diagrama de flujos en clases y relaciones de forma estática, que pueden ser programadas. Las relaciones pueden ser de asociación (la normal), herencia (de persona a cliente, proveedor), agregación (clases que forman parte de otra clase, pero que no desaparecen si la clase principal es destruida, ejemplo empresa y cliente), y composición (lo mismo que agregación, pero si se destruye la clase principal, también se destruye la clase secundaria, ejemplo empresa, trabajador).



***Herencia***

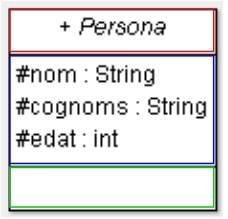


***Agregación***



***Composición***

Una clase es un elemento que agrupa elementos que comparten propiedades como: comportamiento, relaciones, o semántica. El contenido de toda clase es: su nombre, su estereotipo, sus atributos, su visibilidad, y su tipo. La clase se divide en 3 secciones: la primera es para el estereotipo y el nombre de la clase, la segunda es para los atributos, y la tercera es para los métodos.



El nombre de la clase siempre empieza en mayúsculas. No puede haber dos clases con el mismo nombre, el nombre de la clase no puede contener espacios, y si está en cursiva, significa que la clase es abstracta. El estereotipo define las propiedades, comportamientos, y restricciones preestablecidas en la clase y se escriben así <<estereotipo>> colocándose sobre el nombre de la clase. Los atributos se definen con su nombre, y su tipo, siempre empiezan en minúsculas, y sus nombres no pueden contener espacios. Los métodos definen los comportamientos que tendrá la clase (sus funciones), de nuevo siempre comienzan con minúsculas y no pueden contener espacios. A todo nombre de clase, atributo, y función se le aplica una visibilidad en base a un símbolo justo delante de esta. Este símbolo puede ser: + si es público, ~ si es del package predeterminado, # si es protegido, y – si es privado.

**Comandos de github**

Primero git init

Luego git config --global user.name “Nombre del usuario”

git config --global user.email correo electrónico

No olvidar hacer el fork y el git pull request

Revisar el estatus del trabajo git status

Visualizar cambios git diff <archivo>

Para enviar archivos git add <archivo>, luego git commit -m “mensaje”, y por último git push

Para ver el historial de cambios git log

Para ver cambios en los commit git show <commit>

Quitar el archivo del área de preparación git reset HEAD <archivo>

Etiquetar algo git tag

Crear ramas git branch <nombre de la rama>

Ver las ramas disponibles git branch

Cambiar de rama git checkout

Fusionar ramas git merge <nombre de la rama a fusionar>

Eliminar una rama git branch -d <nombre de la rama>

Clonar un repositorio git clone

Ver repositorios remotos git remote -v

Añadir remoto git remote add <nombre remoto> <url repositorio>

Eliminar remoto git remote rm <nombre remoto>

Renombrar remoto git remote rename <nombre remoto original> <nombre remoto nuevo>

Traer y fusionar cambios del remoto git pull [nombre remoto] [nombre rama]

Enviar cambios al remoto git push [nombre remoto] [nombre rama]

Enviar los cambios de una rama al remoto y crear una rama remota asociada git push -u nombre remoto nombre rama

**Preguntas test de prácticas de github**

* ¿Cuál de los siguientes comandos crea una nueva rama local llamada develop?

**git checkout -b develop**

* ¿Cuál es el comando para listar los cambios (commit) que han producido en el repositorio y ver el cambio que han seguido sus ramas?

**git log --all --graph**

* ¿Qué comando es necesario para crear un repositorio de trabajo local?

**git init**

* ¿Qué ha salido mal si después de hacer un commit obtengo el siguiente mensaje? Your branch is up to date with ‘origin/master’.

**No he añadido al área de preparación mis últimos cambios.**

* ¿Qué operación es necesaria para copiar un repositorio remoto de un proyecto en el que no participas de su organización?

**Necesitas hacer un fork.**

* ¿Qué operación es necesaria para solicitar que una aportación en un repositorio remoto de un proyecto público sea añadida?

**Necesitas pull request**

* ¿Qué resultado obtendré al ejecutar el comando git diff HEAD~1?

**Muestra las diferencias entre el estado actual y el penúltimo commit.**

* ¿Qué resultado obtendré al ejecutar git --reset HEAD~1?

**El HEAD pasa a apuntar al penúltimo commit y en el directorio de trabajo se deshacen los últimos cambios.**

* ¿Qué resultado obtendré al ejecutar git revert HEAD?

**HEAD apunta a un nuevo commit en el que se han deshecho los últimos cambios.**

* Si deseas obtener una copia de un repositorio Git existente en GitHub, el comando que necesitas es:

**git clone**