




Módulo Profesional 05: Entornos de desarrollo

UF3 Actividad A

**CICLO FORMATIVO DE GRADO SUPERIOR EN
DESARROLLO DE APLICACIONES MULTIPLATAFORMA
MODALIDAD ONLINE**



ALUMNA:

MARÍA LAURA RONDINA IOBBI

Descripción de la actividad:

En la siguiente actividad se valorarán los conceptos más importantes de la actividad, Por lo que deberás de dar respuesta a los siguientes ejercicios en el mismo documento para entregar por el campus. Poner nombre al documento y entregar en formato Word/pdf.

1. Los diagramas se agrupan en dos grandes grupos, ¿Cuáles son? Indícalos y defínelos. [1,0 puntos]

Modelar consiste en hacer un diseño previo de una aplicación antes de proceder a su desarrollo e implementación, y para ello se utilizan los diagramas UML, que pueden agruparse en dos grandes grupos:

1) Diagramas de comportamiento. A diferencia de los diagramas estructurales, muestran cómo se comporta un sistema de información de forma dinámica, describiendo los cambios que sufre un sistema a través del tiempo cuando está en ejecución. Hay un total de 4 diagramas de comportamiento, clasificados de la siguiente forma:

-Diagrama de actividades, también llamados modelos de flujo de control y flujo de objetos. Se usa para mostrar una secuencia de actividades, mostrando el flujo de trabajo desde el punto de inicio hasta el punto final y detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad. También pueden usarse para detallar situaciones donde el proceso paralelo puede ocurrir en la ejecución de algunas actividades.

-Diagrama de casos de uso. Describe un conjunto de acciones (casos de uso) que algunos sistemas o sujetos deben realizar en colaboración con uno o más usuarios externos del sistema (actores) para proporcionar unos resultados observables y valiosos a los actores u otros interesados del sistema.

-Diagrama de máquina de estados. Ofrecen un método orientado a objetos de mostrar el comportamiento de un objeto y documentar cómo el objeto responde a determinados eventos, incluidos estímulos internos y externos.

- Diagramas de interacción. Es un subconjunto de los diagramas de comportamiento. Comprende los siguientes diagramas:

Diagrama de secuencia:

Es el tipo más común de diagramas de interacción y se centra en el intercambio de mensajes entre líneas de vida (objetos) para ejecutar una función antes de que la línea de vida termine.

Diagrama de comunicación:

Modela las interacciones entre objetos o partes en términos de mensajes en secuencia. La secuencia de mensajes se da a través de una numeración.

Diagrama de tiempos:

Son una representación especial de interacción en forma de ondas digitales que se enfoca en el tiempo de los mensajes enviados entre objetos.

Diagrama global de interacciones:

Describe en detalle un determinado escenario de un caso de uso, donde un conjunto de objetos cooperan en la realización de dicho escenario. Brindan una descripción general del flujo de control donde los nodos del flujo son interacciones.

2) Diagramas estructurales. Presentan modelos estáticos del objeto, como clases, paquetes o componentes, aunque también pueden ser utilizados como modelos en tiempo de ejecución. Muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción. Existen un total de 7 tipos de diagramas de estructura:

- Diagrama de clases:

Describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

- Diagrama de componentes:

Muestra cómo un sistema de software es dividido en componentes, y las dependencias entre ellos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes.

- Diagrama de despliegue:

Muestra la arquitectura del sistema como despliegue (distribución) de artefactos de software. Los elementos usados son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.

- Diagrama de objetos:

Un gráfico de instancias, incluyendo objetos y valores de datos. Es una instancia de un diagrama de clase, que muestra una vista completa o parcial de los objetos de un sistema en un instante de ejecución específico.

- Diagrama de paquetes:

Muestra los paquetes y las relaciones entre los paquetes.

- Diagrama de perfiles:

Diagrama UML auxiliar que permite definir estereotipos personalizados, valores etiquetados y restricciones como un mecanismo de extensión ligero al estándar UML. Los perfiles permiten adaptar el metamodelo UML para diferentes plataformas o dominios.

- Diagrama de estructura compuesta:

Muestra la estructura interna (incluidas las partes y los conectores) de un clasificador estructurado, que representa una clase. Una *estructura compuesta* es un conjunto de elementos interconectados que colaboran en tiempo de ejecución para lograr algún propósito, donde cada elemento tiene algún *rol* definido en la colaboración.

2. ¿Qué es un diagrama de Comportamiento? ¿Y para qué se utiliza? [1,0 puntos]

Un **diagrama de comportamiento** se define formalmente como un diagrama que expresa las secuencias de estados por los que pasa un objeto a lo largo de su vida en respuesta a eventos. Se trata de diagramas que muestran diferentes estados de un proceso, y mediante los cuáles se plasman de forma gráfica los procesos a programar. En definitiva, son aquellos que nos muestran la forma

que tienen de reaccionar ante los estímulos externos. Son diagramas dinámicos, basados en su devenir en tiempo de ejecución.

Estos diagramas **se usan para visualizar y especificar, a la vez que documentar**, aspectos dinámicos de un sistema. Cuando hablamos de desarrollo de software, estos aspectos dinámicos pueden ser mensajes que se generan, acciones de entrada de datos, eventos, etc...

Los diagramas de comportamiento pueden ser muy simples o muy complejos en función del proceso que están representando. Hacen referencia a objetos, ya que se emplean mucho en la programación orientada a objetos. Un objeto puede ser cualquier entidad con un determinado estado y comportamiento.

Por ejemplo, mediante un diagrama de este tipo podremos comprobar los diferentes estados de una máquina cortadora de hilos en una línea de producción textil durante las veinticuatro horas de un día laborable; es decir, podremos observar su evolución durante la jornada de trabajo para conocer su comportamiento y su relación con otros elementos externos.

3. ¿En cuántos grupos se dividen los Diagramas de Comportamiento? Menciónalos y descríbelos. [1,0 puntos]

Podemos clasificar los diagramas de comportamiento en dos grupos: **diagramas de comportamiento** propiamente dichos y **diagramas de interacción**; no obstante, este último se considera un subgrupo del anterior.

1) Diagramas de comportamiento:

-**Diagrama de actividad.** Se utiliza para mostrar la traza de operaciones que se realiza en un proceso del sistema cuestionado.

-**Caso de uso.** Muestra como un elemento cliente, llamado actor, interactúa con el sistema.

-**Máquina de estados.** Muestra el comportamiento de un objeto en cuanto a sus cambios de estado resultantes de la aplicación secuencial de eventos.

2) Diagramas de interacción:

-**Secuencias.** Los diagramas de secuencia ofrecen una visión de las interacciones de los objetos, ordenadas según el momento en el que ocurren.

-**Colaboración.** Los diagramas de colaboración muestran la interacción y vinculaciones que existen entre los objetos.

-**Interacción general.** Focaliza la visión general del flujo de control de las interacciones mediante el uso de marcos adicionales. Se considera una variante del diagrama de actividad.

-**Tiempo.** El diagrama de tiempo visualiza el comportamiento de los objetos en el transcurso del tiempo.

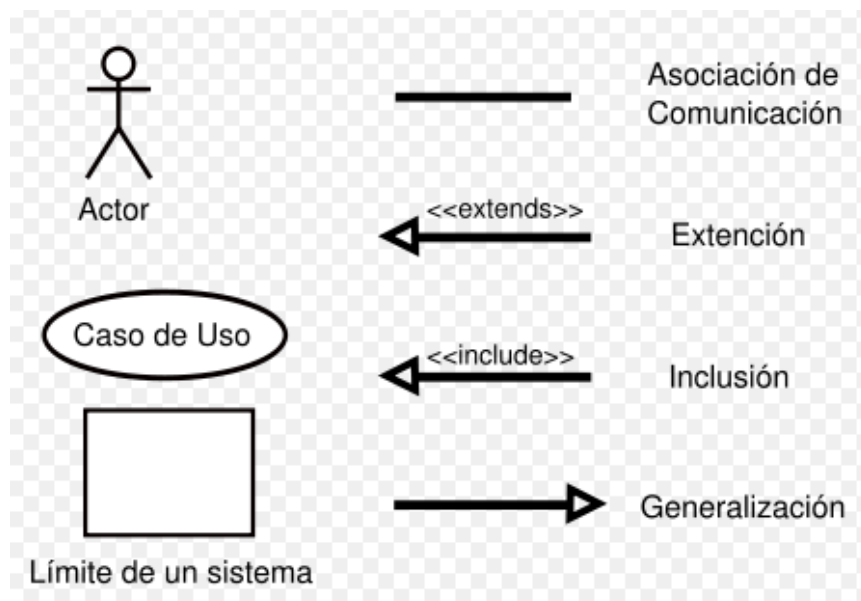
4. ¿Qué representan los Diagramas de Caso de uso? ¿Qué son los actores dentro de los Diagramas de casos de uso y cómo se representan? Ejemplificarlo. [1,0 puntos]

Los **diagramas de casos de uso** sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas, mostrando la relación entre los actores y los casos de uso en un sistema. Se emplean diversos componentes en los casos de uso, como las distintas Relaciones de comunicación, Requerimientos (que representan las funcionalidades que el caso de uso debe proveer al usuario final (actor)), Limitaciones (que representan las condiciones previas que deben darse para poder aplicar el caso de uso, y las condiciones posteriores que deben ser ciertas después de realizarse el caso de uso) y Escenarios (que describen el flujo de eventos acaecidos durante la ejecución de un caso de uso).

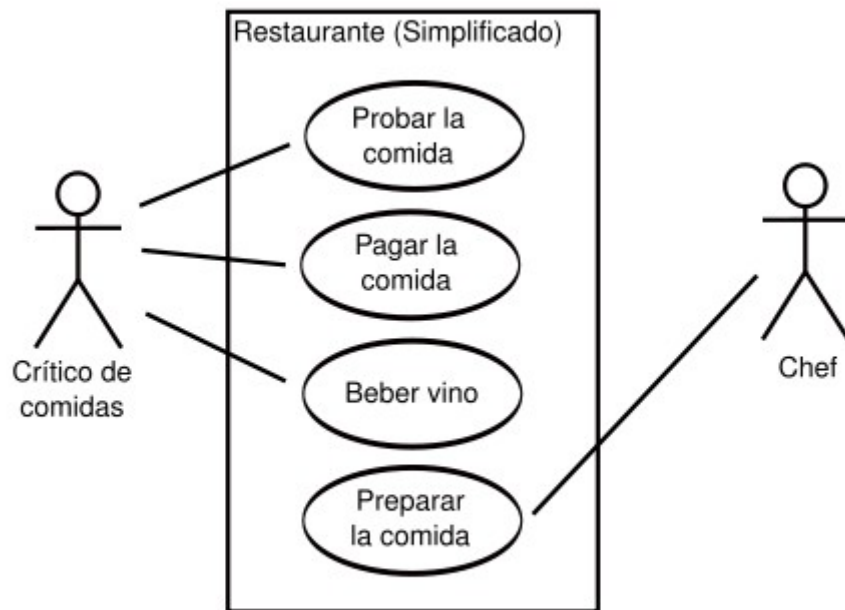
En el Lenguaje Unificado de Modelado (UML), un **actor** especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto. Los actores pueden representar roles jugados por usuarios humanos, hardware externo, u otros sujetos. No necesariamente representan una entidad física específica, sino simplemente una faceta particular (es decir, un "rol") de alguna actividad que es relevante a la especificación de sus casos de uso asociados. Así, una única instancia física puede jugar el rol de muchos actores diferentes y, asimismo, un actor dado puede ser interpretado por múltiples instancias diferentes. A los actores **podemos representarlos con un dibujo o caricatura, o también con un rectángulo**, tipo clase, con el nombre del actor.

UML 2 no permite asociaciones entre Actores. A pesar de todo, esta restricción es usualmente violada en la práctica, en la medida que la generalización/especialización de relaciones entre actores es útil para el modelado de los comportamientos de superposición entre actores.

Ilustración de un actor, junto con los demás elementos de un diagrama de casos de uso.



Ejemplo sencillo de diagrama de casos de uso, donde se muestran los actores *crítico de comidas* y *chef*:



El diagrama describe la funcionalidad de un **Sistema Restaurante** muy simple. Los **casos de uso** están representados por elipses y los **actores** con caricaturas. Los casos de uso se muestran como parte del sistema que está siendo modelado, los actores no.

En este caso, podemos apreciar tanto declaraciones correctas como incorrectas. El probar la comida y pagarla es un **requerimiento funcional** del sistema, pero beber vino no lo es, por lo tanto este caso de uso está incorrecto.

La **interacción entre actores** no se ve en el diagrama de casos de uso. Si esta interacción es esencial para una descripción coherente del comportamiento deseado, quizás los límites del sistema o del caso de uso deban de ser re-examinados. Alternativamente, la interacción entre actores puede ser parte de suposiciones usadas en el caso de uso. Sin embargo, los actores son una especie de **rol**, y un usuario humano u otra entidad externa puede jugar varios papeles o roles. Así, el Chef y el Cajero podrían ser realmente la misma persona.

5. ¿Qué es un Diagrama de Secuencia? y ¿Cuáles son los elementos que conforman un Diagrama de Secuencia? [1,0 puntos]

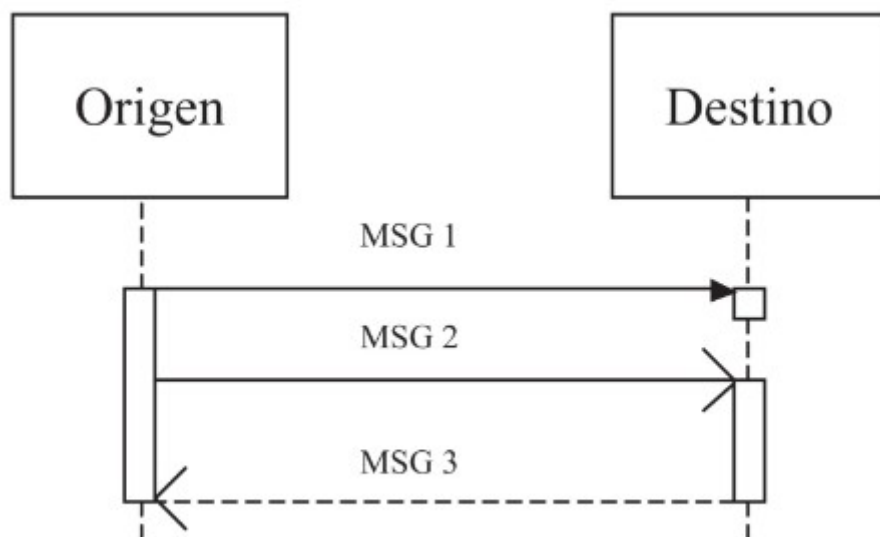
Un **diagrama de secuencias** es un tipo de diagrama de interacción que muestra el progreso de los **objetos** que lo componen a través del tiempo, representado por unas líneas, llamadas **líneas de vida**, que se despliegan como una línea discontinua hacia abajo en el espacio reservado para nuestro modelaje, y partiendo del símbolo del objeto. Los diagramas de secuencia muestran la comunicación entre objetos y las circunstancias en las que se produce.

En el transcurso de la línea de vida de un objeto, puede ocurrir que un objeto emita o reciba un **mensaje**. Este mensaje desencadenará la activación de determinadas operaciones en el objeto

receptor. En general, podemos clasificar los mensajes en dos tipos: síncronos y asíncronos. Los **mensajes síncronos** se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que envía el mensaje queda bloqueado hasta que termina la llamada. Es decir, el objeto emisor del mensaje espera constatar que el receptor ha recibido dicho mensaje. Este tipo de mensajes se representan con flechas con la punta rellena. En los **mensajes asíncronos** el objeto emisor no espera que el mensaje sea recibido y continúa su ejecución de procedimientos si los tuviere. Es decir, estos mensajes terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la punta hueca.

A causa de la recepción de un mensaje, se **activa** una ocurrencia en el objeto. Éste se muestra mediante un rectángulo vertical tan largo como su duración en el tiempo. La respuesta a un mensaje se representa con una flecha discontinua.

Ejemplo: Diagrama de secuencia con las líneas de vida, activaciones (rectángulos) y envío y recepción de mensajes entre objetos.



MSG1 es un mensaje síncrono (representado por una flecha con cabeza rellena); MSG2 es un mensaje asíncrono (representado por una flecha con punta hueca), y MSG3 es un mensaje de retorno asíncrono (línea discontinua).

6. ¿Qué es un Diagrama de Colaboración? y ¿Cuáles son los elementos que conforman un Diagrama de Colaboración? [1,0 puntos]

El **diagrama de colaboración**, también conocido como **diagrama de comunicación**, es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información mostrando cómo interactúan los objetos entre sí, es decir, con qué otros objetos tiene vínculos o intercambia mensajes un determinado objeto.

Un diagrama de colaboración muestra la misma información que un diagrama de secuencia pero de forma diferente. En los diagramas de colaboración no existe una secuencia temporal en el eje vertical; es decir, la colocación de los mensajes en el diagrama no indica cuál es el orden en el que se suceden. Además, la colocación de los objetos es más flexible y permite mostrar de forma más clara cuáles son las colaboraciones entre ellos. En estos diagramas la comunicación entre objetos se denomina vínculo o enlace (*link*) y estará particularizada mediante los mensajes que intercambian.

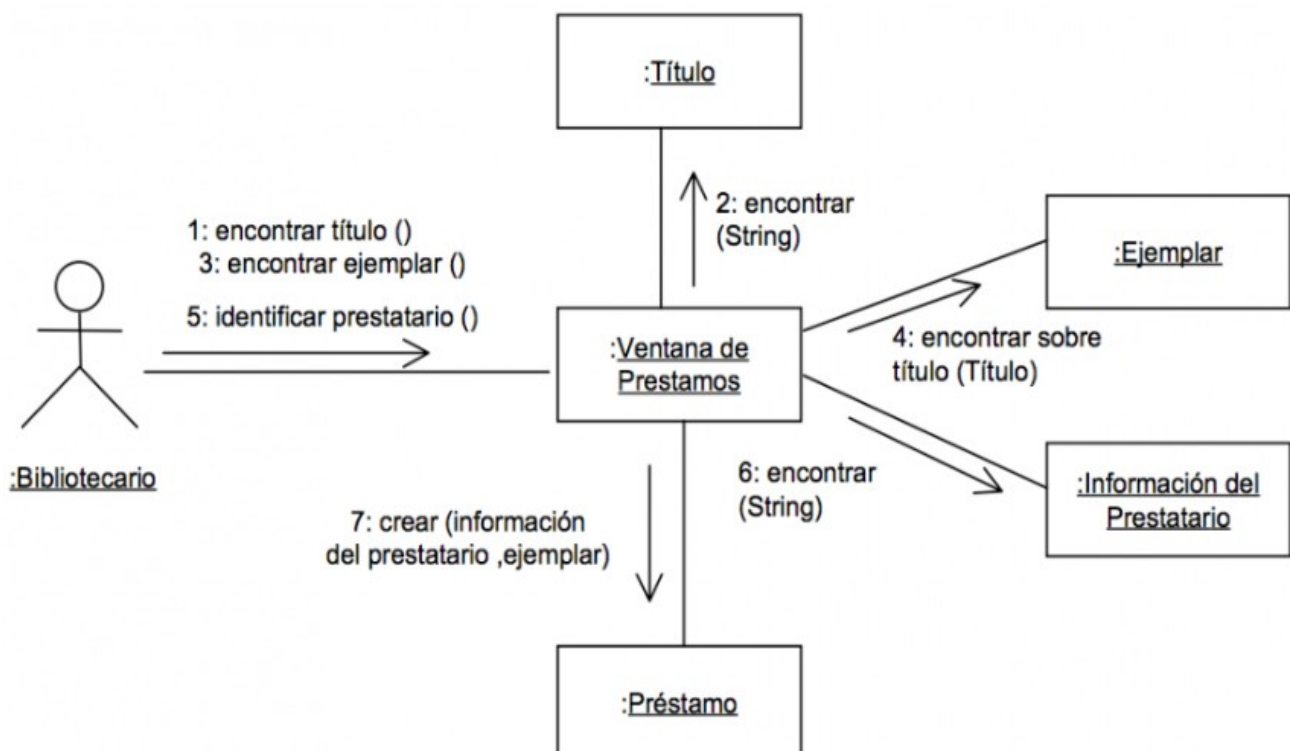
Los elementos que conforman este tipo de diagramas son los siguientes:

-OBJETOS. Un objeto se representa con un rectángulo dentro del que se incluye el nombre del objeto y, si se desea, el nombre de la clase, separando ambos por dos puntos.

-VÍNCULO Ó ASOCIACIÓN. Un vínculo se representa como una línea continua que une ambos objetos y que puede tener uno o varios mensajes asociados en ambas direcciones. Como un vínculo instancia una relación de asociación entre clases, también se puede indicar la navegabilidad del mismo mediante una flecha. Esta asociación de roles muestra cómo se comportarán ambos roles en una situación concreta.

-MENSAJES. Un mensaje se representa con una pequeña flecha colocada junto a la línea del vínculo al que está asociado. La dirección de la flecha va del objeto emisor del mensaje al receptor del mismo. Junto a ella, se coloca el nombre del mensaje y sus argumentos. A diferencia de los diagramas de secuencia, en los diagramas de colaboración siempre se muestra el número de secuencia del mensaje delante de su nombre, ya que no hay otra forma de conocer la secuencia de los mismos. Se utiliza si es preciso notación anidada (Ej: 1.2, 1.3). Además, los mensajes pueden tener asociadas condiciones e iteraciones que se representarán como en los diagramas de secuencia.

-ROLES DE CLASE. Describen cómo se comportan los objetos. Para representarlos se utiliza el símbolo de objeto, pero sin listar los atributos.

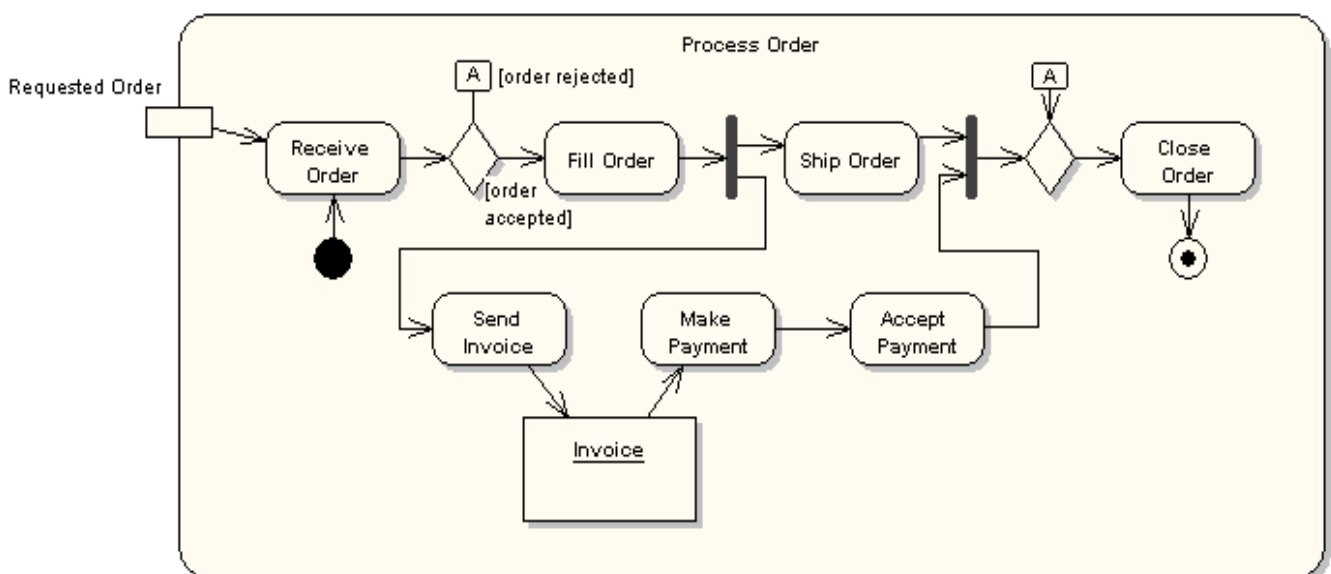


7. ¿Qué es un Diagrama de Actividades? y ¿Cuáles son los elementos que conforman un Diagrama de Actividades? [1,0 puntos]

El **diagrama de actividades** se utiliza para mostrar una secuencia de actividades. Se suele indicar el inicio y el final de la actividad, detallando todos los caminos que bifurquen mientras dura su realización. Es muy útil para detallar actividades paralelas, es decir, aquellas ejecutadas simultáneamente, lo que permite determinar sus ejecuciones y dependencias entre sí y tener una visualización completa del comportamiento dinámico del sistema.

El diagrama de actividades se compone de los siguientes elementos:

- **Estados iniciales y finales.** El inicio del diagrama se representa con un círculo relleno, del que surgirá el circuito de actividades con sus transiciones. Por último, un círculo relleno circunscrito en uno sin rellenar indica el estado final, y fin de diagrama.
- **Actividad.** Es un consenso que tenemos sobre una determinada secuencia de conducta. El alcance de dicha actividad puede ser general o detallado. La actividad se representa con un rectángulo de puntas redondeadas.
- **Acción.** Es un paso fundamental dentro de una actividad. Utiliza el mismo símbolo, aunque se coloca dentro del de la actividad.
- **Transiciones.** Los momentos de cambio de actividad están simbolizados mediante una flecha de cabeza normal. La transición indica qué actividad o acción es la que sigue a continuación.
- **Decisiones.** Las decisiones, bifurcaciones en el flujo de actividades en función de que se cumpla una determinada condición, se representan con un cuadrado con un vértice de base. Ejemplo: podría ocurrir que, al introducir la tarjeta de pago en un cajero, el lector no funcionara; en estos casos, los flujos alternativos deben tenerse en cuenta.
- **Condición necesaria.** Mediante corchetes puede mencionarse una condición que se debe cumplir para que el flujo continúe.
- **Combinación (Fusión).** Dos flujos alternativos, creados por una decisión previa, vuelven a converger. El símbolo es el mismo que el utilizado para la decisión.



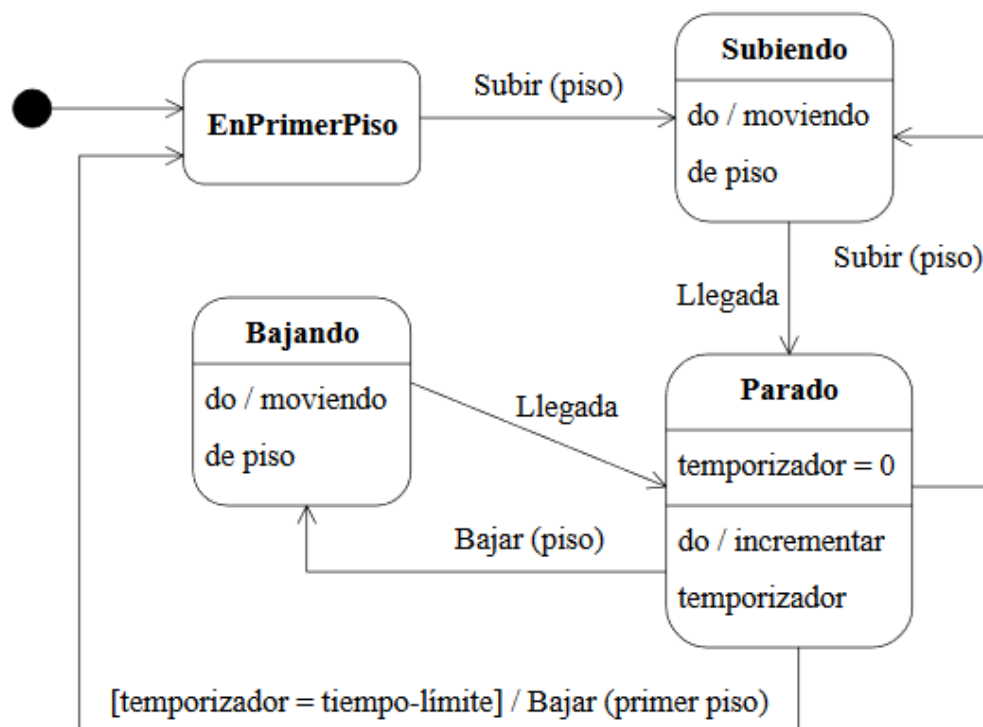
8. ¿Qué es un Diagrama de Estado? y ¿Cuáles son los elementos que conforman un Diagrama de Estado? [1,0 puntos]

Un **diagrama de estado (diagrama de máquina de estado)** se encarga de mostrar el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación en respuesta a eventos (por ejemplo, mensajes recibidos, tiempo rebasado o errores), junto con sus respuestas y acciones. También ilustran qué eventos pueden cambiar el estado de los objetos de la clase.

Se compone de los siguientes elementos:

- **Estado.** Un estado de objeto se simboliza con un rectángulo de puntas redondeadas con el nombre del estado en su interior. Por ejemplo, una bombilla tendría dos estados, que se representarían mediante dos rectángulos de puntas redondeadas: uno para el estado encendido y otro para el estado de apagado.
- **Eventos.** Los sucesos que pueden cambiar los estados son finitos. En el caso de la bombilla, el estado puede cambiar al responder al evento “encender” cuando está apagada, y al responder al evento “apagar” cuando está encendida.
- **Señal.** Un mensaje enviado por un objeto que provoca una transición en otro objeto se considera una señal; la señal también se considera como un objeto.
- **Transiciones.** Las transiciones de un estado a otro se simbolizan mediante una flecha. Las transiciones pueden tener los siguientes elementos: - **Disparador.** Es la causa de la transición. Puede ser una señal, un evento o un cambio en alguna condición. - **Guarda.** Condición necesaria para que el disparador cause la transición. - **Efecto.** Acción en el objeto que se desencadena por la transición.

Ejemplo: Diagrama de estado para un ascensor, donde se combinan los estados con las transiciones simples.



El ascensor empieza estando en el primer piso. Puede subir o bajar. Si el ascensor está parado en un piso, ocurre un evento de tiempo rebasado después de un período de tiempo y el ascensor baja al primer piso. Este diagrama de estado no tiene un punto de finalización (estado final). El evento de la transición entre los estados EnPrimerPiso y Subiendo tiene un argumento, piso (el tipo de este parámetro ha sido suprimido). Lo mismo sucede con los eventos de las transiciones entre Parado y Subiendo y entre Parado y Bajando. El estado Parado (Idle state) asigna el valor cero al atributo temporizador, luego lo incrementa continuamente hasta que ocurra el evento Bajar (piso) o el evento Subir (piso) o hasta que la guard-condition [temporizador = tiempo-límite] se convierta en verdadera. La transición de estado entre Parado y EnPrimerPiso tiene una guard-condition y una expresión-acción. Cuando el atributo temporizador es equivalente a la constante tiempo-límite, se ejecuta la acción Bajar (primerpiso) y el estado del ascensor cambia de Parado a EnPrimerPiso. Esta transición de estado [temporizador = tiempo-límite] / Bajar (primerpiso) se puede convertir en una cláusula-envío tal como: [temporizador = tiempo-límite] ^ Self.Bajar (primerpiso)

9. En la programación orientada a objetos existen los siguientes conceptos: clase, atributo, método y visibilidad. Defínelos. [1,0 puntos]

- **Clase.** Una clase es una abstracción que define un tipo de objeto especificando qué propiedades y operaciones disponibles va a tener. Es un patrón o “plantilla” para construir objetos (que son variables pertenecientes a una clase determinada). La clase es una declaración, no tiene asociado ningún objeto. Pero todo objeto debe pertenecer a una clase.

En Java crearemos la clase precedida de la palabra clave public, la cual afecta a en qué partes del programa o por parte de quién se va a poder acceder a ella. El espacio comprendido entre la apertura de la clase y su cierre (es decir, el espacio comprendido entre los símbolos { y } de la clase), es el Cuerpo de la clase. Un esquema que explicaría las partes de una clase sería:

Clase Taxi { → Taxi es el nombre de la clase

Propiedades: → Matrícula

(Atributos) → Distrito por el que circula

→ Tipo de motor (diésel o gasolina)

Constructor de la clase → Definición de qué ocurre cuando se crea un objeto del tipo definido por la clase.

Operaciones disponibles → los Métodos de la clase:

➔ Asignar una matrícula

➔ Asignar un distrito

➔ Asignar un tipo de motor

- **Atributo.** Se trata de una característica o propiedad aplicada a un elemento de la clase. Son todos aquellos datos que van a ser compartidos por todos los objetos que de dicha clase se deriven. Los definiremos normalmente después de la apertura de la clase, fuera de los constructores o métodos que puedan existir.

Los datos incluidos en un objeto son como las variables en los lenguajes de programación clásicos, pero están encapsuladas dentro de un objeto y, salvo que se indique lo contrario, son invisibles desde el exterior. En este sentido, resulta importante considerar que existen tres tipos de atributos:

* **Públicos (public).** Son aquellos a los que se puede acceder o modificar directamente, tanto a nivel interno como externo de la clase. * **Privados (private).** Sólo pueden ser modificados directamente a nivel interno de la clase a la que pertenecen. * **Protegidos (protected).** Son accesibles desde la clase que los define (similar en este punto al anterior), pero además también lo son desde las clases derivadas, es decir, desde cualquier otra que herede de esta clase inicial. * También se define como **propiedad estática (static)** a aquella que es única a nivel de clase, no de objeto.

● **Método.** En la programación, un método es una subrutina cuyo código es definido en una clase y puede pertenecer tanto a una clase, como es el caso de los **métodos de clase o estáticos**, como a un objeto, como es el caso de los **métodos de instancia**. Como elemento de la clase, el método contiene un procedimiento que fijará la conducta manifestada por los objetos pertenecientes a dicha clase. Representan a las acciones o funciones comunes a todos los objetos que de dicha clase se deriven, e igual que los atributos, pueden ser de tipo: ***public, *private y *protected**. Como métodos de clase, definirán las funciones que serán capaces de desempeñar nuestras clases a lo largo de la ejecución del programa.

● **Visibilidad.** La visibilidad de una propiedad o de un método se define como la posibilidad que tengan de ser accedidos desde otras clases.

Dependiendo de cada lenguaje de programación, la visibilidad de los objetos se declara de distintas maneras, usando palabras reservadas o definiendo algún prefijo en el nombre de la variable. En Java, usaremos las palabras reservadas public, private y protected antes definidas. Si no hubiera declarada ninguna visibilidad, el objeto tendrá visibilidad de paquete, es decir, sólo se podrá acceder a él desde la propia clase o desde el paquete que lo contiene. Por el principio de encapsulación, y a no ser que se vayan a usar para otra cosa, todos los atributos de una clase tienen que ser private, de manera que sólo se pueda acceder a ellos mediante métodos provistos por la propia clase para ello.

10. Define con tus propias palabras las relaciones: Herencia, composición y agregación, dentro de la programación orientada a objetos. [1,0 puntos]

- **Herencia.** La herencia define las relaciones entre clases en un lenguaje orientado a objetos, donde es posible diseñar nuevas clases basándose en clases ya existentes: esto es lo que se llama Herencia. Cuando una clase hereda de otra, toma todos los atributos y todos los métodos de su clase “madre” o superclase, y puede añadir los suyos propios. Esto ocurre porque a veces, algunos de los métodos o datos heredados no son útiles, por lo que pueden ser enmascarados, redefinidos o simplemente eliminados en la nueva clase. Las clases derivadas de otras se conocen como clases hijas o subclases.

Podemos encontrar dos tipos de herencia en los lenguajes de programación:

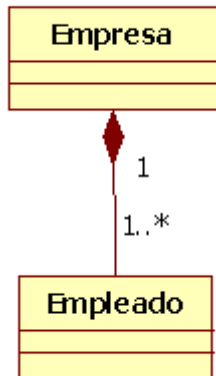
***Simple:** una clase puede derivar únicamente de otra clase (sólo hereda de una superclase).

***Múltiple:** una clase puede heredar comportamientos y características de más de una superclase.

- **Composición.** Es una forma de asociación en la que un objeto forma parte integral de otro, es decir, que la composición es un tipo de relación **dependiente** en donde un objeto más complejo es conformado por objetos más pequeños. Esto implica que el objeto formado (u objeto contenedor) tiene validez mientras los objetos componentes también la tengan, por lo tanto, el objeto contenido desaparecerá si el objeto que los contiene desaparece. Por ejemplo, los departamentos de una pyme (contabilidad, marketing y ventas) mantienen una relación de “composición” con dicha empresa, de

modo que si la pequeña empresa desaparece, sus departamentos también desaparecerán, pues no tienen “vida propia” fuera de ella.

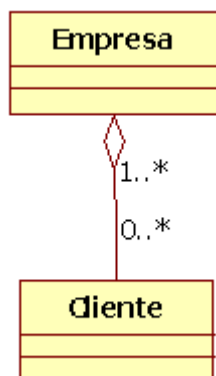
El símbolo de composición es un diamante de color negro colocado en el extremo en el que está la clase que representa el “todo” (Compuesto):



• Tenemos una clase Empresa. • Un objeto Empresa está a su vez compuesto por uno o varios objetos del tipo empleado. • El tiempo de vida de los objetos Empleado depende del tiempo de vida de Empresa, ya que si no existe una Empresa no pueden existir sus empleados.

- Agregación. En la agregación también tenemos objetos contenidos y objetos que los contienen, pero la relación es parametrizada y su existencia, o inexistencia, no les afecta. Si el objeto contenedor desaparece, sus componentes siguen existiendo. Habitualmente se da con mayor frecuencia que la composición.

La agregación se representa en UML mediante un diamante de color blanco colocado en el extremo en el que está la clase que representa el “todo”.



• Tenemos una clase Empresa. • Tenemos una clase Cliente. • Una empresa agrupa a varios clientes. Si la empresa desaparece, los clientes seguirán existiendo y buscarán otra empresa en la que realizar sus compras de productos o servicios.