ENTORNOS DE DESARROLLO – UF3 (ACTIVIDAD A)

Los diagramas se agrupan en dos grandes grupos, ¿Cuáles son? Indícalos y defínelos.

En el terreno de los diagramas debemos tener muy presente el UML (Unified Modeling Language o Lenguaje Unificado de Modelado). Se trata de una serie de estándares y normas que marcan las pautas de como se deben representar los esquemas, diagramas o la documentación relativas al software. Se trata de una herramienta usada internacionalmente por personas con conocimientos avanzados en programación como los analistas funcionales o los analistas programadores.

El lenguaje UML ofrece hasta 14 tipos de diagramas que se clasifican en dos grandes grupos:

* **Diagramas estructurales.** Presentan modelos estáticos del objeto, como clases, paquetes o componentes, aunque también pueden ser utilizados como modelos en tiempo de ejecución.
* **Diagramas de comportamiento.** Son los que muestran la forma que tienen de reaccionar ante los estímulos externos. Son diagramas dinámicos, basados en su devenir en tiempo de ejecución.

¿Qué es un diagrama de comportamiento? ¿y para qué se utiliza?

Se trata de un tipo de diagrama que define las secuencias de estados por los que pasa un objeto durante su ciclo de vida. Se muestran los diferentes estados y, de forma gráfica, los procesos a programar. Se trata de diagramas dinámicos, que se basan en el comportamiento en tiempo de ejecución. Estos diagramas se usan para visualizar y especificar, a la vez que documentar, aspectos dinámicos de un sistema. Estos diagramas dependen su complejidad al proceso que representan.

¿En cuántos grupos se dividen los Diagramas de comportamiento? Menciónalos y descríbelos.

Los diagramas de comportamiento se dividen en cuatro grupos:

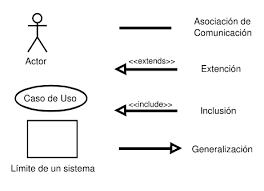
* **Diagrama de actividades.** Muestra la secuencia y las condiciones para coordinar los comportamientos de nivel inferior, en lugar de los clasificadores que poseen esos comportamientos. Estos son comúnmente llamados modelos de flujo de control y flujo de objetos.
* **Diagrama de casos de uso.** Describe un conjunto de acciones que algunos sistemas deben o pueden realizar en colaboración con uno o más usuarios externos del sistema para proporcionar algunos resultados observables y valiosos a los actores u otros interesados del sistema.
* **Diagrama de máquina de estados.** Se utiliza para modelar el comportamiento discreto a través de transiciones de estados finitos. Además de expresar el comportamiento de una parte del sistema, las máquinas de estado también se pueden usar para expresar el protocolo de uso de parte de un sistema.
* **Diagramas de interacción.** Se trata de un tipo de diagrama de comportamiento que nos lleva a otros tipos de diagramas:
  + **Diagrama de secuencia.** Es el tipo más utilizados de los de interacción y se centra en el intercambio de mensajes entre líneas de vida.
  + **Diagrama de comunicación.** Se enfoca en la interacción entre líneas de vida donde la arquitectura de la estructura interna y cómo esto se corresponde con el paso del mensaje es fundamental. La secuencia de mensajes se da a través de una numeración.
  + **Diagrama de tiempos.** Se centran en las condiciones que cambian dentro y entre las líneas de vida a lo largo de un eje de tiempo lineal.
  + **Diagrama global de interacciones.** Estos brindan una descripción general del flujo de control donde los nodos del flujo son interacciones o usos de interacción.

¿Qué representan los Diagramas de Caso de uso? ¿Qué son los actores dentro de los Diagramas de casos de uso y cómo se representan? Ejemplifícalo

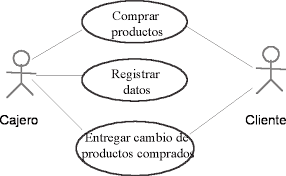
Los diagramas de casos de uso representan un conjunto de escenarios que tienen una meta de usuario común. Se trata de una descripción de un proceso, relativamente largo, que incluye varias etapas o transacciones. Cuando hablamos de los diagramas de caso de uso nos referimos al uso particular que se hace del sistema, desencadenada por el estímulo que le produce un “actor o rol” externo.

Un actor representa el rol jugado por una persona o cosa que actúa con el sistema. Cuando nos referimos a un actor lo hacemos con la finalidad del rol y no como si de una persona particular se tratase. Así que, tanto actor como rol, son lo mismo y expresan la labor que se realiza frente al sistema.

Los elementos que aparecen en un diagrama de caso de uso son:



A continuación, muestro un ejemplo de diagrama de caso de uso:



¿Qué es un Diagrama de Secuencia? Y ¿Cuáles son los elementos que conforman un Diagrama de Secuencia?

Se trata de un tipo de diagrama de interacción. Su objetivo es representar el intercambio de mensajes que se pueden dar entre los objetos de un sistema, definiendo cual será su comportamiento en ese contexto. Suele usarse junto al diagrama de casos de uso porque tiene relación directa al funcionamiento de un caso de uso. También se usa para comprender mejor el diagrama de clases ya que, el diagrama de secuencia, muestra como los objetos de esas clases interactúan intercambiándose mensajes.

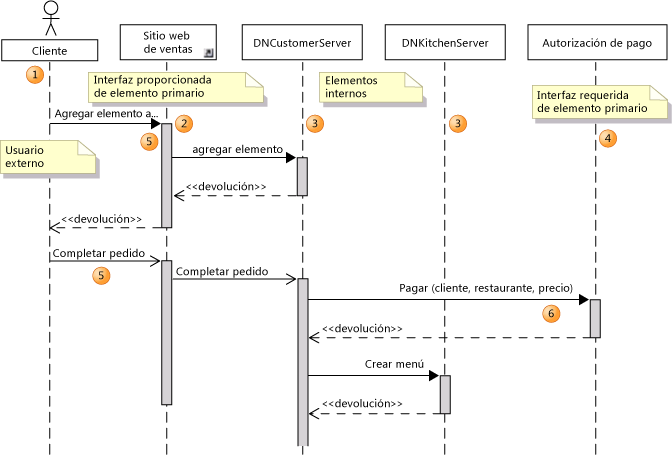
Un diagrama de secuencia se construye a partir de dos dimensiones:

* HORIZONTAL: Donde se representan los objetos que van a participar en la secuencia.
* VERTICAL: Donde se representa la línea del tiempo sobre la que van a actuar los elementos y su flujo de ejecución va de arriba abajo.

Los diagramas de secuencia están compuestos por:

* **Objetos.** Representados como rectángulos con un texto subrayado. Se colocan en la parte superior del diagrama.
* **Mensajes.** Se representa con una línea continua que termina con una flecha. Hay tres tipos:
  + **Mensaje simple.** Transferencia de un control a otro.
  + **Mensaje sincrono.** Cuando se necesita una respuesta antes de continuar.
  + **Mensaje asíncrono.** Cuando no necesita una respuesta para poder continuar.
* **Tiempo.** Representado por una progresión vertical.

A continuación, muestro un ejemplo de como sería un diagrama de secuencia con sus elementos:



¿Qué es un Diagrama de Colaboración? Y ¿Cuáles son los elementos que conforman un Diagrama de Colaboración?

Un diagrama de colaboración es una forma de representar interacción entre objetos, mostrando las relaciones y los mensajes que se envían los diferentes objetos que puedan intervenir. Dichos objetos están conectados por enlaces en los que se representan los mensajes enviados acompañados por una flecha que es la encargada de indicar la dirección. Con este tipo de diagramas se obtiene una visión más amplia a la hora de comprender la participación de un objeto en un sistema.

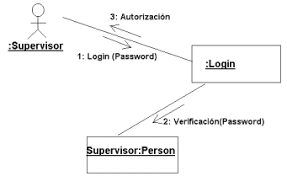
Los diagramas de colaboración permiten elegir el orden en que pueden hacerse las cosas y nos dejan describir procesos o casos de uso, nos muestran los aspectos dinámicos de un sistema, establece las reglas de secuencia a seguir y ayuda al programador a desarrollar código a través de una descripción lógica de un proceso.

Los elementos que intervienen en los diagramas de colaboración son:

* **Objeto.** Representados de forma usual, con un rectángulo que contiene el nombre y la clase del objeto.
* **Enlaces.** Se representan con una línea continua que une a dos objetos.
* **Mensaje.** Se representa con una pequeña flecha colocada junto a la línea del vínculo al que se asocia.

Una particularidad que diferencia los diagramas de colaboración con los de secuencia es que los primeros muestran un número de secuencia del mensaje delante de su nombre.

Un ejemplo de representación de diagrama de colaboración sería:



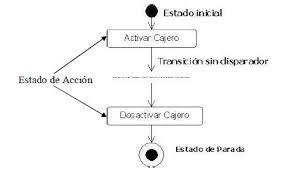
¿Qué es un Diagrama de Actividades? Y ¿Cuáles son los elementos que conforman un Diagrama de Actividades?

Un diagrama de actividades muestra el flujo de control, haciendo énfasis en su secuencia y sus condiciones. Se utilizan para representar la forma en la que un sistema hace una implementación. Los diagramas de actividades muestran un flujo de trabajo que va desde un punto inicial a uno final.

Este tipo de diagramas cuentan con los siguientes elementos:

* **Actividad.** Se trata de una conducta que se representa como un flujo de acciones. Se representa con un rectángulo con los bordes redondeados que incluye el nombre de la actividad en su interior
* **Flujo entre actividades.** Se trata de una clase abstracta usada para representar las conexiones dirigidas a lo largo de las cuales los tokens y objetos fluyen entre los nodos de actividad. Incluye flujos de control y flujos de objetos. La fuente y el objetivo de un borde deben estar en la misma actividad que el borde. Se representan con una flecha que simboliza el orden de ejecución.
* **Nodo inicial.** El nodo inicial marca el control en el que se inicia el flujo cuando comienza la actividad. Existirá uno por diagrama pero cada actividad podrán tener más de un nodo inicial; en este caso, cuando se inicia la actividad, se iniciará un flujo por cada nodo, como si de un thread se tratase. Los nodos iniciales se representan como un círculo pequeño relleno (como un lunar).
* **Nodo final.** Este nodo se encarga de detener todos los flujos existentes en una actividad. La representación de estos nodos se hace por medio de un círculo pequeño relleno (como un nodo inicial) rodeado por un círculo hueco.

Un ejemplo de representación de un diagrama de actividades sería:



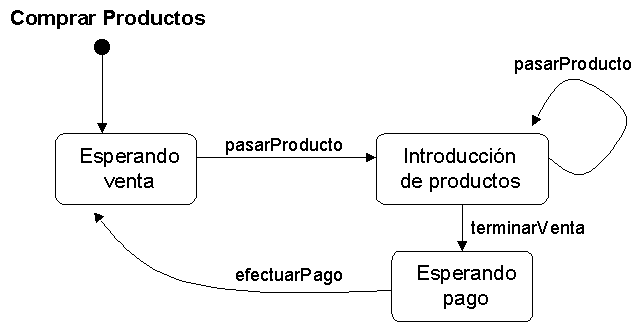
¿Qué es un diagrama de Estado? Y ¿Cuáles son los elementos que conforman un Diagrama de Estado?

Son los tipos de diagramas que se emplean para representar el comportamiento del objeto de una clase. En él se representa toda la secuencia de estados por los que dicho objeto pasará durante su ciclo de vida, siguiendo el camino que marcan las acciones que va realizando. Cuando hablamos de un estado, nos referimos a la representación de los sucesos que expresa un objeto durante los diferentes estados por los que pasa, empieza en un estado de inicio y acaba en un estado final. Cada uno de los eventos representados expresan algo que puede provocar cambios en nuestro objeto. Existen unas líneas, a las que llamamos líneas de transición, cuya finalidad es describir el movimiento que hace el objeto cuando va de un estado a otro; a estas líneas le ponemos el nombre de evento.

Los elementos con los que contamos para la representación de este tipo de diagramas son:

* **Pseudoestado de opción.** Es un símbolo con forma de diamante indicando una condición dinámica con resultados potenciales ramificados.
* **Punto de salida.** Se trata del punto de donde se sale de un estado compuesto, o de una máquina de estados. Se representa con un círculo que lleva una “X” dentro.
* **Evento.** Anteriormente descrito pero lo podemos resumir diciendo que es lo que activa una transición.
* **Estado final.** Se representa con un círculo negro con una flecha de transición y expresa lo que su propio título ofrece.
* **Protección.** Es una condición booleana que permite o detiene una transición.
* **Estado.** Muestra la situación en la que se encuentra un objeto y se representa con un rectángulo redondeado.
* **Subestado.** Es un estado contenido dentro de la región de un estado compuesto.
* **Disparador.** Es un mensaje que mueve un objeto de estado en estado.
* **Transición.** Representada con una flecha que corre de un estado a otro e indica un estado cambiante.
* **Estado compuesto.** Es un estado que contiene subestados.
* **Comportamiento de transición.** Es el comportamiento resultante que ocurre durante la transición de un estado. Se escribe arriba de la flecha de transición.

Un ejemplo de diagrama de estado sería:



En la programación orientada a objetos existen los siguientes conceptos: clase, atributo, método y visibilidad. Defínelos.

* **CLASE.** Se trata de una especie de plantilla creada para que sirva de punto de partida para crear objetos, como un “molde”. Las clases se utilizan para representar una entidad o un concepto y es una de las características principales y fundamentales que nos aporta la programación orientada a objetos. Cada clase se define gracias a variables y funciones, que son los que forman los atributos y los métodos.
* **ATRIBUTO.** Los atributos son las características individuales que posee un objeto y se expresan por medio de variables. Cuando se hacen instancias de una clase que tiene definidos los atributos, dichos objetos podrán darle un valor propio, estas son las denominadas variables de instancias; también existen la variables de clase, que son las que aplican su valor a la clase y a sus instancias.
* **MÉTODO.** Un método es una función que aporta un conjunto de instrucciones que se ejecutarán al ser invocado. Cada método deberá llevar un nombre propio que será el que servirá al programador para identificarlo y ejecutarlo. Los métodos podrán requerir datos para proceder a su acceso, estos datos en forma de variables irán expresados dentro de unos paréntesis, justo a continuación de su nombre. También cabe destacar que estas funciones podrán devolver un dato en el punto del programa donde se hace la llamada o bien no devolverán nada (de tipo “void”).
* **VISIBILIDAD.** Esta es la cualidad que marca el tipo de acceso que tendrá una propiedad o un método. Dependiendo de cada lenguaje de programación, esta visibilidad se podrá declarar de diferentes maneras, usando palabras reservadas o definiendo algún prefijo en el nombre de la variable. Por ejemplo, en Java usaremos:
  + **public.** Permite el acceso desde cualquier parte de la aplicación.
  + **private.** Sólo se podrá acceder desde la clase que lo contiene.
  + **protected.** Permite el acceso dentro del mismo paquete de la clase, fuera de este o para las clases que heredan.
  + **(si no se declara ninguna visibilidad).** Cuando se da este supuesto, su visibilidad será dentro del paquete que contiene esta clase.

Define con tus propias palabras las relaciones: herencia, composición y agregación, dentro de la programación orientada a objetos.

* **HERENCIA.** Se trata de una de las principales características de la programación orientada a objetos. Se trata de un mecanismo que nos permite extender el código de una clase a otras clases. La clase principal, de la que heredarán otras, se le denomina clase padre y será la superclase; la clase que recibe dicha herencia es la clase hijo y será la subclase. Cuando una subclase hereda de una superclase tendrá acceso a todos los métodos y atributos de esta e incluso podrá sobrescribir en ese código afectándole, únicamente, a esta subclase.
* **COMPOSICIÓN.** Este es el mecanismo en el que una clase se construye a partir de otros objetos de igual o distinto tipo, pudiéndolos combinar para obtener la funcionalidad deseada. Mientras que en la herencia hablamos que la relación que tienen los objetos “conectados” sería “es un”, en la composición la relación sería “usa/tiene un” con aquellos objetos que se hacen a partir de dicha clase.
* **AGREGACIÓN.** Cuando hablamos de agregación dentro de la programación orientada a objetos nos referimos a la asociación que indica que una clase es parte de otra clase y sus componentes podrán ser compartidos.