Tarea 1

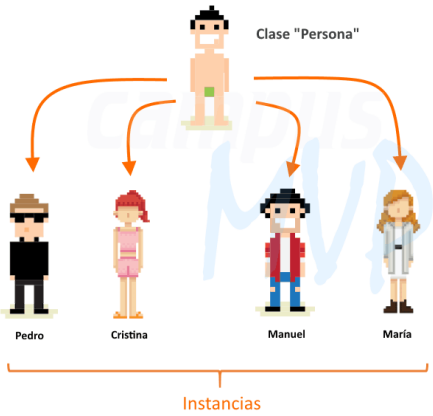
Realiza un resumen del Análisis de la programación visual

Conceptos de programación orientada a objetos.

Clases, Objetos e Instancias.

Una clase es una plantilla. Define de manera genérica cómo van a ser lo objetos de determinado tipo. Debe de tener una serie de atributos y una serie de comportamientos que se implementan como métodos de la clase. Una clase por sí sola no sirve de nada, pues no es más que un concepto, sin entidad real.

Para poder utilizar una clase en un programa lo que hay que hacer es instanciarla. Instanciar una clase consiste en crear un nuevo objeto concreto de la misma. un objeto es ya una entidad concreta que se crea a partir de la plantilla que es la clase. Este nuevo objeto tiene ya "existencia" real, puesto que ocupa memoria y se puede utilizar en el programa.



Encapsulación

Permite que todo lo referente a un objeto quede aislado dentro de éste. Es decir, que todos los datos referentes a un objeto queden "encerrados" dentro de éste y sólo se puede acceder a ellos a través de los miembros que la clase proporcione.

Así, internamente tenemos un dato que es el nombre de la persona y accedemos a él a través de la propiedad pública Nombre que define la clase que representa a las personas. De este modo damos acceso sólo a lo que nos interese y del modo que nos interese.

Abstracción

Nos abstrae de la complejidad que haya dentro dándonos una serie de atributos y comportamientos que podemos usar sin preocuparnos de qué pasa por dentro cuando lo hagamos. Así, una clase debe exponer para su uso solo lo que sea necesario. Cómo se haga "por dentro" es irrelevante para los programas que hagan uso de los objetos de esa clase.

Herencia

Una clase que hereda de otra obtiene todos los rasgos de la primera y añade otros nuevos y además también puede modificar algunos de los que ha heredado.

A la clase de la que se hereda se le llama clase base, y a la clase que hereda de ésta se le llama clase derivada. Todos los objetos de estas clases heredan las propiedades y los métodos. Lo bueno de la herencia es que podemos reutilizar todo lo que tuviésemos en la clase base.

Polimorfismo

Se refiere al hecho de que varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta, sin tener que saber de qué clase exacta son para poder hacerlo.

El polimorfismo nos permite utilizar a los objetos de manera genérica, aunque internamente se comporten según su variedad específica.

El polimorfismo puede ser más complicado que eso ya que se puede dar también mediante la sobrecarga de métodos y, sobre todo, a través del uso de interfaces, pero el concepto es el que acabo de explicar.

Características y aplicaciones de eventos.

Modelo de la programación de computadoras, donde se utilizan los eventos que suceden para la determinación del flujo de control de un programa.

No es un tipo de tecnología o lenguaje de programación, sino un enfoque que se implementa durante la etapa de desarrollo del producto.

Básicamente, separa la lógica de procesamiento de eventos del resto del código de un programa.

Usando un procedimiento apropiado de manejo de eventos para tratarlos, normalmente mediante una llamada a una función o método.

Teóricamente, el estilo de esta programación es compatible con todos los lenguajes de programación, aunque puede ser diferente en la forma de implementarse.

En general, en una aplicación controlada por eventos hay un bucle principal que “escucha” los nuevos eventos entrantes, activando una llamada a una función cuando estos se detectan. Por tanto, su funcionamiento se enfoca en los eventos, decidiendo estos qué ejecutar y en qué orden.

Dependencia de eventos

Los eventos en sí pueden ser desde aceptar o rechazar una solicitud de préstamo, denominado evento de alto nivel, hasta que un usuario presione una tecla, que es un evento de bajo nivel.

Orientada al servicio

Se utiliza para escribir programas diseñados para el servicio sin ralentizar la computadora, ya que la orientación al servicio solo consume poco poder de procesamiento. Además, los servicios se ejecutan por lo general en el trasfondo del sistema operativo.

Eventos

Condición que surge durante la ejecución de un programa y que requiere alguna acción por parte del sistema. Cada evento es diferente por naturaleza, algunos requieren que el programa recobre y muestre cierta información, y otros que se inicien algunos cálculos y cambios de estado.

Los eventos incluyen al ratón, al teclado, una interfaz de usuario y las acciones que se deben activar en el programa cuando ocurran. Esto significa que el usuario debe interactuar con un objeto en el programa, como hacer clic en un botón del ratón, usar el teclado para seleccionar un botón.

Controlador de eventos

Es un tipo de función o método que ejecuta una acción específica cuando se activa un evento determinado.

Por ejemplo, podría ser un botón que cuando el usuario haga clic en él muestre un mensaje y cuando vuelva a hacer clic en ese botón cierre el mensaje.

Funciones de activación

Son funciones que deciden qué código ejecutar cuando se produce un evento específico. Se utilizan para seleccionar qué controlador de eventos emplear al producirse un evento.

Tiempo controlado

Es una tarea preestablecida por hacer.

La actualización de Windows es un ejemplo de tiempo controlado, donde el usuario puede establecer cuándo actualizar o cuándo verificar y descargar la actualización.

Características de componentes y métodos visuales y no visuales.

Componente visual

Un componente es visual cuando tiene una representación gráfica en tiempo de diseño y ejecución (botones, barras de scroll, cuadros de edición, etc.).

los componentes visuales podemos dividirlos a su vez en dos tipos:

Componentes interactivos: permiten que el usuario final los manipule, ya sea introduciendo datos, seleccionado elementos, etc. De forma que estos componentes pueden recibir el foco, así como los eventos propios del teclado y del ratón.

Normalmente, el propio sistema operativo es el encargado de dibujar el aspecto del componente, haciendo el componente las llamadas correspondientes para que este aspecto cambie.

Componentes gráficos: el propio componente es el encargado de dibujar en la pantalla lo que crea oportuno, bien a través de las funciones básicas de API de Windows o bien a través de otras librerías gráficas.

Estos componentes, no suelen recibir eventos del usuario final, aunque si eventos del propio programador, ya que su cometido no suele ir más allá de mostrar ciertos gráficos o imágenes en la pantalla.

Dado que un componente es un objeto como otro cualquiera, podremos aplicar en él todas las técnicas de la orientación a objetos: encapsulación, herencia y polimorfismo.

La técnica de la herencia, aplicada a los componentes, nos permite personalizar cualquier componente, o porque queremos ampliar las posibilidades del componente.

Componentes no visuales

Los componentes no visuales se pueden colocar en los formularios de la misma manera que los controles, aunque en este caso su posición es irrelevante.

Los componentes no-visuales deben heredar directamente de TComponent, ya que este proporciona las características básicas.

TControl: se trata de la clase padre para todos los componentes visuales, ya sean gráficos o no.

Los componentes no visuales incluyen Time Control, SerialPort y ServiceController, entre otros.

Procesos de desarrollo visual en proyectos distribuidos y de escritorio.

¿Qué es gestión visual?

Conjunto de técnicas orientadas a transmitir información:

- De manera ordenada
- Preferiblemente de manera visual (iconos, símbolos, gráficos)
- De forma auto explicativa
- Que promueva la toma de decisiones y la mejora continua

El objetivo principal es gestionar nuestro trabajo, definiendo:

- Como estamos ahora
- Que hay que hacer
- Quien lo va a hacer
- En qué orden se van a hacer

Y todo aquello de manera dinámica.

Para potenciar el trabajo en equipo:

- Facilita el conocimiento por parte de todo el equipo de trabajo
- Genera un mayor sentido de pertenencia en el proyecto
- Proporciona una oportunidad para que unos miembros informen a otros
- Permite obtener el feedback de la información

Para facilitar la gestión de conocimiento

- Reduce los silos de la información mas importante
- Disminuye los problemas producidos por a falta de comunicación
- Muestra el trabajo en progreso del proyecto
- Ayuda a identificar
    - Riesgos en el proyecto
    - Defectos en procesos o en entregables
    - Detectar cuellos de botella y bloqueos

Requerimientos visuales de proyectos distribuidos y de escritorio

Se pueden utilizar otros lenguajes de modelado como UML para especificar

Los diagramas de UML más utilizados en el modelado de sistemas distribuidos son:

Diagramas de caso de uso, de secuencia, de estado, de actividades, de componente y de despliegue.

Tableros físicos:

Empleado en murales, pizarras, paredes o ventanales.

Es importante lograr una lectura cómoda a 2-3 metros de distancia

Resulta habitual emplear combinación de graficas impresas y material elaborado a mano.

Estratégicamente situado, facilita la visión de todo el equipo

Fomenta las conversaciones espontaneas sobre el estado del proyecto

El material empleado facilita la creatividad y el pragmatismo

Resulta muy fácil de poner en marcha

Tableros virtuales:

Abordable como un tablero Kanban o como un dashboard

Es muy importante la usabilidad, preferible acceso web y responsive desing

La gestión de archivos adjuntos, foros y discusiones potencial en el uso tradicional de tarjetas

No requieres espacios físicos o ubicaciones concretas

Permite consultas en cualquier miento y lugar

Ideal para equipos distribuidos o con alta movilidad

Resulta más fácil actualizar que un tablero físico

Kanban:

Tablero de gestión de tareas, organizadas por columnas

Cada tarea en una nota o tarjeta

Cada estado en una columna

Posibilidad de emplear codificación de colores o divisiones horizontales par mejorar su estructura.

Herramientas y lenguajes de programación visual.

En la programación visual se emplean bloques gráficos con diferentes colores y formas geométricas que pueden acoplarse o enlazarse intuitivamente del mismo modo que un puzle o un juego de construcción.

La programación visual es un lenguaje de programación que permite a los humanos describir procesos mediante ilustraciones. Mientras que un lenguaje de programación típico basado en texto hace que el programador piense como un ordenador, un lenguaje de programación visual permite al programador describir el proceso en términos que tengan sentido para los humanos.

El enfoque gráfico en la programación permitirá al ordenador procesar las representaciones espaciales en dos o más dimensiones. Esta programación gráfica se denomina Lenguaje de Programación Visual en el que se definen imágenes mediante un lenguaje basado en texto.

Se utiliza con frecuencia en los diseños de sistemas de ingeniería para traducir la información de los eventos y las vibraciones durante las pruebas de los motores de automóviles en una lectura visual.

Las herramientas de programación gráfica se utilizan para emplear los diagramas de bloques, los instrumentos virtuales y los conectores. También puede utilizarse para supervisar y controlar el proceso automatizado. Además, puede analizar algunas funciones matemáticas como el uso de procesamiento insignificante. Permitirá también a los usuarios tener acceso a bases de datos de información sobre su terreno, edificios y demografía o utilizarlo en el diseño de sistemas celulares.

Scratch (MIT): Permite el desarrollo de aplicaciones interactivas a modo de aprendizaje como pequeños juegos, etc. También permite el desarrollo de aplicaciones para controlar dispositivos robóticos Lego

Bolt (Ludiq): Plugin de Unity incluido gratuitamente a partir de la versión 2019 para el desarrollo visual de scripts como alternativa a C#

ArduinoBlocks (Didactronica): Permite el desarrollo de scripts para ejecutarse en placas de desarrollo Arduino de modo alternativo a C++

MakeCode (Microsoft): Plaraforma visual de desarrollo para múltiples micro-dispositivos robóticos de modo alternativo a Javascript / Python.

AppInventor (MIT): Plataforma online que permite el desarrollo visual de aplicaciones para dispositivos móviles.