PROGRAMACION II MARK DIG

LECTURAS SESION DIEZ

L1

Qué es Polimorfismo

En programación orientada a objetos el polimorfismo se refiere a la posibilidad de definir clases diferentes que tienen métodos o atributos denominados de forma idéntica, pero que se comportan de manera distinta. El concepto de polimorfismo se puede aplicar tanto a funciones como a tipos de datos. Así nacen los conceptos de funciones polimórficas y tipos polimórficos. Las primeras son aquellas funciones que pueden evaluarse o ser aplicadas a diferentes tipos de datos de forma indistinta; los tipos polimórficos, por su parte, son aquellos tipos de datos que contienen al menos un elemento cuyo tipo no está especificado.

Tipos de polimorfismo

Polimorfismo de sobrecarga

El polimorfismo de sobrecarga ocurre cuando las funciones del mismo nombre existen, con funcionalidad similar, en clases que son completamente independientes una de otra (éstas no tienen que ser clases secundarias de la clase objeto). Por ejemplo, la clase complex, la clase image y la clase link pueden todas tener la función "display". Esto significa que no necesitamos preocuparnos sobre el tipo de objeto con el que estamos trabajando si todo lo que deseamos es verlo en la pantalla.

Por lo tanto, el polimorfismo de sobrecarga nos permite definir operadores cuyos comportamientos varían de acuerdo a los parámetros que se les aplican. Así es posible, por ejemplo, agregar el operador + y hacer que se comporte de manera distinta cuando está haciendo referencia a una operación entre dos números enteros (suma) o bien cuando se encuentra entre dos cadenas de caracteres (concatenación).

Polimorfismo paramétrico (también llamado polimorfismo de plantillas)

El polimorfismo paramétrico es la capacidad para definir varias funciones utilizando el mismo nombre, pero usando parámetros diferentes (nombre y/o tipo). El polimorfismo paramétrico selecciona automáticamente el método correcto a aplicar en función del tipo de datos pasados en el parámetro.

Por lo tanto, podemos por ejemplo, definir varios métodos homónimos de addition() efectuando una suma de valores.

*El método int addition (int,int) devolvería la suma de dos números enteros.
*El método float addition (float, float) devolvería la suma de dos flotantes.
*El método char addition (char, char) daría por resultado la suma de dos caracteres definidos por el autor.

Una signature es el nombre y tipo (estático) que se da a los argumentos de una función. Por esto, una firma de método determina qué elemento se va a llamar.

Polimorfismo de inclusión (también llamado redefinición o subtipado)

La habilidad para redefinir un método en clases que se hereda de una clase base se llama especialización. Por lo tanto, se puede llamar un método de objeto sin tener que conocer su tipo intrínseco: esto es polimorfismo de subtipado. Permite no tomar en cuenta detalles de las clases especializadas de una familia de objetos, enmascarándolos con una interfaz común (siendo esta la clase básica).

Ejemplo de polimorfismo

En este ejemplo haremos uso del lenguaje C++ para mostrar el polimorfismo. También se hará uso de las funciones virtuales puras de este lenguaje, aunque para que el polimorfismo funcione no es necesario que las funciones sean virtuales puras, es decir, perfectamente el código de la clase "superior" (en nuestro caso Empleado) podría tener código

```
#include<stdio.h>;
#include<conio.h>;
#include<iostream>;
using namespace std;

class figuras {
  public:
  float base;
  float altura;

  public:
  float captura();
  virtual unsigned float perimetro()=0;
  virtual unsigned float area()=0;
};

class rectangulo: public figura{
  public:
```

```
  void imprime();
  unsigned float perimetro(){return 2*(base+altura);}
  unsigned float area(){return base*altura;}
};

class triangulo: public figura{
  public:
  void muestra();
  unsigned float perimetro(){2*altura+base}
  unsigned float area(){return (base*altura)/2;}
};

void figura::captura(){
    cout<<"CALCULO DEL AREA Y PERIMETRO DE UN TRIANGULO ISOCELES Y UN
RECTANGULO:" <<endl;
    cout<<"escribe la altura: ";
    cin>>altura;
    cout>>"escribe la base: ";
    cin>>base;
};
```

Polimorfismo con interface

En este caso haremos que "moderador" se pueda referir tanto a "Zguillez" como a "Zah". Definiremos una Interface que contendrá las funciones que estamos delegando a la clase asociada.

```
package
{public interface IModerador
{function moderar():void;
}
}
```

Diferencias entre Polimorfismos y Sobrecarga

- El polimorfismo, suele ser bastante ventajoso aplicado desde las interfaces, ya que permite crear nuevos tipos sin necesidad de tocar las clases ya existentes (imaginemos que deseamos añadir una clase Multiplicar), basta con recompilar

todo el código que incluye los nuevos tipos añadidos. Si se hubiera recurrido a la sobrecarga durante el diseño exigiría retocar la clase anteriormente creada al añadir la nueva operación Multiplicar, lo que además podría suponer revisar todo el código donde se instancia a la clase.

- **La sobrecarga se da siempre dentro de una sola clase, mientras que el polimorfismo se da entre clases distintas.**

- Un método está sobrecargado si dentro de una clase existen dos o más declaraciones de dicho método con el mismo nombre pero con parámetros distintos, por lo que no hay que confundirlo con polimorfismo.

- En definitiva: La sobrecarga se resuelve en tiempo de compilación utilizando los nombres de los métodos y los tipos de sus parámetros; el polimorfismo se resuelve en tiempo de ejecución del programa, esto es, mientras se ejecuta, en función de que clase pertenece un objeto.

L2

Gestión de Errores y Excepciones

ÁREA SEG_ERRORES

Uno de los focos que originan vulnerabilidades en nuestras aplicaciones se basa en la falta de control sobre los errores que se producen en su ejecución y el tratamiento correcto de las excepciones. Veremos cómo disminuir los riesgos de ser atacados a partir de la generación de un error o excepción en la aplicación.

El manejo de errores y excepciones son dos aspectos para realizar un seguimiento de fallos dentro de una aplicación. En términos generales, hay tres situaciones que hacen que las excepciones sean lanzadas:

- **Excepciones que se producen en el código del cliente**: El código cliente intenta hacer algo que no está permitido por la API, de esta forma viola el contrato. El cliente puede tomar algún camino alternativo, si hay información útil proporcionada en la excepción. Por ejemplo: una excepción es lanzada cuando se está analizando un documento XML que no está bien formado. La excepción contiene información útil acerca de la localización en el documento XML donde se produce el problema. El cliente puede utilizar esta información para recuperarse del problema.
- **Excepciones por fallos en los recursos mantenidos:** Las excepciones se producen si existe un fallo derivado de un recurso. Por ejemplo, el agotamiento de memoria o el fallo de conexión cuando se cae una red. La respuesta del cliente a los recursos que fallan depende del contexto. El cliente puede reintentar la operación después de algún tiempo o simplemente registrar el fallo del recurso y detener la aplicación.
- **Excepciones por errores derivados del código programado:** En esta categoría, las excepciones se producen en la ejecución del código. El código cliente usualmente no puede hacer nada con estos errores.

L3

La infraestructura de integración soporta diversos formatos de mensaje, protocolos para intercambiar mensajes y tanto el procesamiento de mensajes síncrono como

asíncrono. La gestión de errores requiere varias opciones para satisfacer las diversas configuraciones de implementación que se pueden elegir.
La infraestructura de integración utiliza colas JMS como mecanismo de transferencia para los mensajes entrantes y salientes. **La gestión de errores de cola se inicia cuando se identifica una condición de error y se pueden ver, corregir, cancelar y reprocesar mensajes problemáticos.**

- **Gestión de errores que no son de cola**

  Al iniciar el procesamiento síncrono de mensajes de integración entrantes o salientes, se le notificará cualquier error en el momento de la ejecución.

- **Gestión de errores basada en cola**

  Puede utilizar la aplicación Nuevo proceso de mensajes para gestionar mensajes de integración asíncronos entrantes y salientes erróneos que utilizan colas JMS.

- **Configuración de la gestión de errores**

  Para configurar la gestión de errores, debe configurar propiedades del sistema y configurar el sistema externo.

- **Notificación de errores**

  Cuando una transacción entrante o saliente causa un error en una cola, se envía una notificación por correo electrónico al administrador del sistema sólo si hay otros errores no resueltos esperando en la misma cola. Si existen varios errores en la cola, el administrador del sistema debe resolverlos todos antes de que se envíe la notificación de nuevos errores.

- **Nuevo proceso de mensajes**

  En la aplicación Nuevo proceso de mensajes, puede gestionar mensajes marcados con un error realizando acciones como cambiar el estado del mensaje, corregir el mensaje o suprimirlo de la base de datos.

- **Gestión de errores con importación de datos basada en archivo**

  La infraestructura de integración soporta la gestión de errores con la aplicación Nuevo proceso de mensajes, que permite examinar, corregir, reprocesar y suprimir mensajes que producen errores cuando se procesan desde una cola entrante. Cuando los datos se importan de un archivo XML o un archivo sin formato, hay disponible una segunda opción que gestiona los errores utilizando un archivo descargado en lugar de la aplicación Nuevo proceso de mensajes.

- **Gestión de errores de tablas de interfaz**

La tabla de interfaz se puede utilizar para el intercambio de datos entrantes y salientes. Se pueden producir errores al grabar datos entrantes de una tabla de interfaz a una cola o datos salientes de una cola a una tabla de interfaz.

- **Causas habituales de errores**

  Los errores durante el proceso de mensajes en la cola saliente se dan cuando existe un problema al entregar un mensaje al punto final especificado para el sistema externo. Normalmente, los errores durante el proceso de mensajes en una cola entrante están relacionados con una validación de regla de negocio o en el proceso entrante del servicio empresarial.

- **Búsqueda de errores**

  Cuando reciba una notificación de error, busque el mensaje XML de la aplicación Nuevo proceso de mensajes. En función del tipo de cola (secuencial o continua), el número de mensajes de la cola puede ser cero, uno o más en la aplicación Nuevo proceso de mensajes para una cola individual.

- **Seguimiento de mensajes**

  La aplicación Seguimiento de mensajes realiza un seguimiento y visualiza el historial de procesamiento de mensajes de canal de publicación y de mensajes de servicios empresariales basados en cola.