Abstracción

De una clase abstracta se pueden crear referencias a objetos de la clase.

Destacamos 3 Niveles de Abstracción.

En Java, si un método no abstracto f() definido en una superclase se sobrescribe en una de sus subclases, se puede invocar el método de la superclase desde la subclase mediante la instrucción super.f().

La abstracción es una supresión intencionada (u ocultación) de algunos detalles de un proceso o artefacto, con el fin de destacar más claramente otros aspectos, detalles o estructuras.

La clase abstracta se caracteriza por declarar al menos un método abstracto.

Los métodos abstractos son métodos con enlace dinámico.

Ser abstracto implica tener enlace dinámico.

Una clase abstracta se caracteriza por declarar al menos un métodos abstracto.

Una clase interfaz no debe tener atributos de instancia. Una clase abstracta si puede tenerlos.

Agregacion y composicion y herencia

La agregacion es una relacion mucho mas restrictiva que la herencia.

Ni la asociacion ni la agregacion manejan la creacion/destruccion de objetos tipo parte.

La Composicion maneja la creacion/destruccion de los objetos tipo parte.

Tanto composicion como herencia son mecanismos de reutilizacion del software.

Herencia

La herencia de interfaz se implementa mediante herencia publica.

La herencia publica implica herencia de implementacion y de interfaz mientras que la herencia privada o protegida implica herencia de implementacion pero no de interfaz.

Tanto composicion como herencia son mecanismos de reutilizacion del software.

Una de las principales fuentes de problemas cuando utilizamos herencia multiple es que las clases bases hereden de un ancestro comun.

Atributo y operaciones

En JAVA, un atributo de clase debe declararse dentro de la clase con el modificador static.

Un atributo de clase publico puede ser accedido desde fuera de la clase a traves de un objeto de la clase, un puntero o referencia al mismo o mediante el nombre de la clase seguido del operador de ambito.

Un atributo privado en la clase base no es directamente accesible en la clase derivada, independientemente del tipo de herencia utilizado.

Metodo

En el cuerpo de una operacion de clase no se puede acceder a ningun atributo/operacion del objeto receptor del mensaje.

En el cuerpo de una operacion de clase se puede acceder a unicamente a atributo/operacion de clase.

Una operacion de clase no puede tener enlace dinamico.

Una operacion de clase solo puede acceder directamente a atributos de clase.

Una operacion de instancia puede acceder directamente a atributos de clase y de instancia.

El enlace de la invocacion a un metodo sobrescrito se produce en tiempo de ejecucion en funcion del tipo del receptor del mensaje

Hablamos de shadowing cuando el metodo a invocar se decide en tiempo de compilacion

La signatura de tipo de un metodo incluye el tipo devuelto por el metodo.

Los metodos definidos en una clase derivada nunca pueden acceder a las propiedades privadas de una clase base.

Sea un metodo llamado glue(), sin argumentos, implementado en una superclase y sobrescrito en una de sus subclases. Siempre podremos invocar a la implementacion del metodo en la superclase desde la implementacion del metodo en la subclase usando la instruccion super.glue();.

Un metodo de clase (estatico) no puede tener enlace dinamico.

Buena Practica(cohesion) y visibilidad

Las sentencias ‘switch’ son un caso de codigo sospechoso (codigo con mal olor).

Una forma de mejorar el diseño de un sistema es reducir su acoplamiento y aumentar su cohesion.

Publica (+): se puede acceder al miembro de la clase desde cualquier lugar.

Protected(#). Accesible por todas las clases del mismo paquete y las derivadas (extends).

Package(~) (Por defecto). Accesible por todas las clases del mismo paquete pero desde fuera del paquete no.

Privada(-). Solo metodos de la misma clase.

En la misma clase, podemos definir constructores con distinta visibilidad.

C++

En C++ no podemos hacer sobrecarga de operadores para tipos predefinidos.

En C++, si no se define ningún constructor, el compilador proporciona por defecto uno sin argumentos.

La siguiente clase en C++: class S {public: virtual ~S()=0;}; define una interfaz.

En C++, un atributo de la clase debe declararse dentro de la clase con el modificador static.

La siguiente clase: class S {public: virtual ~S()=0; virtual void f()=0;}; constituye una interfaz en C++.

En C++, la clausula throw() tras la declaracion de una funcion indica q esta no lanza ninguna excepcion.

Cuando creamos un objeto en C++ mediante una variable automatica el constructor se autoinvoca. Tambien se autoinvoca el destructor del mismo al salir del ambito de la funcion donde se creo.

A diferencia de otros lenguajes de programacion en C++ la sobreesctritura en relaciones de herencia se debe indiciar de forma explicita en la clase padre.

La forma canonica de la clase esta formada por el constructor, el constructor de copia, el destructor y el operador de asignacion.

Constructor

Cuando se crea un objeto de la clase D que deriva de una clase B, el orden de ejecucion de los constructores es siempre B() D().

El objetivo de un constructor es crear e inicializar objetos.

En cuanto a los constructores son funciones miembro constantes.

En la misma clase podemos definir constructores con distinta visibilidad.

Es conveniente definir siempre un Constructor por Defecto que permita la inicializacion sin parametros de un objeto, donde los atributos se inicializan con valores por defecto.

Es posibles definir un constructor de copia invocando en su cuerpo al operador de asignacion.

Si en una clase no se declara, implicita o explicitamente, un constructor por defecto, no se pueden crear instancias de esa clase.

Downcasting

El downcasting implica deshacer el principio de sustitucion.

En JAVA el downcasting siempre se realiza en tiempo de ejecucion.

En el principio de sustitucion implica una coercion entre tipos de una misma jerarquia de clases.

Encapsulacion

Hablamos de encapsulacion cuando agrupamos datos junto con las operaciones que pueden realizarse sobre esos datos.

Hablamos de encapsulacion cuando diferenciamos entre interfaz e implementacion.

La encapsulacion es un mecanismo que permite separar de forma estricta interfaz e implementacion.

Errores

En JAVA, siempre es obligatorio especificar que excepciones verificadas (checked exceptions) lanza un metodo mediante una clausula throws tras la lista de argumentos.

Todas las excepciones son checked exception salvo las runtime que son uncheked exception.

Es una forma sistematica de introducir mejoras en el codigo que minimiza la posibilidad de introducir errores en el.

La instruccion throw en JAVA solo permite lanzar objetos que son de la clase throwable o clases derivadas de esta.

La instruccion throw permite lanzar como excepcion cualquier tipo de dato.

No se puede definir un bloque catch sin su correspondiente bloque try.

Podemos poner en bloque finally sin poner bloques catch.

Si en el bloque try de un catch, que captura chequed exception, es imposible que se produzca dicha excepcion el codigo no compilara, indicando que el bloque catch es inalcanzable.

Todo espacio de nombres define su propio ambito, distinto de cualquier otro ambito.

Un metodo sobrecargado es aquel que tiene mas de una implementacion, diferenciando cada uno por el ambito en el que se declara, o por el número, orden y tipo de argumentos que admite.

Un objeto se caracteriza por poseer un estado, un comportamiento y una identidad.

Uno de los objetivos del tratamiento de errores mediante excepciones es el manejo de errores del resto del codigo.

La instruccion throw permite lanzar como excepcion cualquier tipo de dato.

Si utilizamos los mecanismos de manejo de excepciones disminuye la eficiencia del programa incluso si no se llega a lanzar nunca una excepcion.

Frameworks y librerias

Arraylist es una implementacion en el Java Collection Framework de la interfaz List.

El usuario de un framework implementa al componente declarado de los interfaces de framework mediante herencia de interfaz.

El usuario de un framework implementa el comportamiento declarado en los interfaces del framework mediante herencia de interfaz.

JCF(Java Collection Framework) tiene un conjunto de clases en el JDK representando tipos de datos abstractos.

JDBC(Java Data Base Collection) es un framework que permite conectar Java con Bases de Datos.

JDBC es un framework de Java que usan los fabricantes de sistemas de gestion de bases de datos para ofrecer un acceso estandarizado a las bases de datos.

La inversion de control en los frameworks es posible gracias al enlace dinamico de metodos.

Para poder utilizar un framework, es necesario crear clases que implementen todas las interfaces declaradas en el framework.

Un Framework es un esqueleto para fines especificos que debemos completar y personalizar mediante la implementacion de interfaces, herencia de clases abstractas y ficheros de configuracion.

Un framework invoca mediante enlace dinamico a nuestra implementacion de interfaces propios de framework.

Una libreria de clases proporciona una funcionalidad completa, es decir, no requiere que el usuario implemente nada.

Genericidad

Dada una clase generica, se pueden derivar de ella clases no genericas.

En Java, una clase generica puede ser parametrizada empleado mas de un tipo.

En los metodos genericos solo podemos usar los metodos definidos en Object.

Hay dos tipos de genericos: Clases Genericas y Metodos Genericos.

La genericidad es un tipo de polimorfismo.

La genericidad es una propiedad que permite definir a una clase o una funcion sin tener que especificar el tipo de datos o algunos de sus miembros o argumentos.

La genericidad restringida permite indicar que los tipos genericos pertenezcan a una determinada jerarquia de herencia.

La genericidad se considera una caracteristica opcional de los lenguajes orientados a objetos.

Los metodos definidos en una clase generica son a su vez genericos.

Herencia

La herencia de interfaz se implementa mediante herencia publica.

La herencia publica implica herencia de implementacion y de interfaz mientras que la herencia privada o protegida implica herencia de implementacion pero no de interfaz.

Tanto composicion como herencia son mecanismos de reutilizacion del software.

Una de las principales fuentes de problemas cuando utilizamos herencia multiple es que las clases bases hereden de un ancestro comun.

Interfaz e implementacion

ArrayList es una implementacion en el Java Collection Framework de la interfaz List.

El principio de segregacion de interfaz indica que el codigo cliente no debe ser forzado a depender de interfaces que no utiliza.

El siguiente codigo en Java define una interfaz: interface S{}.

El usuario de un framework implementa el comportamiento declarado en los interfaces del framework mediante herencia de interfaz.

La herencia de interfaz se implementa mediante herencia publica.

Una clase interfaz no debe tener atributos de instancia. Una clase abstracta si puede tenerlos.

Una clase interfaz no puede tener instancias.

Una interfaz en Java obliga a que las clases no abstractas que la implementan definan todos los metodos que la interfaz declara.

Una interfaz es la definicion de un protocolo para cierto comportamiento, sin especificar la implementacion de dicho comportamiento.

Lenguajeoo

Una de las caracteristicas basicas de unos lenguajes orientados a objetos es que todos los objetos de la misma clase pueden recibir los mismos mensajes.

Un objeto se caracteriza por poseer un estado, un comportamiento y una identidad.

Todo espacio de nombres define su propio ambito, distinto de cualquier otro ambito.

Un espacio de nombres es un ambito con nombre.

Un sistema de tipos de un lenguaje asocia a cada tipo una expresion.

Cuando diseñamos sistema OO las interfaces de las clases que diseñamos deberian estar abiertas a la extension y cerradas a la modificacion.

Una de las caracteristicas basicas de una lengua orientada a objetos es que todos los objetos de la misma clase pueden recibir los mismos mensajes.

En el paradigma orientado a objetos, un objeto siempre es instancia de alguna clase.

En el paradigma orientado a objetos, un programa es un conjunto de objetos que se comunican mediante el paso de mensajes.

Metodo

La signatura de tipo de un metodo incluye el tipo devuelto por el metodo.

Los metodos definidos en una clase derivada nunca pueden acceder a las propiedades privadas de una clase base.

Sea un metodo llamado glue(), sin argumentos, implementado en una superclase y sobrescrito en una de sus subclases. Siempre podremos invocar a la implementacion del metodo en la superclase desde la implementacion del metodo en la subclase usando la instruccion super.glue();.

Un metodo de clase (estatico) no puede tener enlace dinamico.

Polimorfismo(this)

El cambio de una condicional por el uso de polimorfismo es un ejemplo de refactorizacion.

El cambio de una sentencia condicional por el uso de polimorfismo es un ejemplo de refactorizacion.

En Java los metodos de instancia con polimorfismo puro pero no abstractos tienen enlace dinamico.

La genericidad es un tipo de polimorfismo.

La sobrescritura es una forma de polimorfismo.

this es un ejemplo de variable polimorfica en JAVA.

Un metodo tiene polimorfismo puro cuanto tiene como argumentos al menos una variable polimorfica.

Una variable polimorfica puede hacer referencia a diferentes tipos de objetos en diferentes instantes de tiempo.

Refactorizacion

El cambio de una sentencia condicional por el uso de polimorfismo es un ejemplo de refactorizacion.

El codigo duplicado es un caso de codigo sospechoso en el que se aconseja el uso de tecnicas de refactorizacion para eliminarlo.

El principio de segregacion de interfaz indica que el codigo cliente no debe ser forzado a depender de interfaces que no utiliza.

Existe un catalogo de refactorizaciones comunes, de forma que el programador no se ve obligado a usar su propio criterio y metodologia para refactorizar el codigo.

La existencia de una solida coleccion de pruebas unitarias es una precondicion fundamental para la refactorizacion.

La refactorizacion debe hacerse siempre apoyandonos en un conjunto de tests completo y robusto.

La refactorizacion ha sido identificada como una de las importantes innovaciones en el campo del software.

Los metodos grandes (con muchas instrucciones) son estructuras que sugieren la posibilidad de una refactorizacion.

Refactorizar es una forma sistematica y segura de realizar cambios en una aplicacion con el fin de hacerla mas facil de comprender.

Un ejemplo de refactorizacion seria mover un metodo arriba o abajo en la jerarquia de herencia.

Una clase con un gran número de metodos y atributos es candidata a ser refactorizada.

Reflexion

La API de reflexion de JAVA incluye metodos para obtener la signatura de todos los metodos.

La Reflexion es una infraestructura del lenguaje que permite a un programa conocerse y manipularse a si mismo en tiempo de ejecucion.

La reflexion permite que un programa obtenga informacion sobre si mismo en tiempo de ejecucion.

La Reflexion puede usarse para construir nuevos objetos y arrays, acceder y modificar atributos de instancia, invocar metodos de instancia y estaticos.

Mediante reflexion no podemos saber cual es el metodo que se esta ejecutando en un determinado momento.

Mediante reflexion podemos saber cuales son las clases derivadas de una clase dada.

Podemos usar reflexion para encontrar un metodo heredado (solo hacia arriba) y reducir codigo condicional.

Sobrecarga

En la sobrecarga basada en ambito los metodos pueden diferir unicamente en el tipo devuelto.

En la sobrecarga de operadores binarios para objetos de una determinada clase, si se sobrecarga como funcion miembro, el operando de la izquierda es siempre un objeto de la clase.

La sobrecarga basada en ambito permite definir el mismo metodo en dos clases diferentes.

Un metodo sobrecargado es aquel que tiene mas de una implementacion, diferenciando cada una por el ambito en el que se declara, o por el numero, orden y tipo de argumentos que admite.