SEMANA UNO

ABSTRACCIÓN Y EFICIENCIA

Abstraccion: es un proceso mediante el cual se identifican los aspectos relevantes de un problema ignorando los detalles. Partes Importantes (Bajo acoplamiento, alta cohesion, Suficiencia y completitud, Primitividad)
Eficiencia: El programador debe expresar algoritmos eficientes para el codigo creado.

ENTORNO

Entorno: Aunque los lenguajes sean debiles, si tiene un entorno amigable son altamente utilizados.

SINTAXIS Y SEMANTICA

Sintaxis: conjunto de reglas que debemos seguir para que el compilador sea capaz de reconocer un programa como válido.
Semantica: reglas que determina el significado de los programas constituyen la semántica de los lenguajes de programación

PRAGMÁTICA Y VALOR

Grado de éxito con el que un programa cumple sus objetivos tanto en su fidelidad con el modelo de computación subyacente como su utilidad para los programadores
Valor: Un valor puede ser virtualmente cualquier clase de dato

PORTABILIDAD Y SEGURIDAD

Portabilidad: El lenguaje debe facilitar la creacion de programas en la mayoria de los entornos computacionales.
Seguridad: Lo ideal es que los programas incorrectos no pertenezcan al lenguaje y sean rechazados por el compilador.

LENGUAJE DE PROGRAMACIÓN Y TIPOS

Un conjunto de caracteres, símbolos, representaciones y reglas que permiten introducir y tratar la información en una computadora (ordenador). Tipos: Estandar, Usuario, simple, c

CONCISIÓN NOTACIONAL

El lenguaje proporciona un marco conceptual para pensar algoritmos y expresar dichos algoritmos con el nivel de detalle adecuado.

ORTOGONALIDAD Y EXTENSIBILIDAD

Ortogonalidad: un lenguaje es ortogonal si puede ser comprendida y combinada de forma independiente
Extensibilidad: El lenguaje debe contenes ayudas para el progrmador, para crear nuevas construccioes

EXPRESIONES

Combinación de constantes, variables o funciones, que es interpretada de acuerdo a las normas particulares de precedencia y asociación para un lenguaje de programación en particular. Las expresiones están compuestas de operadores, operandos, paréntesis* y llamadas a funciones

PROGRAMA Y DATO

Programa: Conjunto de instrucciones en código máquina según el cual el computador realizará una serie de acciones
Dato: representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa

SEMANA DOS

ALMACENAMIENTO Y CONTROL

La ejecución de un programa requiere que diversos elementos se almacenen en memoria: – Código del programa (instrucciones). – Datos: • Permanentes • Temporales – Direcciones de control de flujo de ejecución del programa

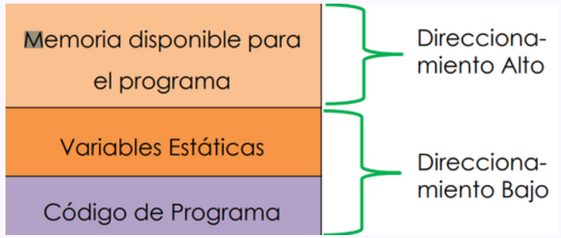
MEMORIA ESTÁTICA

El tamaño no cambia durante el tiempo de ejecución. • Algunos lenguajes de programación usan la palabra static para estos casos. • Ejemplos: – Código – Variables de programa. – Variables estáticas

MEMORIA DINÁMICA

Define el tamaño del espacio de memoria necesario para un programa en tiempo de ejecución. • Puede cambiar durante la ejecución. • Son variables dinámicas las que cambian de tamaño. • Se almacenan en el espacio de memoria llamado heap.

ASIGNACIÓN DE MEMORIA



LIBERACIÓN DE MEMORIA

La liberación de memoria estática se da cuando el programa finaliza su ejecución. • La memoria dinámica se libera de dos formas: – Explicita: • Cuando se programador lo indica – Implícita: • El lenguaje de programación lo hace por medio del garbage collector.

HEAP

Espacio en memoria que se utiliza en tiempo de ejecución • Almacena estructuras dinámicas – Parámetros y variables de funciones – Instancias de clases (objetos) – Arreglos

GARBAGE COLLECTOR

• Maneja la asignación y liberación de memoria de las aplicaciones • Se encarga de eliminar los objetos cuando ya no existen referencias a ellos. • Se ejecuta automáticamente de manera periódica.Administra el heap • Optimiza los tiempos de “colección” • El proceso revisa las asignaciones de memoria una a una

ASOCIACIÓN “BINDING”

Es la relación que se establece entre la invocación de un método y el código que lo ejecuta. • El binding tiene un scope (alcance) que es el conjunto de instrucciones en donde se encuentra activo (significa que la entidad podría ser accedida)

ASOCIACIÓN “BINDING”

Tiempo de vida: es la duración de un binding en la ejecución de un programa, desde su creación hasta su destrucción: – Variables globales: todo el programa. – Variables locales: durante la invocación de un método. – Variables estáticas: todo el programa.

ASOCIACIÓN “BINDING”

Tipos de binding:
– Estático: Se realiza en tiempo de compilación. const n=3; • int c;
– DinámicoDato: – Se realiza en tiempo de ejecución

```
void imprimir(Expresion E) {
   system.println( E.value() );
}
```

SEMANA DOS

ABSTRACCIÓN

La abstracción es un proceso mediante el cual se identifican los aspectos relevantes de un problema ignorando los detalles. • Permite distinguir aquellas características fundamentales de un objeto que lo hacen diferente del resto. • Es la capacidad de centrarse en las similitudes e ignorar las diferencias.

ABSTRACCIÓN

Se deben revisar los siguientes aspectos para lograr abstracciones de calidad: • Acoplamiento. • Cohesión. • Suficiencia y completitud. • Primitividad
– Acoplamiento: • Minimizar el grado de asociación entre los diferentes abstracciones.

ENCAPSULACIÓN

Es la capacidad que permite mantener oculta la implementación de una abstracción para los usuarios de la misma. • El objetivo es la ocultación de la implementación, para evitar dependencia en sistemas complejos

ENCAPSULACIÓN

Se enfoca en cómo se construye un objeto. • Se encarga de los detalles de la implementación. • Genera la barrera de la abstracción. • Genera modificadores de visibilidad: public, private y protected.

ABSTRACCIÓN

• Ayuda a separar el comportamiento esencial de un objeto de su implementación. • Los lenguajes OO facilitan abstraer porque permiten definir interfaces comunes para comunicarse con clases de objetos. • Esas interfaces son los métodos (funciones).

ENCAPSULACIÓN

Este lineamiento asegura código altamente modularizado. • Nuevas características o correcciones del código van a ser más fáciles de construir.

ABSTRACCIÓN

La abstracción permite la conceptualización de clases, que es la abstracción de un objeto. • Se añaden propiedades que son variables internas al objeto o a la clase que definen el estado. • Métodos y propiedades también se conocen cómo miembros de la clase.

ABSTRACCIÓN

Cohesión: • Maximizar el grado de asociación dentro de una abstracción.
– Suficiencia y completitud: • Que tenga las características precisas para permitir un funcionamiento eficiente y completo.
Primitividad: • Las operaciones de una abstracción deben ser lo más básicas posibles.

ENCAPSULACIÓN

Los atributos de una instancia deberían ser leídos o escritos por métodos de esa misma clase. • Los métodos encapsulan el conocimiento sobre lo que hay que hacer con los datos.

SECUENCIADORES

Un secuenciador es una construcción del lenguaje para transferir el control a algún otro punto en un programa. • Un secuenciador implementa un flujo de control.

SEMANA DOS

SECUENCIADORES

Permite variaciones del flujo normal de control. • Permite que más flujos generales puedan ser realizados por los programas: – Rutinas con múltiples salidas – Rutinas con múltiples entradas Ejemplos: – Jumps – Escapes – Excepciones – Corroutinas (co-rutines)

JUMPS

Transferencia explícita de control desde un punto de programa a otro punto. • Ejemplos: goto. • Se encuentra en C, Pascal, Fortran y Basic

ESCAPES

Termina la ejecución de comando compuesto. • Es una forma estructurada del jump. • Ayuda hacer programas más claros. • Permite para una sóla de entrada múltiples salidas.
Salida de un ciclo – Ejemplos: exit en Ada, break en C, Java • Salida de una método – Ejemplo: return en C y Java • Escape especializado – Break para salir de un switch en Java o C#

EXCEPCIONES

Evento anómalo o excepcional que ocurre durante la ejecución de un programa • Ocurre con poca frecuencia • Cambia el flujo normal del programa
Hacer más robusto el programa • Recuperarse de errores • Ejemplo: – División por cero – Archivo no encontrado – Memoría agotada

EXCEPCIONES

• El compilador de un lenguaje de programación detecta errores, pero estos son sólo errores de sintaxis • Hay otros errores que surgen durante la ejecución del programa

CORRUTINA

Mecanismo que permite a una rutina hacer una pausa de su ejecución en cierto punto, hacer un jump a otro punto y reanudar desde el punto subsecuente de la llamada del jump. • Ejemplo: – Dentro de un método ejecutar otro y continuar.