Tipos de Datos Abstractos

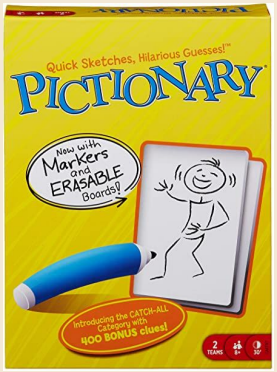
75.41 - Algoritmos y Programación II

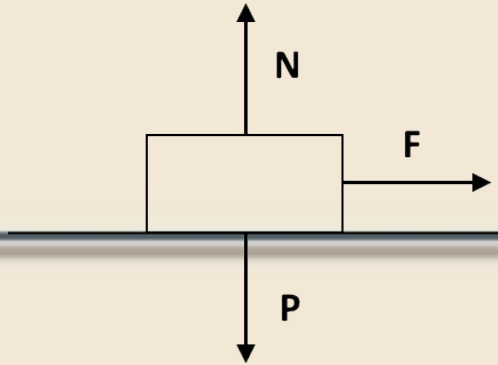
2° Cuatrimestre 2021

Resumen

- Abstracción ¿Que es?
- Tipos de Datos
- ¿Qué?¿Cómo?
- Ejemplos/Implementaciones
- El proceso de Desarrollo de Software
- Ventajas

¿Que tienen en común estas cosas?





El concepto de Abstracción

“Separar aisladamente en la mente las características de un objeto o un hecho, dejando de prestar atención al mundo sensible para enfocarse solo en el pensamiento.”

- Separamos lo que nos **importa.** ¿Porque nos importa eso?
- De lo innecesario **nos olvidamos.** ¿Por qué eso no?
- ¿Qué **consideramos?**

No es Por que,
es **Para que**

Los Tipos de Datos

¿Qué tipos de dato conocen?

¿Que definimos con ellos?

- El conjunto de todos los valores posibles
- Las operaciones que pueden utilizar

Tipo	Conjunto de Valores	Operador	Operación	Resultado
int	-2^{31} y $(2^{31}-1)$ implementado como complemento a 2	+ (suma)	$5+3$	8
		- (resta)	$5-3$	2
		* (multiplicación)	$5*3$	15
		/ (división entera)	$5/3$	1
		% (resto)	$5\%3$	2

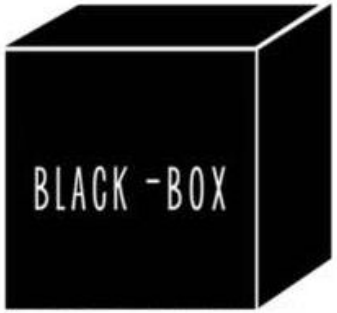
Tipo de Dato Abstracto

“Define una clase de objetos abstractos los cuales están completamente caracterizados por las operaciones que pueden realizarse sobre esos objetos.”
--Liskov y Zilles

El Qué y el Cómo

El Qué

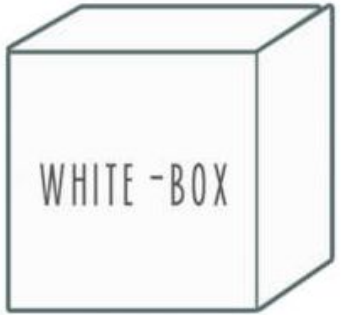
- Acá hablamos de las operaciones que podemos hacer con algo.
- Hablamos de funcionalidad, de ahí la pregunta: ¿**Qué** hace esto?
- Hablamos de **caja negra**.



ZERO KNOWLEDGE

El Cómo

- Acá hablamos de la forma en que algo está diseñado o implementado.
- Nos interesa la estructura interna o la forma en la cual se lleva a cabo algo.
- La pregunta es: ¿**Cómo** lo hago?
- Hablamos de **caja blanca**.



FULL KNOWLEDGE

El Qué y el Cómo

CONTRATO

Implementa

tda.h

Archivo encabezado

Utiliza

tda.c

Archivo Fuente

tda.o

Archivo Objeto

Ejemplo: Número Complejo

¿Qué cosas deberíamos poder hacer con un número complejo?

- Crearlo
- Sumar
- Restar
- Multiplicar
- Dividir
- Imprimir

¿Y la estructura?

```
typedef struct complejo {
    double real;
    double imag;
} complejo;
```

Ejemplo: Número Complejo

```
/* c = a + b */
void add_c (complejo * c, complejo a, complejo b) {
    c-> real = a. real + b. real ;
    c-> imag = a. imag + b. imag ;
}

/* c = a * b */
void mul_c (complejo *c, complejo a, complejo b ) {
    c-> real = a. real * b. real - a. imag * b. imag ;
    c-> imag = a. imag * b. real + a. real * b. imag ;
}


/* c = a / b */
void div_c (complejo *c, complejo a, complejo b) {
    double _abs_sq = b. real * b. real + b. imag * b. imag ;
    c-> real = (a. real * b.real + a.imag * b.imag ) / _abs_sq ;
    c-> imag = (a. imag * b.real - a.real * b.imag ) / _abs_sq ;
}
```

Ejercicio! Un caso de uso que todos conocemos...

crear: esta operación debería generar una tarjeta SUBE válida.

destruir: esta operación deberá eliminar una tarjeta SUBE.

cargar: esta operación permite cargar saldo en pesos a la tarjeta SUBE.

realizar-viaje: esta operación permite descontar el importe de un viaje a la tarjeta SUBE

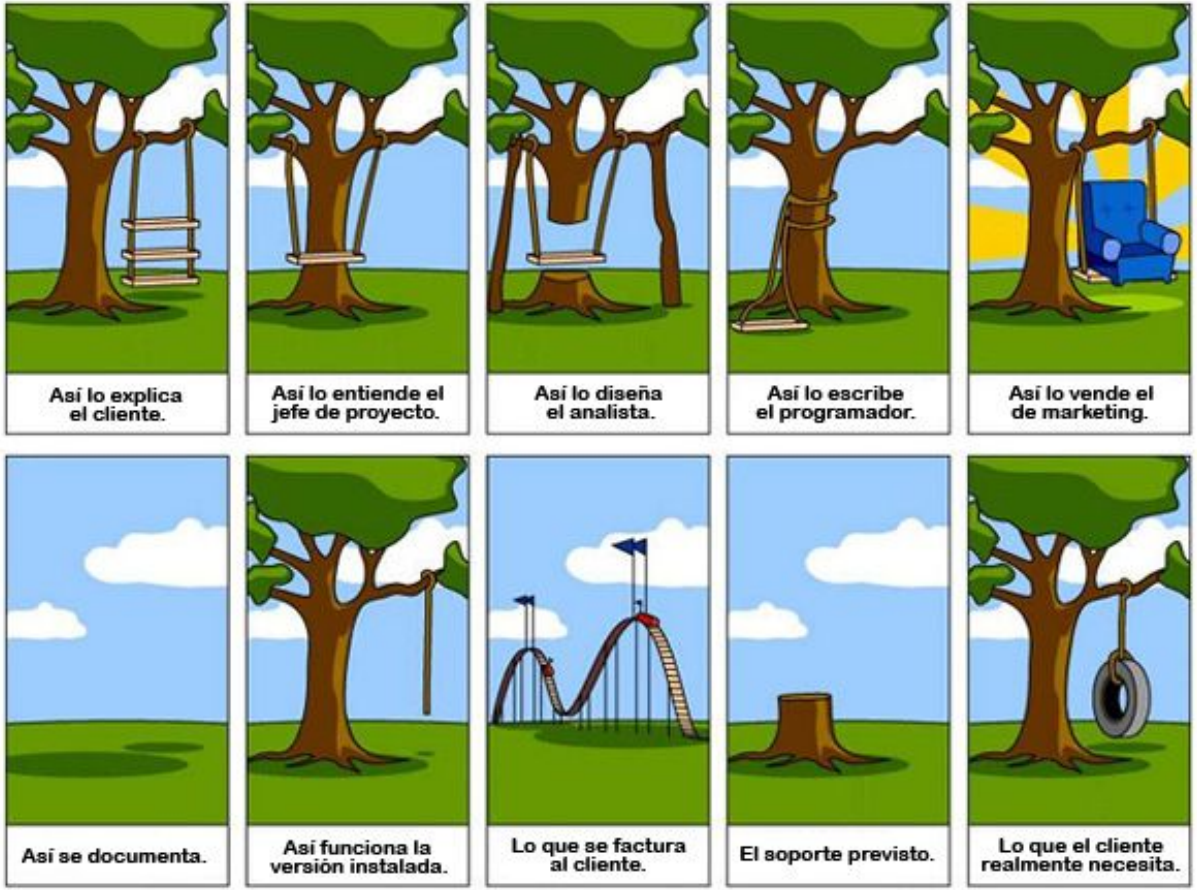
saldo: esta operación devuelve el importe del saldo de la tarjeta SUBE.

propietario: devuelve el dni del propietario de la tarjeta SUBE

El Proceso de Desarrollo de Software


*
INICIO
REQUERIMIENTOS
ANÁLISIS
DISEÑO
IMPLEMENTACIÓN
PRUEBAS
INSTALACIÓN
FIN ?
MANTENIMIENTO!
*

Un comentario desde la Experiencia...



- Pensar en el **que** antes de meterse en el como.
- Tener en claro las ideas uno y ser claro a la hora de transmitirlas.
- Pensar en el contrato del TDA.

Entonces… ¿Por qué usamos TDAs?

Manejan la Abstracción

La abstracción permite simplificar la realidad mediante el despojo de complejidad que no es propio del problema que estamos resolviendo.

Encapsulamiento

Es la propiedad por la cual un TDA debe exponer la menor cantidad posible de informacion del **como** esta implementado, haciendo que el usuario se base en las funciones que él mismo entiende.

Localización del Cambio

Cuando existen errores dentro de un programa, es más fácil detectarlo, pues la utilización de los TDAs fuerza la modularización.

Directed by
ROBERT B. WEIDE

¿Preguntas?