Repaso TADs

Que es? El que y no el como. Tiene estado Se manipula atravez de operaciones

Diseño de TADs Un diseño es una estrctura de datos y una serie de algoritmos que nos indica como se representa.
- Tenemos que elegir una estructura de datos para implementarlo, con sus pros y sus contras.
Hay muchas formas de hacer lo mismo, las personas piensan de formas distintas y/o tienen distintos motores, (necesitas mas rapidez, eficiencia, etc)
Lo importante es cumplir el qué(la especificacion), el como cada uno lo maneja.

Ocultar informacion Principio de ocultacion. No tengo por que mostrar como hago las cosas mientras se respete el resultado final.
Facilita la comprension.
Favorece el reuso.
Ayuda a modularizar y separar el trabajo, es tambien el sistema mas resistente a cambios.

Encapsulamiento = god

TAD Punto Primero pienso las operaciones, los observadores solo sirven para explicar que hacen.

```
modulo PuntoImpl implementa Punto
var rho: float
var theta: float
```

da igual como lo hagamos, mientras haga lo que debe.

Cuando definiamos un tad usabamos tipos basicos, el mundo de la "matematica", ahora los tipos de las variable son los tipos de implementacion.
-int, float, char...
-tupla arrays
estructs, modulos de otros tads, y mas cosas.

El invariante de representacion Cuando elegimos los tipos tenemos que tener cuidado de que pueda cumplir con lo que necesitamos (si queremos hacer una lista de ocho elementos, por ejemplo una tupla de, que es fija, de dos casillas no nos serviria)
Volviendo al ejemplo del punto:
Tenemos al menos dos formas de almacenar el angulo theta, normalizado (entre cero y dos pi, o entre menos pi y mas pi), desnormalizado(cualquier valor real)
Tenemos que ser consistentes, nuestras operaciones pueden asumir que se cumplen estas cosas, pero al final de nuestra operacion debemos asegurarnos de que todo siga siendo consistente y cumplamos nuestro propio çontrato".
Es algo que me dice el estado de consistencia de mi representacion interna.

TAD Punto: Invariante de representacion El invariante se escribe en logica(usando lenguaje de especificacion) haciendo referencia a la estructura de implementacion.

```
pred InvRep (p':PuntoImpl) {
```
$-\pi \leq p'.theta < \pi \wedge rho \geq 0$
```
}
```

Funcion o predicado de abstraccion Tenemos dos estrucutras, el tad y la implementacion. ¿Como los relacionamos?
Nos indica para cada instancia de la implementacion, a que instancia del tad representa, que instancia del tad .es su abstraccion"
Hace referencia a las variables de estado de la implementacion y a los observadores del TAD (porque tiene que vincular unas cosas con otras)
Para definir se puede suponer que vale el invariante de representacion.

TAD Punto: funcion (predicado) de abstraccion

```
FuncAbs(p':PuntoImpl): Punto {
p: punto |
    p.x=p'.rho*cos(p'.theta)&& p.y=p'.rho*sin(p'.theta) }
```

Un tad es una entidad con estado que manipulas con operaciones, no te importa cuanto vale solo te importa lo que puede hacer.

La funcion abstraccion conecta ambos mundos. Nos explica como se mapean los observadores, en que se traducen en la realidad.

Tambien podemos escribirla como un predicado si nos resulta mas comodo.

Es la explicacion de como conectamos lo concreto con lo abstracto.

TAD Punto: algoritmos Solo nos queda escribir los algoritmos:

impl mover(inout c': PuntoImpl, in deltaX: float, in deltaY: float){
float nuevoX := c'.coordX() + deltaX;
float nuevoY := c'.coordY() + deltaY;
c'.rho := $sqrt(nuevoX^2 + nuevoY^2)$;
c'.theta := arctan(nuevoY / nuevoX);
}

impl `hola(prueba)`{
`u:= cosa`
}

TAD Conjunto: diseño

colgue toda esta parte

Correctitud en TADs

Verificacion: Se puede demostrar que la implementacion de un tad es correcta respecto a la especificacion.
El abstracto y el concreto se tienen que "mover igual"
Hay que demostrar que cada operacion conserva el invariante, y que el algoritmo respeta la pre y post condicion del tad.
agarras el concreto, abstraelo, fijate la pre, aplica la funcion, fijate la post

$preTAD \rightarrow_{abstraccion} preImpl \rightarrow postImpl \rightarrow_{abstraccion} postTAD$

TAD ConjuntoAcotado