Programación II. Algoritmos y Estructuras de datos II. 2024

Programación II. Algoritmos y EDD II. 2024

Agenda 18 de Marzo 2024:

- Repaso TDA
- TDA PILA. Implementación
- TP1. Implementación

Programación II. Algoritmos y EDD II. 2024

Dentro de la filosofía de los **objetos**, los TDAs son un elemento clave.

Como se puede deducir de lo que vimos hasta hoy, es más fácil tratar con una abstracción que tener en cuenta una multiplicidad de detalles que pueden ser en su mayoría superfluos para nuestros objetivos.

Programación II. Algoritmos y EDD II. 2024

Dentro de la filosofía de los **objetos**, los TDAs son un elemento clave.

Como se puede deducir de lo que vimos hasta hoy, es más fácil tratar con una abstracción que tener en cuenta una multiplicidad de detalles que pueden ser en su mayoría superfluos para nuestros objetivos.

Un TDA se define a través de su semántica (comportamiento.) Los detalles de la implementación quedan ocultos al usuario.

Programación II. Algoritmos y EDD II. 2024

Dentro de la filosofía de los **objetos**, los TDAs son un elemento clave.

Como se puede deducir de lo que vimos hasta hoy, es más fácil tratar con una abstracción que tener en cuenta una multiplicidad de detalles que pueden ser en su mayoría superfluos para nuestros objetivos.

Un TDA se define a través de su semántica (comportamiento.) Los detalles de la implementación quedan ocultos al usuario.

El TDA se define a través de una interfaz, que es una clase abstracta que contiene la especificación de los métodos que ofrece el TDA.

Programación II. Algoritmos y EDD II. 2024

Dentro de la filosofía de los **objetos**, los TDAs son un elemento clave.

Como se puede deducir de lo que vimos hasta hoy, es más fácil tratar con una abstracción que tener en cuenta una multiplicidad de detalles que pueden ser en su mayoría superfluos para nuestros objetivos.

Un TDA se define a través de su semántica (comportamiento.) Los detalles de la implementación quedan ocultos al usuario.

El TDA se define a través de una interfaz, que es una clase abstracta que contiene la especificación de los métodos que ofrece el TDA.

Hay otra clase que hereda los métodos de la interfaz y los implementa. Esta clase podría suplantarse por otra implementación diferente de manera que el usuario no podría percibirlo.

Programación II. Algoritmos y EDD II. 2024

Estado y Comportamiento

- Un TDA tiene un **estado** que está dado por los valores de sus parámetros (por ejemplo, el monto del saldo en una cuenta bancaria.)

Programación II. Algoritmos y EDD II. 2024

Estado y Comportamiento

- Un TDA tiene un **estado** que está dado por los valores de sus parámetros (por ejemplo, el monto del saldo en una cuenta bancaria.)
- Un TDA tiene un **comportamiento** que está definido por los métodos que ofrece (por ejemplo, la posibilidad de efectuar retiros de una cuenta o de depositar dinero en ella.)

Programación II. Algoritmos y EDD II. 2024

Estado y Comportamiento

- Un TDA tiene un **estado** que está dado por los valores de sus parámetros (por ejemplo, el monto del saldo en una cuenta bancaria.)
- Un TDA tiene un **comportamiento** que está definido por los métodos que ofrece (por ejemplo, la posibilidad de efectuar retiros de una cuenta o de depositar dinero en ella.)
- Muchas veces se refiere uno al comportamiento de un TDA como su **semántica**

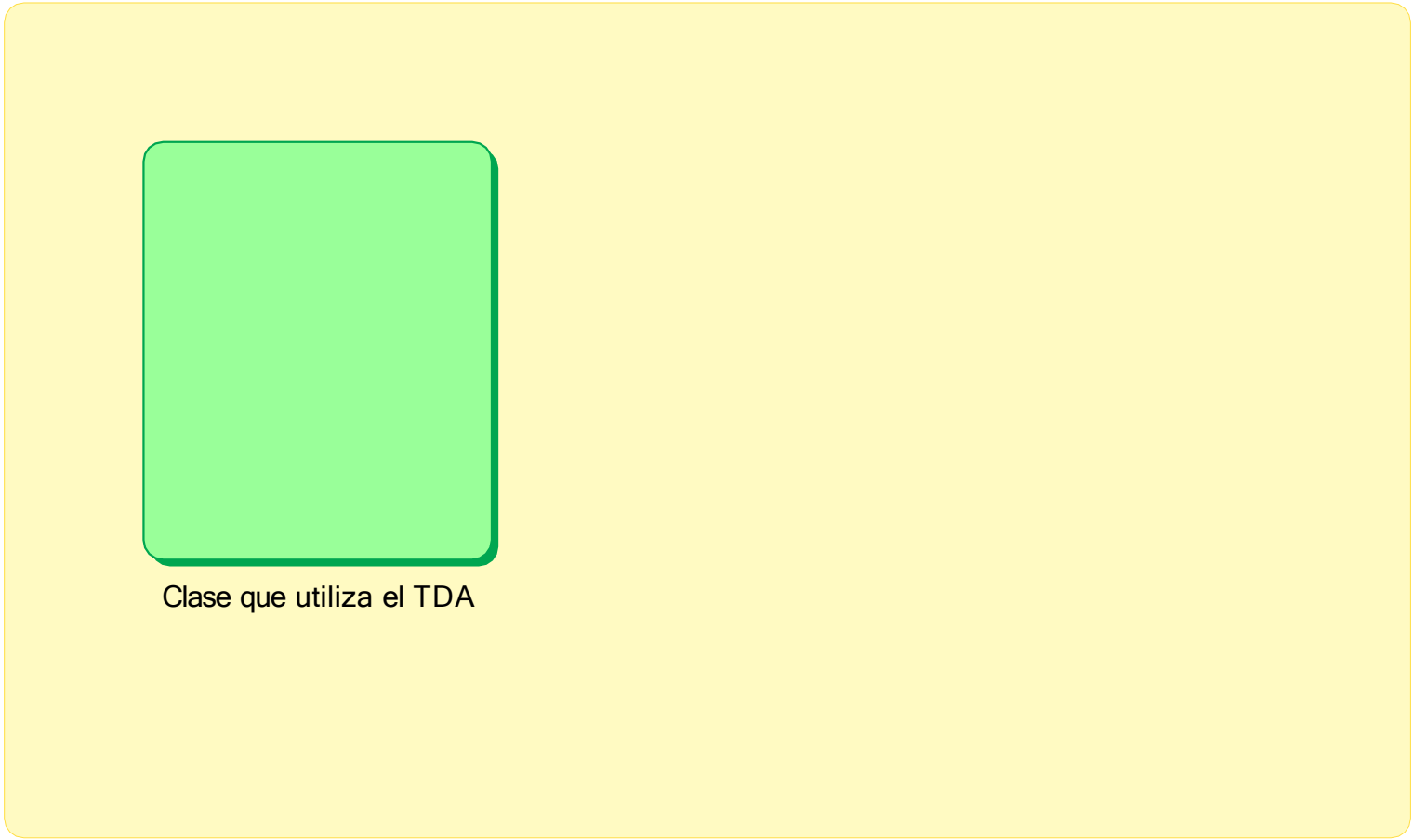
Programación II. Algoritmos y EDD II. 2024

Estado y Comportamiento

- Un TDA tiene un **estado** que está dado por los valores de sus parámetros (por ejemplo, el monto del saldo en una cuenta bancaria.)
- Un TDA tiene un **comportamiento** que está definido por los métodos que ofrece (por ejemplo, la posibilidad de efectuar retiros de una cuenta o de depositar dinero en ella.)
- Muchas veces se refiere uno al comportamiento de un TDA como su **semántica**
- Una implementación es correcta si el comportamiento del TDA es el esperado para cualquier estado posible.

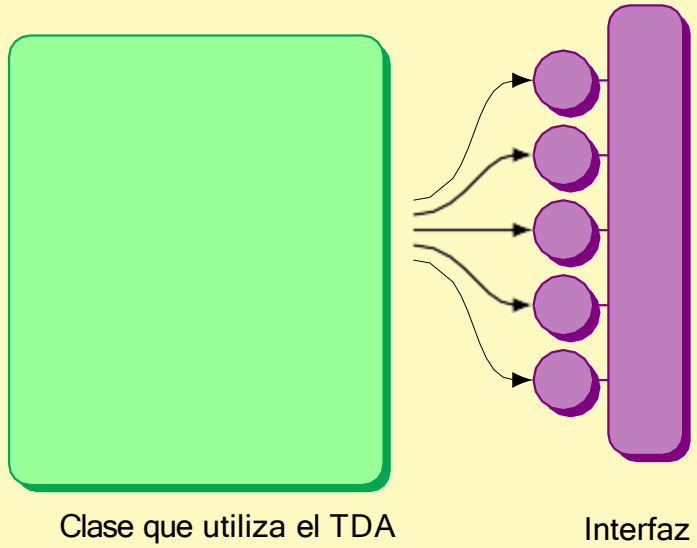
Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



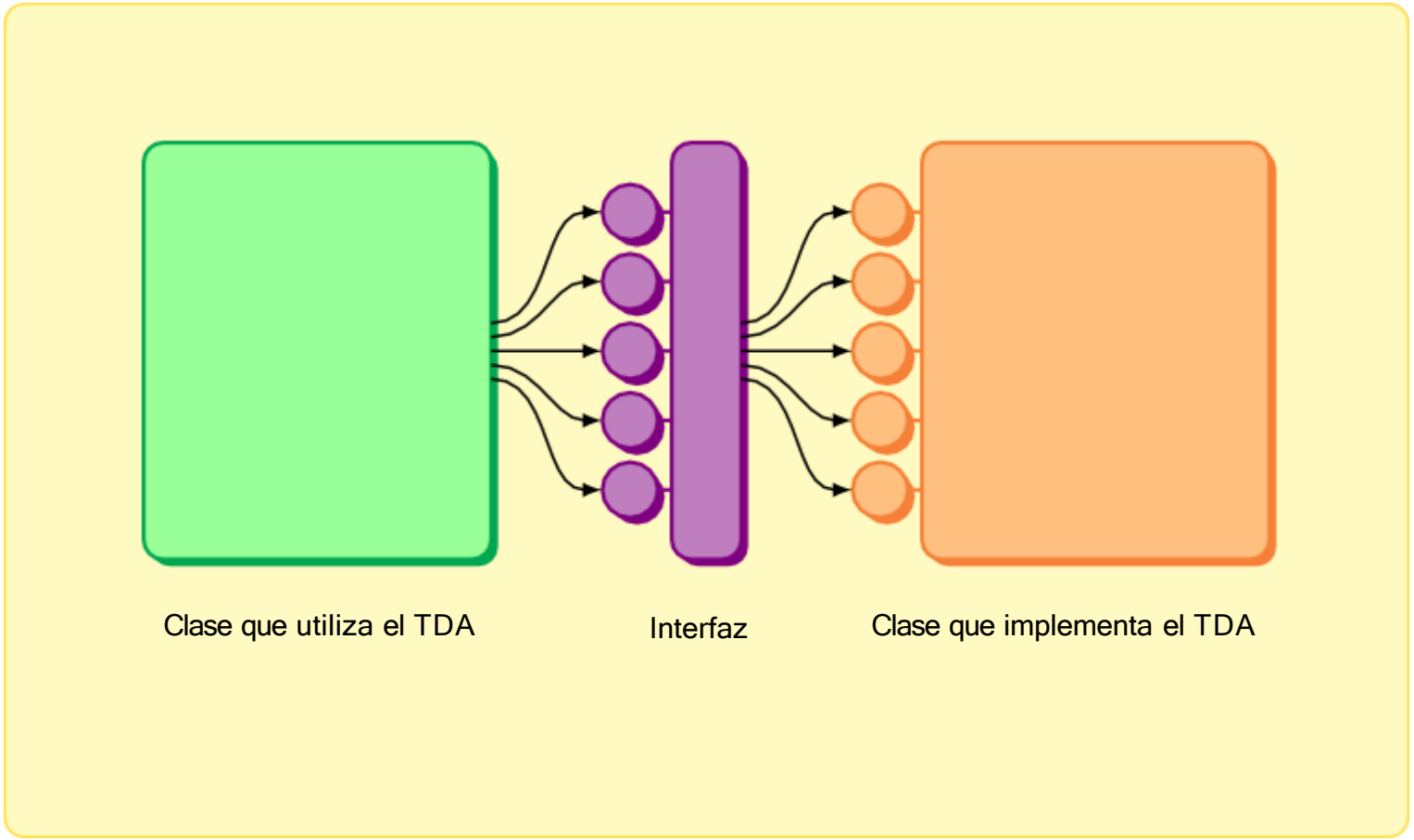
Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



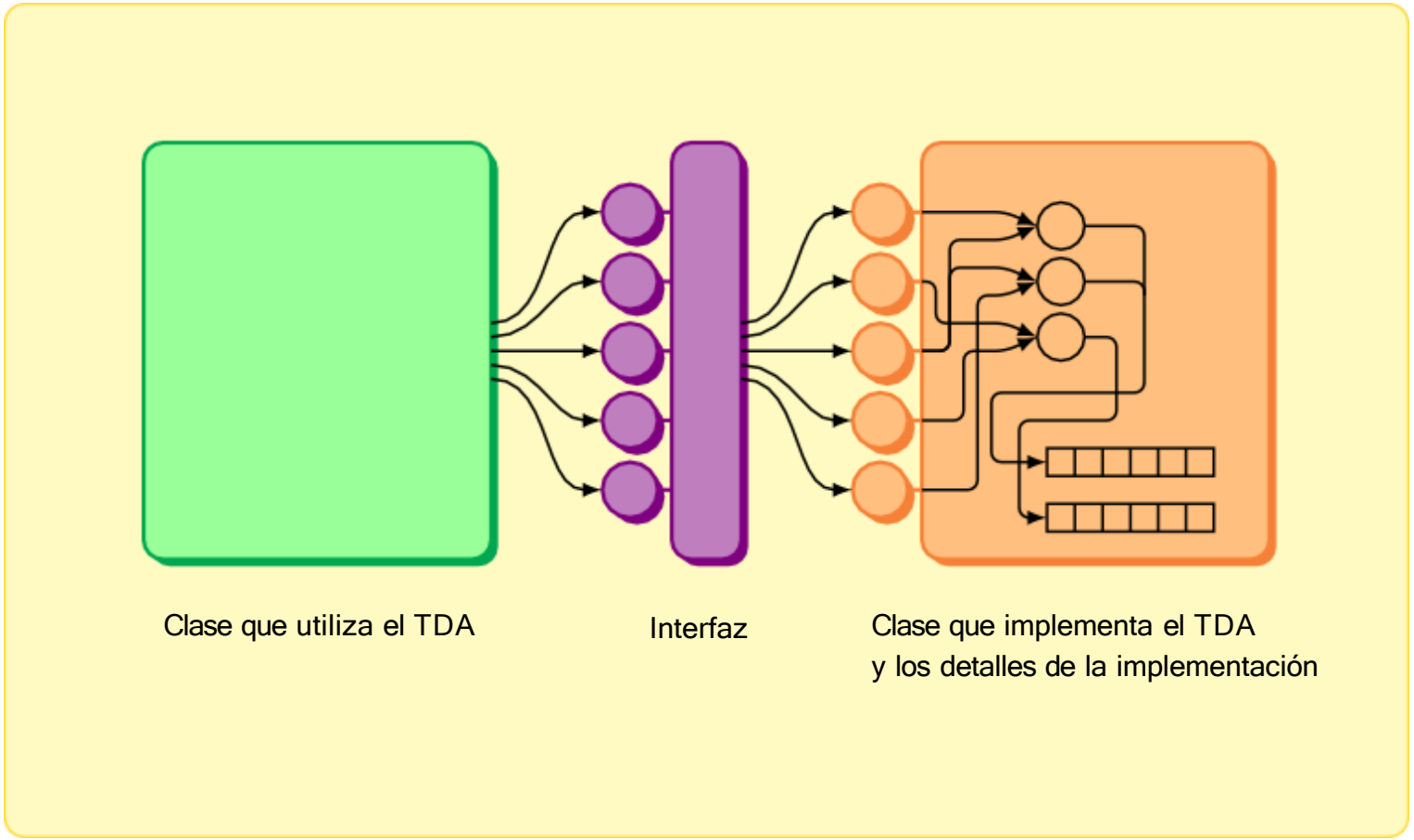
Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



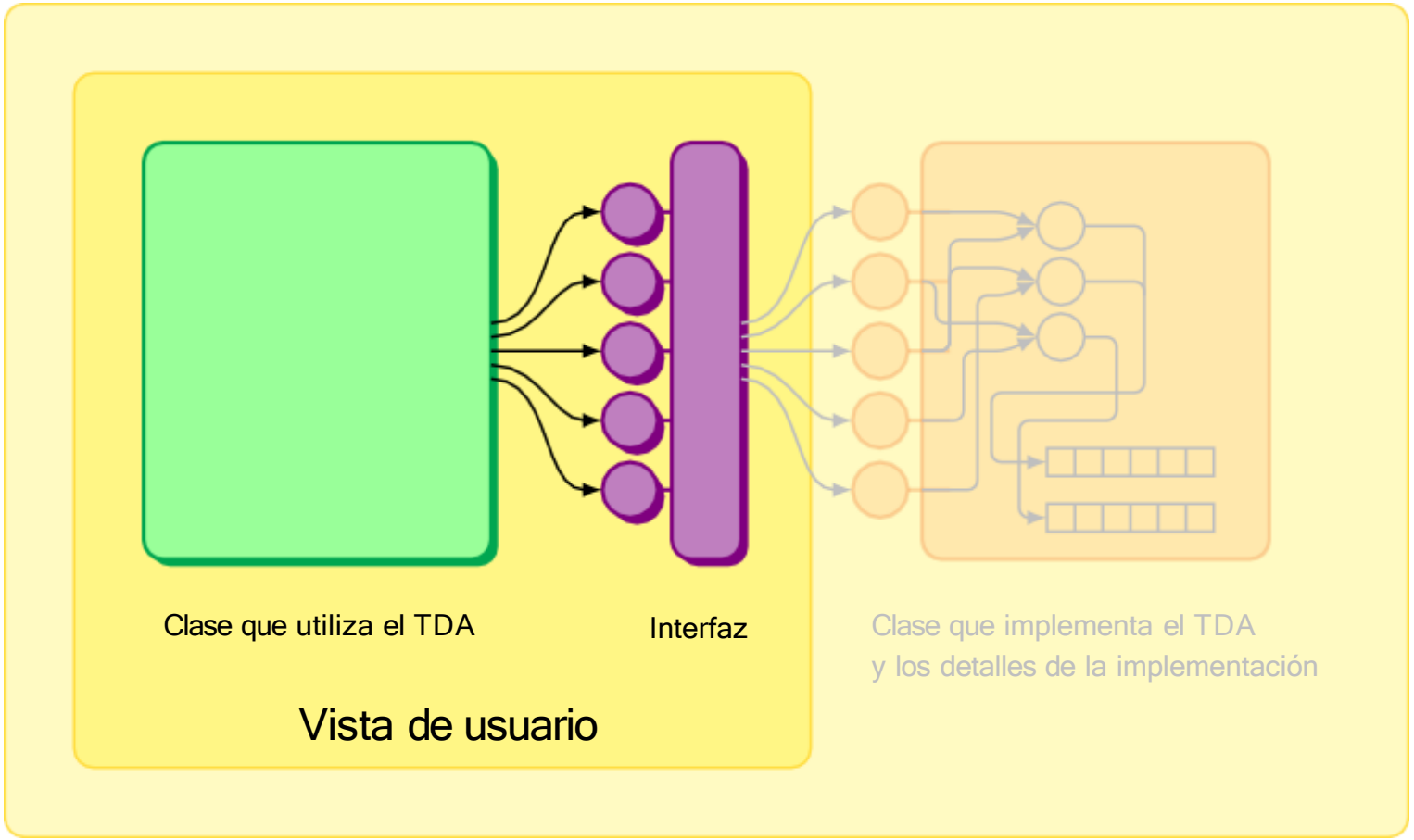
Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



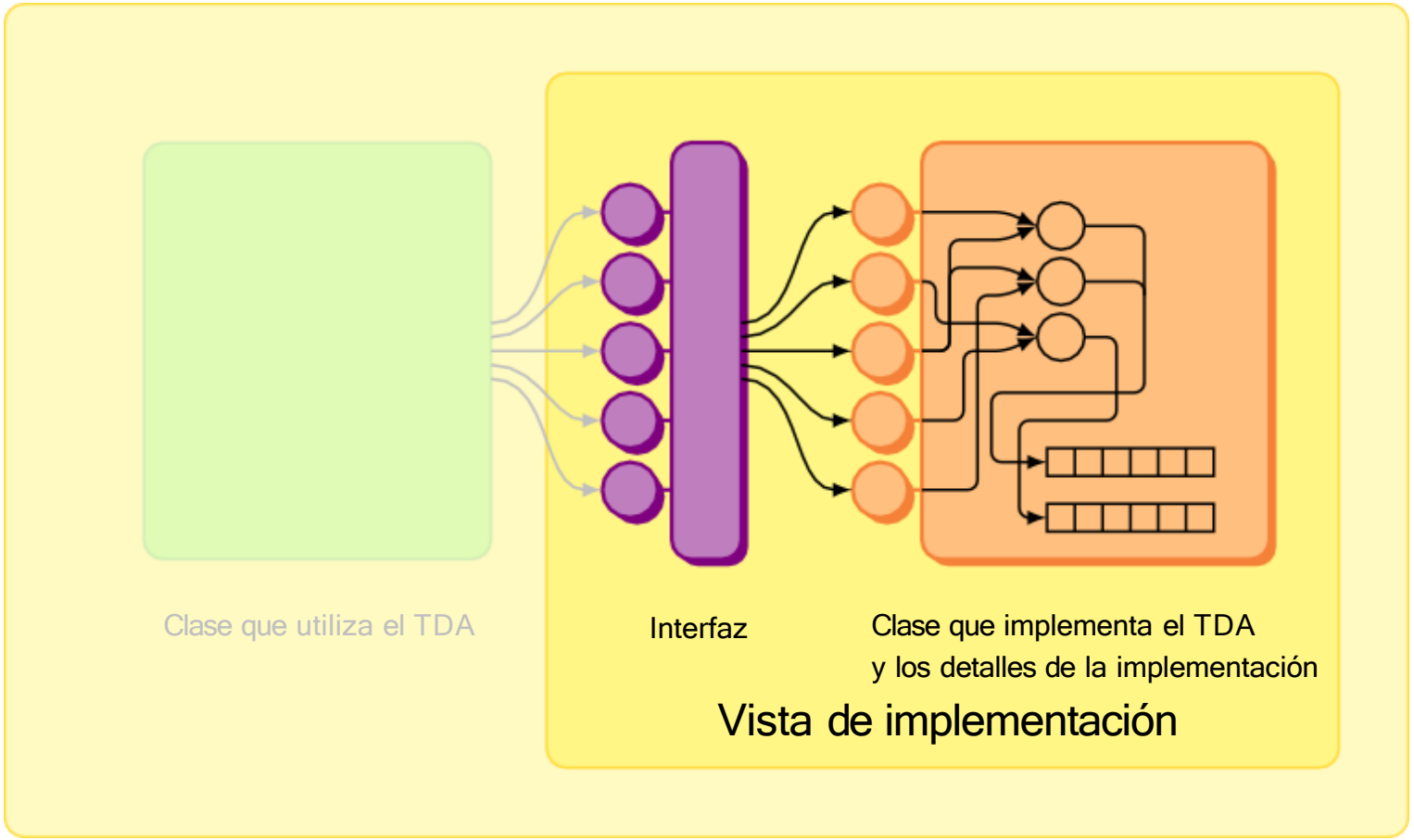
Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



Programación II. Algoritmos y EDD II. 2024

Esquema de las vistas de un TDA



Programación II. Algoritmos y EDD II. 2024

Resumen

Programación II. Algoritmos y EDD II. 2024

Resumen

Es necesario saber exactamente hacia dónde uno va antes de ponerse en marcha. En otras palabras, hay que comprender exactamente qué es lo que debemos hacer y cuál es el problema que debemos resolver.

Programación II. Algoritmos y EDD II. 2024

Resumen

Es necesario saber exactamente hacia dónde uno va antes de ponerse en marcha. En otras palabras, hay que comprender exactamente qué es lo que debemos hacer y cuál es el problema que debemos resolver.

Hay que establecer una clara estrategia de resolución. No hay que comenzar con la codificación hasta no tener definida esta estrategia

Programación II. Algoritmos y EDD II. 2024

Resumen

Es necesario saber exactamente hacia dónde uno va antes de ponerse en marcha. En otras palabras, hay que comprender exactamente qué es lo que debemos hacer y cuál es el problema que debemos resolver.

Hay que establecer una clara estrategia de resolución. No hay que comenzar con la codificación hasta no tener definida esta estrategia.

Esto implica determinar exactamente qué sabemos y qué debemos averiguar y definir las estructuras de datos que necesitaremos para representar nuestro problema, nuestros datos y nuestra solución.

Programación II. Algoritmos y EDD II. 2024

Resumen

Es necesario saber exactamente hacia dónde uno va antes de ponerse en marcha. En otras palabras, hay que comprender exactamente qué es lo que debemos hacer y cuál es el problema que debemos resolver.

Hay que establecer una clara estrategia de resolución. No hay que comenzar con la codificación hasta no tener definida esta estrategia.

Esto implica determinar exactamente qué sabemos y qué debemos averiguar y definir las estructuras de datos que necesitaremos para representar nuestro problema, nuestros datos y nuestra solución.

Un TDA es una abstracción. Es decir, ignoramos determinados detalles y nos concentramos en los que nos interesan.

Programación II. Algoritmos y EDD II. 2024

Resumen

Es necesario saber exactamente hacia dónde uno va antes de ponerse en marcha. En otras palabras, hay que comprender exactamente qué es lo que debemos hacer y cuál es el problema que debemos resolver.

Hay que establecer una clara estrategia de resolución. No hay que comenzar con la codificación hasta no tener definida esta estrategia.

Esto implica determinar exactamente qué sabemos y qué debemos averiguar y definir las estructuras de datos que necesitaremos para representar nuestro problema, nuestros datos y nuestra solución.

Un TDA es una abstracción. Es decir, ignoramos determinados detalles y nos concentramos en los que nos interesan.

En el caso de un TDA, no nos interesa su implementación sino sus métodos.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Comenzaremos por especificar **PilaTDA**.

Recordemos que una pila permite almacenar un conjunto de valores, recuperarlos y eliminarlos. Su particularidad es que el elemento que se recupera o se elimina es siempre el último que ingresó (es una estructura LIFO.) Adicionalmente, podemos saber cuál es el elemento más reciente y si la pila está vacía o no.

Los métodos son los siguientes:

Apilar , que permite agregar un elemento a una pila inicializada.

Desapilar , que permite eliminar el último elemento que ingresó a una pila inicializada y no vacía.

Tope, que permite conocer el último elemento que ingresó a una pila inicializada y no vacía.

PilaVacia, que permite saber si una pila inicializada está vacía o no.

InicializarPila, que permite, como es de imaginarse, inicializar la estructura de la pila.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

La interfaz de PilaTDA es la siguiente:

```
1. public interface PilaTDA; {
2.    void InicializarPila();     // sin precondiciones
3.    void Apilar (int x);        // pila inicializada
4.    void Desapilar ();          // pila inicializada y no vacía
5.    boolean PilaVacia();        // pila inicializada
6.    int Tope();   // pila inicializada y no vacía
7.     }
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Ejemplo: pasaje de los elementos de una pila *Origen* a otra pila *Destino*.

```
1. public void PasarPila (PilaTDA Origen, PilaTDA Destino) {
2.    while (!Origen.PilaVacia()) {
3.       Destino.Apilar(Origen.Tope());
4.       Origen.Desapilar();
2.    }
4. }
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Vamos a ver diferentes implementaciones de los TDA definidos anteriormente, utilizando como estrategia de implementación estructuras de tamaño fijo, es decir, **arreglos**.

Mas adelante ser verá el **costo** asociado a cada una de las operaciones de cada TDA de acuerdo a la implementación realizada.

Programación II. Algoritmos y EDD II. 2024 Implementaciones TDA

Cuando hablamos de **implementación** nos estamos refiriendo a que debemos definir los siguientes puntos:

- En dónde se va a guardar la información,
- De qué manera se va guardar esa información, es decir, definir el criterio con que se va a guardar y recuperar los datos,
- Escribir / implementar cada una de las operaciones del TDA que se esta implementando.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Para implementar un TDA en Java:

Se implementa una clase, a la que le indicaremos que es la implementación de un TDA con la palabra reservada implements.

Luego definiremos la estructura de datos, y por último,

Codificaremos cada uno de los métodos especificados.

Si fuese necesario, se podrán agregar métodos auxiliares, que serán internos a la clase.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

```
public class CLASE implements claseTDA{

}
```

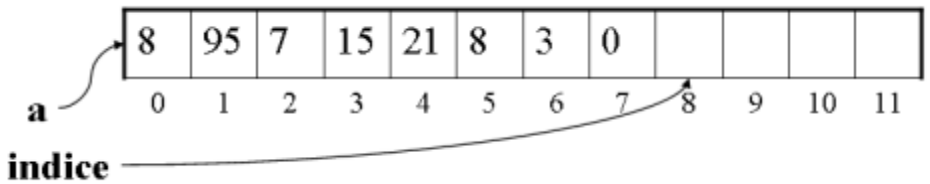
Programación II. Algoritmos y EDD II. 2024 Implementaciones TDA

Una estrategia:

- Implementación en donde se guardan los datos en un arreglo y además se tiene una variable que indica la cantidad de elementos que se tienen guardados en la pila.
- Cuando agregamos un nuevo elemento a la pila, el mismo se guarda en la posición indicada por la variable que me indica la cantidad de elementos.
- Cuando se tiene que desapilar un elemento de la pila, solo es necesario decrementar en una unidad la variable que me indica la cantidad de elementos.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Una estrategia:



Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Métodos:

```
public class PilaTF implements PilaTDA{

    int[] a;  //arreglo en donde se guarda la información
    int indice; //variable entera en donde se guarda la cantidad
       de elementos que se tienen guardados

    public void InicializarPila(){
        a = new int[100];
        indice = 0;
    }

    public void Apilar(int x){
        a[indice] = x;
        indice++;
    }

    public void Desapilar(){
        indice--;
    }
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Métodos:

```
    public boolean PilaVacia(){
        return (indice == 0);
    }

    public int Tope(){
        return a[indice-1];
    }
}
```

Programación II. Algoritmos y EDD II. 2024 Implementaciones TDA

Variante de implementación:
Utilizar también un arreglo para almacenar los datos y tener una variable que indica la cantidad de elementos que se tienen guardados en la pila, pero cuando agregamos un nuevo elemento a la pila, en vez de hacerlo en la posición señalada por la variable que me indica la cantidad de elementos, se guarda en la primera posición (es decir, la posición 0 del arreglo).

Desventaja:
- cuando se agrega un elemento se debe hacer un corrimiento del resto de los elementos hacia la derecha,
- cuando se desapila un elemento de la pila nuevamente se tiene que hacer un corrimiento de elementos pero esta vez hacia la izquierda.
- la variante entre esta implementación y la anterior no esta dada en la estructura utilizada para guardar los elementos sino en el criterio utilizado para almacenar esa información.

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Variante de implementación:

```
public class PilaTI implements PilaTDA{
    int[] a;  //arreglo en donde se guarda la información
    int indice; //variable entera en donde se guarda la cantidad
       de elementos que se tienen guardados

    public void InicializarPila(){
        a = new int[100];
        indice = 0;
    }

    public void Apilar(int x){
        for (int i = indice-1; i >=0; i--)
            a[i+1] = a[i];
        a[0] = x;
        indice++;
    }

    public void Desapilar(){
        for (int i = 0; i <indice; i++)
            a[i] = a[i+1];
        indice--;
    }

    public boolean PilaVacia(){
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Variante de implementación:

```
            return (indice == 0);
    }
    public int Tope(){
        return a[0];
    }
}
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Variante de implementación:

```
            return (indice == 0);
    }
    public int Tope(){
        return a[0];
    }
}
```

Programación II. Algoritmos y EDD II. 2024
Implementaciones TDA

Tarea:

- Implementación TP1
- Implementación TDA PILA