Tema 1. Tipo Abstracto de Datos (TAD)

Estructura de Datos y Algoritmos (EDA)

Objetivos

- Entender el concepto de **abstracción**
- Entender los conceptos de **Tipo Abstracto de Datos** (TAD) and **estructura de datos** y explicar la diferencia entre ambos
- Aprender **cómo especificar formalmente** un TAD
- Escribir **implementaciones correctas** para TAD simples

Contenidos

- Abstracción
- Tipo Abstracto de Datos (TAD)

Abstracción

- Centrarse en los detalles relevantes para un problema y descartar detalles irrelevantes

Student
…
NotaAcceso
ListOfSubjects

Patient
…
High
ListOfAllergies
ListOfSurgeries

Permitir definir software **(qué debe hacer** sin decir **cómo debe hacerse)**

Tipo Abstracto de Datos (TAD)

- Un modelo matemático con una colección de operaciones definidas en ese modelo
  - Ejemplo: conjunto de enteros con las operaciones de unión, intersección y diferencia

- Una estructura de datos es una representación (**implementación**) de un TAD en un lenguaje de programación. Las estructuras de datos a menudo son colecciones de variables de diferentes tipos de datos

- Un TAD puede tener varias implementaciones (estructuras de datos)

Especificación

- Un TAD puede describirse usando dos maneras diferentes:
  - **Especificación No-Formal** (usando lenguaje natural)
  - **Especificación Formal** (usando pseudo-código o incluso algún lenguaje de programación)
    - En JAVA, las interfaces se utilizan para definir los TAD

TAD complejo (Especificación informal)

Un número complejo se puede representar como un par de números: números reales e imaginarios. Este TAD implica las siguientes operaciones:

- ***sum****(Complex):*

$$(a+ib) + (c+id) = (a+c) + (b+d)i$$

- ***times****(Complex):*

$$(a+ib) * (c+id) = (ac-bd) + (ad+bc)i$$

- ***module****(): devuelve el módulo del número complejo.*

$$|a+ib| = \sqrt{(a^2+b^2)}$$

Especificación informal

Operaciones

- ***getReal*** *():* devuelve su número real

  (a+ib->a)

- ***getImag****(): devuelve su número imaginario*

  (a+ib->b)

- ***setReal****(x): toma un número x como entrada y lo asigna a su número real*
- ***setImag****(x): toma un número x como entrada y lo asigna a su número imaginario*

Especificación formal de un TAD Complejo

```
public interface iComplex {

    public float getReal();
    public float getImag();

    public void setReal(float x);
    public void setImag(float x);

    public float module();

    public iComplex sum(iComplex obj);
    public iComplex times(iComplex obj);

    public boolean isequals(iComplex obj);
}
```

Estructuras de datos: implementaciones de TAD

- En Java, las estructuras de datos se implementan como clases
- Una estructura de datos (class) debe implementar todas las operaciones definidas en su TAD (interfaz)
- Una class describe cómo se realizan las operaciones de TAD

```
public class Complex implements iComplex {

    private float real;
    private float imag;
    public Complex(float r, float i) {
        real=r;
        imag=i;
    }
    public float getReal() {
        return real;
    }
    public float getImag() {
        return imag;
    }
    public void setReal(float x) {
        real=x;
    }
    public void setImag(float x) {
        imag=x;
    }
}
```

La clase debe implementar todos los métodos definidos en la interfaz

```
public iComplex sum(iComplex obj) {
    iComplex result=new Complex(real+obj.getReal(),imag+obj.getImag());
    return result;
}

public iComplex times(iComplex obj) {
    iComplex result=new Complex(real*obj.getReal()-imag*obj.getImag(),
            real*obj.getImag()+imag*obj.getReal());
    return result;
}

public boolean isequals(iComplex obj) {
    return (real==obj.getReal() && imag==obj.getImag());
}
```

TAD Fecha (especificación informal)

- Datos para representar una Fecha (*Date*)
- **Operaciones:**
  - ***before****(Date d): devuelve verdadero si la fecha de invocación es anterior a la especificada por d, de lo contrario, devuelve falso.*
  - ***after****(Date d): devuelve verdadero si la fecha de invocación es posterior a la especificada por d, de lo contrario, devuelve falso*
  - ***compareTo****(Date d): devuelve 0 si las fechas son iguales. Devuelve -1 si la fecha de invocación es anterior a d. Devuelve 1 si la fecha de invocación es posterior a d.*

TAD Fecha (especificación informal)

- **Operaciones:**
  - ***getDay()****: devuelve el día de la fecha de invocación.*
  - ***getMonth()****: devuelve el mes de la fecha de invocación*
  - ***getYear()****: devuelve el año de la fecha de invocación*
  - ***show()****: muestra la fecha en el formato* dd-mm-aaaa

Especificación formal TAD iDate

```
public interface iDate {

    public int getDay();
    public int getMonth();
    public int getYear();

    public boolean before(iDate d);
    public boolean after(iDate d);
    public int compareTo(iDate d);
    public void show();
}
```

```java
public class Date1 implements iDate {

    int d;//day
    int m;//month
    int y;//year

    //To simplify, we suppose that the date is right
    public Date1(int d, int m, int y) {
        this.d=d;
        this.m=m;
        this.y=y;
    }

    public int getDay() {
        return d;
    }
```

```
public boolean before(iDate obj) {
    if (y<obj.getYear()) return true;
    else if (y==obj.getYear()) {
        if (m<obj.getMonth()) return true;
        else if (m==obj.getMonth()) {
            if (d<obj.getDay()) return true;
            else return false;
        } else return false;
    }
    //y>obj.getYear()
    return false;
}
```

Encuentra el código incompleto de la clase en aulaglobal. Implementa los otros métodos

Preguntas ???

Otra implementación para iDate???

Con solo 2 atributos???

```java
public class Date2 implements iDate {

    int day; //number of days from 1st January.
             //Ex: 32 is Feb 1.
    int year;

    public Date2(int d, int y) {
        day=d;
        year=y;
    }

    public int getYear() {
        return year;
    }

    //To simplify we suppose there is no leap years.
    public int getDay() {
        if (day<=31) return day;
        if (31<day && day<=59) return day-31;
        //complete
```

Encuentra el código incompleto de la clase en aulaglobal. Implementa los otros métodos.

Resumen

- Un TAD define **qué operaciones**, pero no **cómo hacerlas** (implementar)
- Un TAD puede tener diferentes implementaciones
- En Java, las **interfaces** permiten definir formalmente TAD
- En Java, las clases (**classes)** permiten implementar TAD
- Una clase (**class)** que implementa una interfaz debe implementar todos los métodos de la interfaz