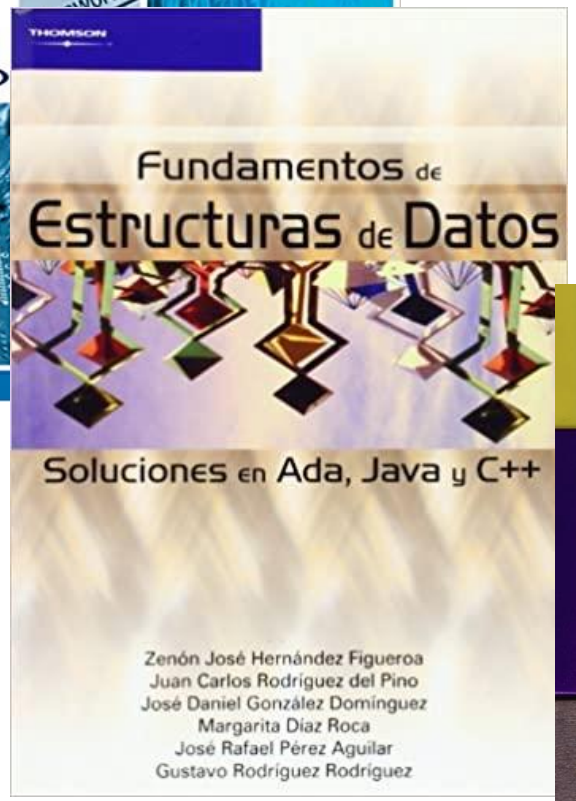
Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

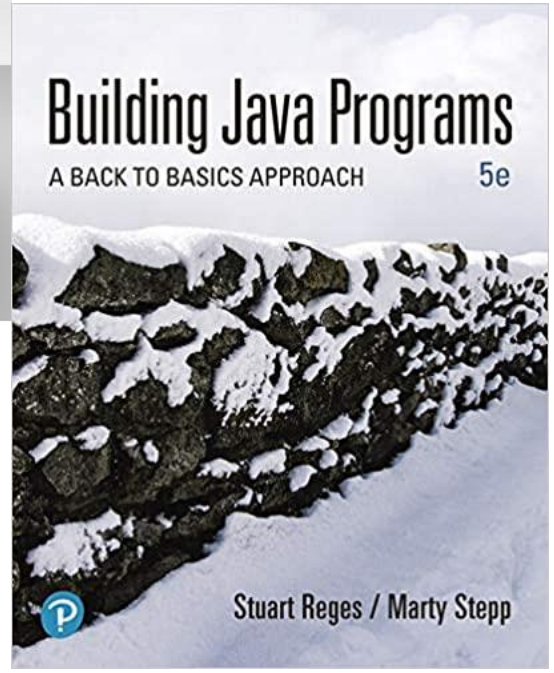
Estructura de Datos

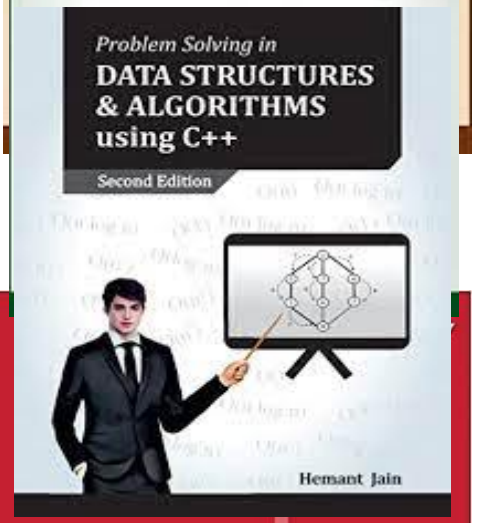
Semana 1

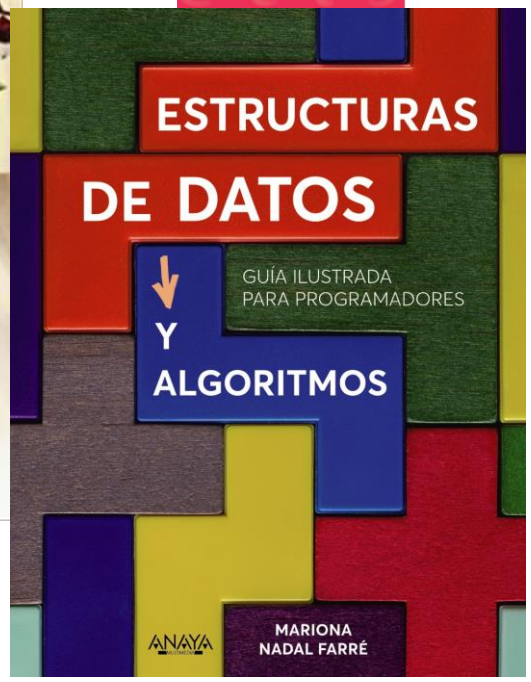
Libros Guía

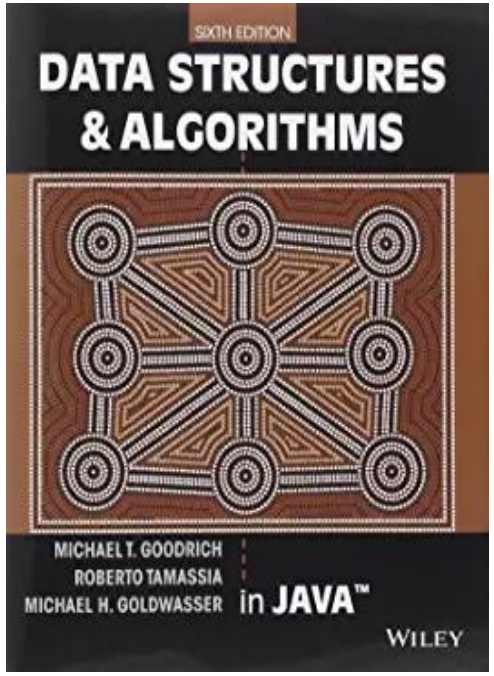


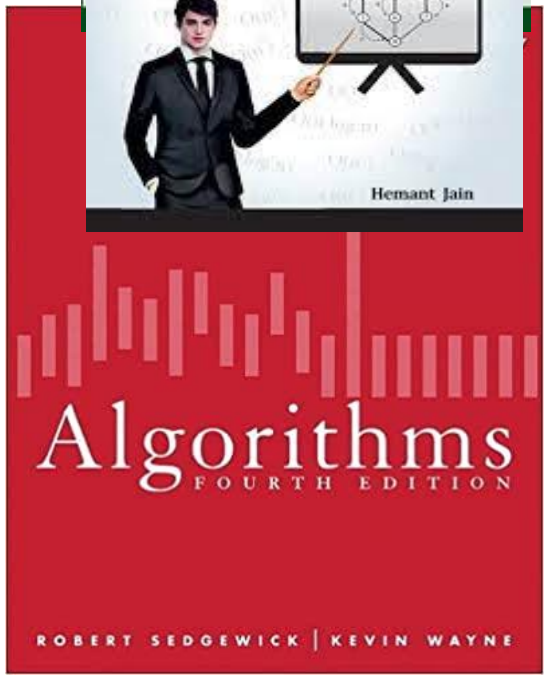
Algoritmos y estructuras de datos
con programas verificados en Dafny
2ª Edición
Ricardo Peña Marí











UNMSM

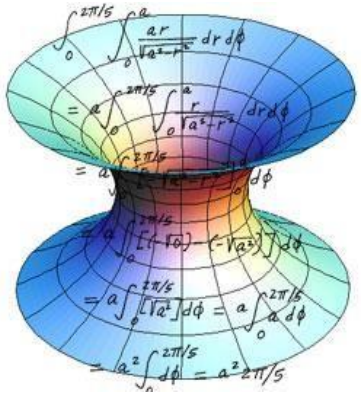
Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

Revisión del Sílabo, elección del delegado y formación de grupos

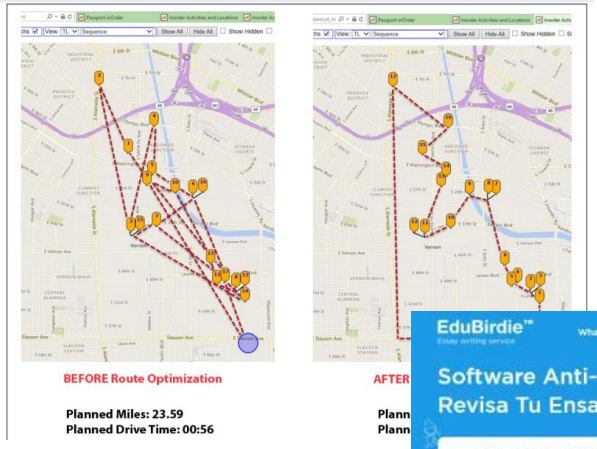
26/03/2023 – Escuela de Ingeniería de Software

Proyecto fin de curso

Calculadora científica



WorkForce Management



Sistema antiplagio

Rúbrica

Criterios	Logrado	En proceso	En inicio
Utilización de diferentes tipos de algoritmos	Al menos utiliza 3 tipos de algoritmos desarrollados en el curso. **(5 puntos)**	Sólo utiliza 2 tipos de algoritmos desarrollados en clase. **(3.5 puntos)**	Sólo utiliza un tipo de algoritmo desarrollado en clase. **(2 puntos)**
Utilización de diferentes estructuras de datos	Al menos utiliza 3 tipos de estructura de datos desarrollados en el curso. **(5 puntos)**	Sólo utiliza 2 tipos de estructuras de datos desarrollados en clase. **(3.5 puntos)**	Sólo utiliza un tipo de estructura de datos desarrollado en clase.. **(2 puntos)**
Se presenta un análisis de complejidad del código de software	Describe cuál es el nivel de complejidad temporal y espacial. **(4 puntos)**	Describe sólo un tipo de complejidad. **(3 puntos)**	No describe ningún análisis de complejidad del código de software. **(0.5 puntos)**
Comportamiento de la aplicación ante diferentes valores de entrada	El programa de software no incrementa su complejidad en forma notable cuando la entrada es muy grande. **(3 puntos)**	El programa de software incrementa su complejidad si la entrada crece a valores enormes y que pueden ocurrir en la realidad. **(1.5 puntos)**	Si la entrada crece a valores grandes y reales la aplicación no funciona correctamente. **(0.5 punto)**
Utilidad de la Aplicación	El trabajo presenta la solución a un problema existente en la realidad y los algoritmos y estructura de datos se pueden utilizar en grandes sistemas. Se puede utilizar por el usuario común **(3 puntos)**	El trabajo no presenta la solución a un problema existente en la realidad, los algoritmos y estructura de datos se pueden utilizar en grandes sistemas El trabajo se puede utilizar por un usuario común **(2 puntos)**	El trabajo no presenta la solución a un problema existente en la realidad, los algoritmos y estructura de datos no se pueden utilizar en grandes sistemas. El trabajo se puede utilizar por un usuario común. **(1 punto)**

UNMSM

Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

Logro de la sesión

Al finalizar la sesión, el estudiante:

- **Comprende los conceptos de Tipo Abstracto de Datos (TAD) y estructura de datos y explica la diferencia entre ambos conceptos**
- **Comprende el concepto de abstracción**
- **Desarrolla implementaciones correctas para TAD simples.**

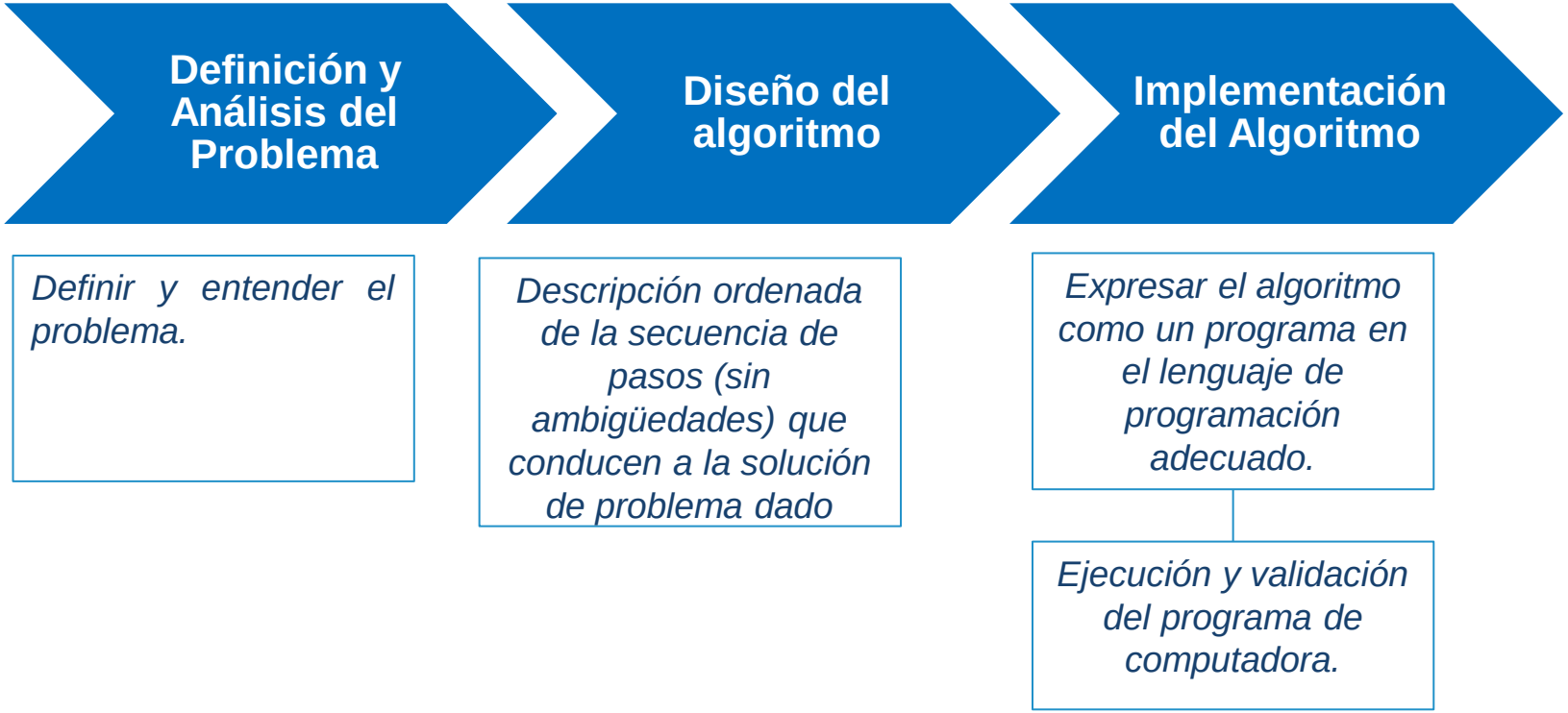
Introducción

Estructura de datos y tipo abstracto de datos

01

Introducción a las Estructuras de Datos

Metodología de programación: Consiste en la metodología aplicada para la resolución de problemas mediante programas.



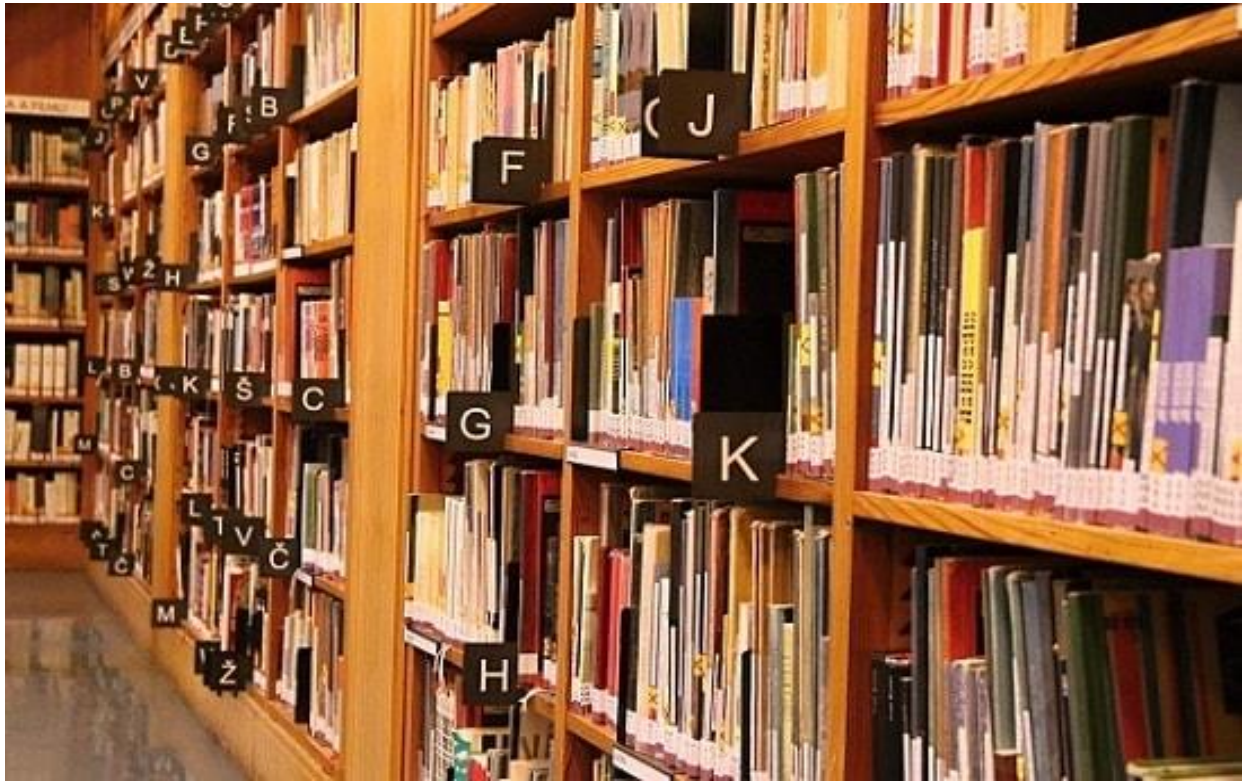
Además del algoritmo, necesitamos diseñar la forma más adecuada de organizar los datos del problema. Una **estructura de datos** es una forma particular de almacenar y organizar datos en una computadora para que pueda usarse de manera eficiente. Cuando diseña un programa para un problema dado, debe encontrar la estructura de datos más eficiente para resolverlo.

Introducción a las Estructura de Datos

¿ Buscar en la siguiente biblioteca, el libro de Estructura de Datos?

A

B



¿Escribe como buscar el libro en la?

Biblioteca A	Biblioteca B
1.	1.
2.	2.
3.	3.
4.	4.

¿Qué son las estructuras de datos?

Piensa en ellas como una forma de representar información

¿Porqué son útiles las estructuras de datos?

Representan información

Por ejemplo:

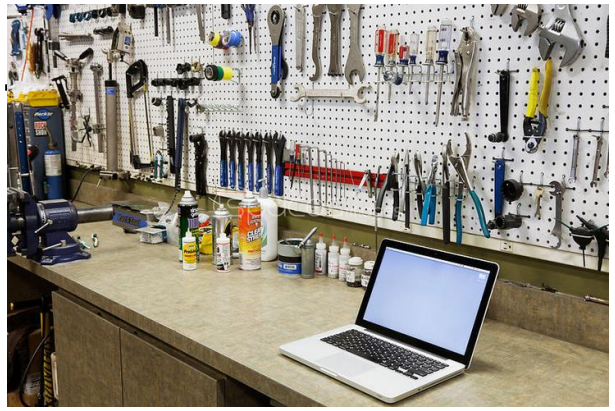
Necesito hacer un programa que apile una serie de nombre de libros como entrada, y que me diga cuál es el último libro, el problema es que vamos a apilar N libros.

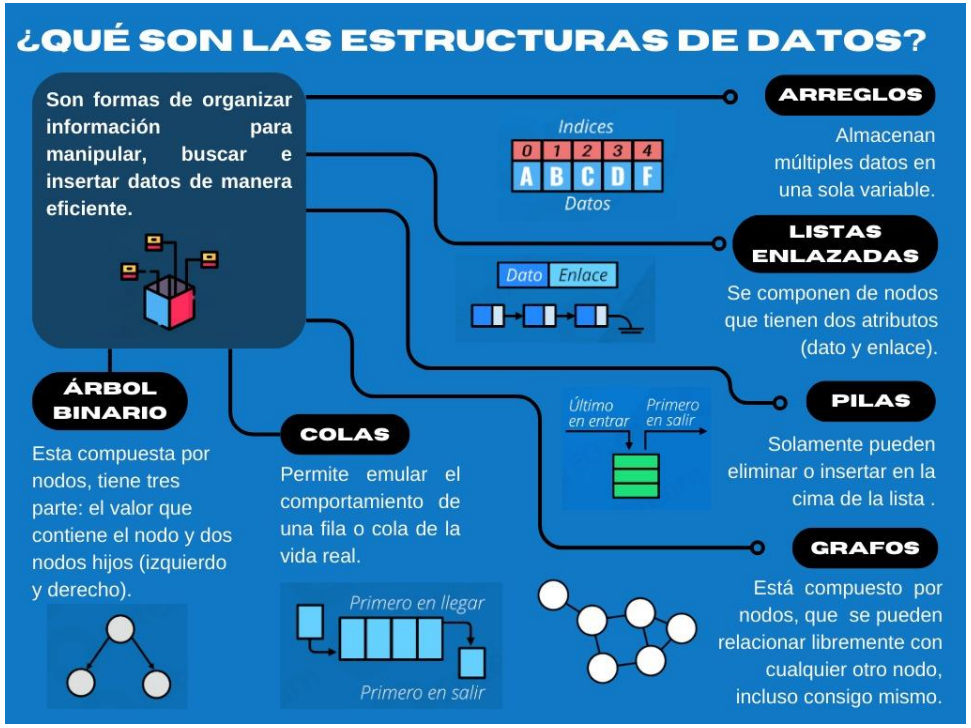
Introducción a las Estructuras de Datos

Definición

Una estructura de datos es un conjunto de elementos de información dotados de una organización. Su organización puede responder a criterios conceptuales (organización de una empresa, datos personales) o criterios prácticos (facilidad de utilizar o actualizar la información, eficiencia de los algoritmos, etc)

- Definición simple:
  - Variables que almacenan otras variables
- Aprenderemos una caja de herramientas llena de estructuras de datos …
  - … y cómo construirlos .
  - … y cómo usarlos.

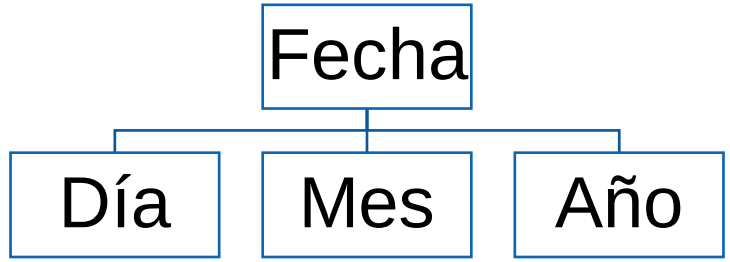




Introducción a las Estructuras de Datos

Debemos distinguir entre estructura lógica y estructura de representación, p.e. :

- Una fecha puede ser considerado como una tripleta (día, mes y año), esa es su estructura lógica



- La estructura de representación de la fecha puede ser una cadena con el formato “dd-mm-aaa”, un registro o vector de tres campos, de dos campos, un número entero que represente el # de días a partir de una fecha inicial, etc.

“06-08-2023”

(06, 02, 2004)

Introducción a las Estructuras de Datos

¿Cómo se clasifican las estructuras de datos?

Por los elementos que lo forman

- Homogéneas
  - Array en Java
- Heterogéneas
  - List en python

Por el modo en que se organizan sus elementos y se relacionan entre sí

- Lineales
  - Array
  - Listas enlazadas

  Una estructura es lineal cuando la relación entre sus componentes es 1:1, cada elemento tiene un sucesor y un predecesor único, salvo el primer y ultimo componente
- No Lineales: Jerárquicas y Multirrelacionales
  - Tree (jerarquica)
  - Graph (Multirrelacional)

  En las estructuras jerárquicas, las relaciones son 1:n (uno a muchos). En las estructuras multirrelacionales las relaciones son n:n muchos a muchos

Por la variación del # de elementos

- Estáticas
  - Array

  Es una colección ordenada y homogénea de elementos del mismo tipo y agrupados bajo el mismo nombre de una variable que no crece en tiempo de ejecución. Formada por un número fijo de elementos contiguos
- Dinámicas
  - Queue
  - Stack
  - Linked list

  Son una colección de elementos denominados nodos en donde la estructura puede aumentar o disminuir en tiempo de ejecución. Formada por un número variable de elementos no contiguos, relacionados mediante encadenamiento
- Elásticas
  - Files

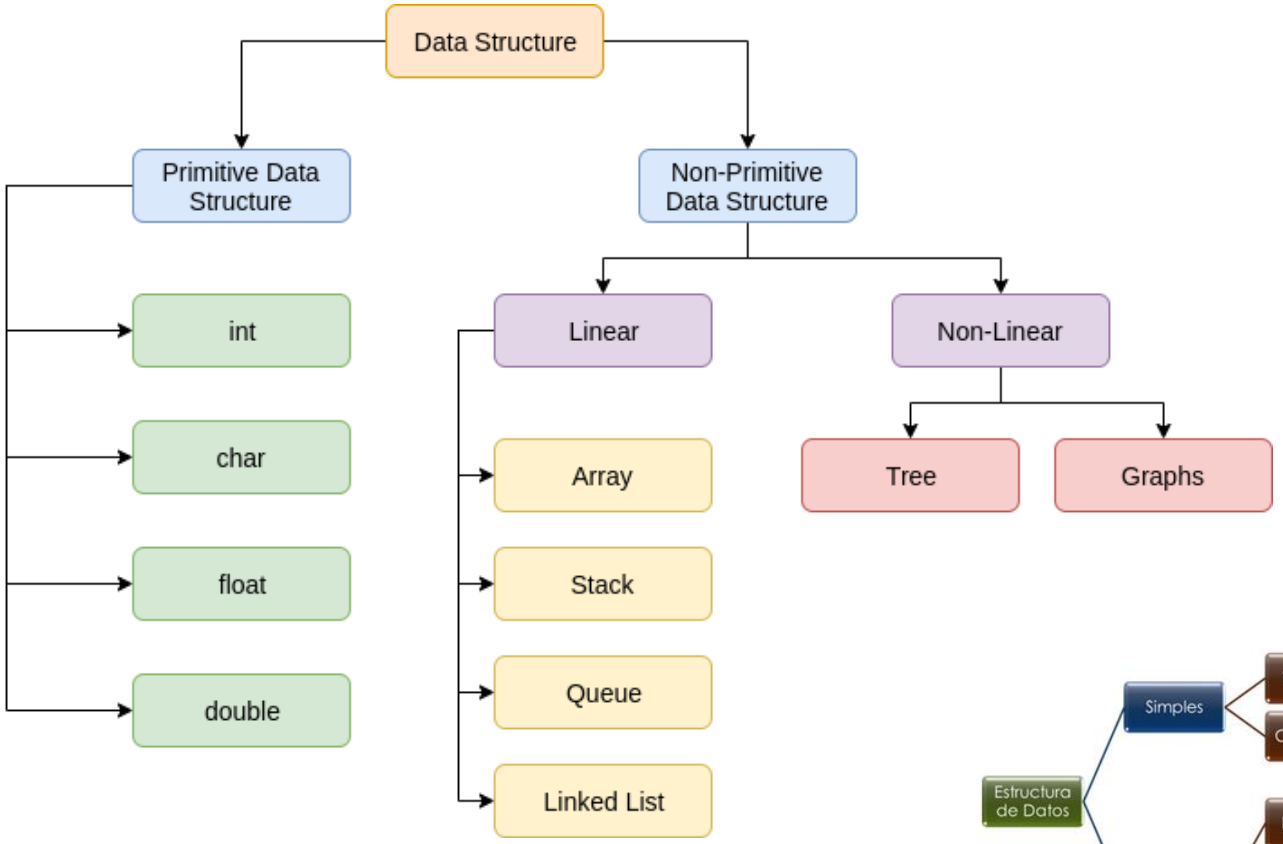
  Formada por un numero variable de elementos contiguos

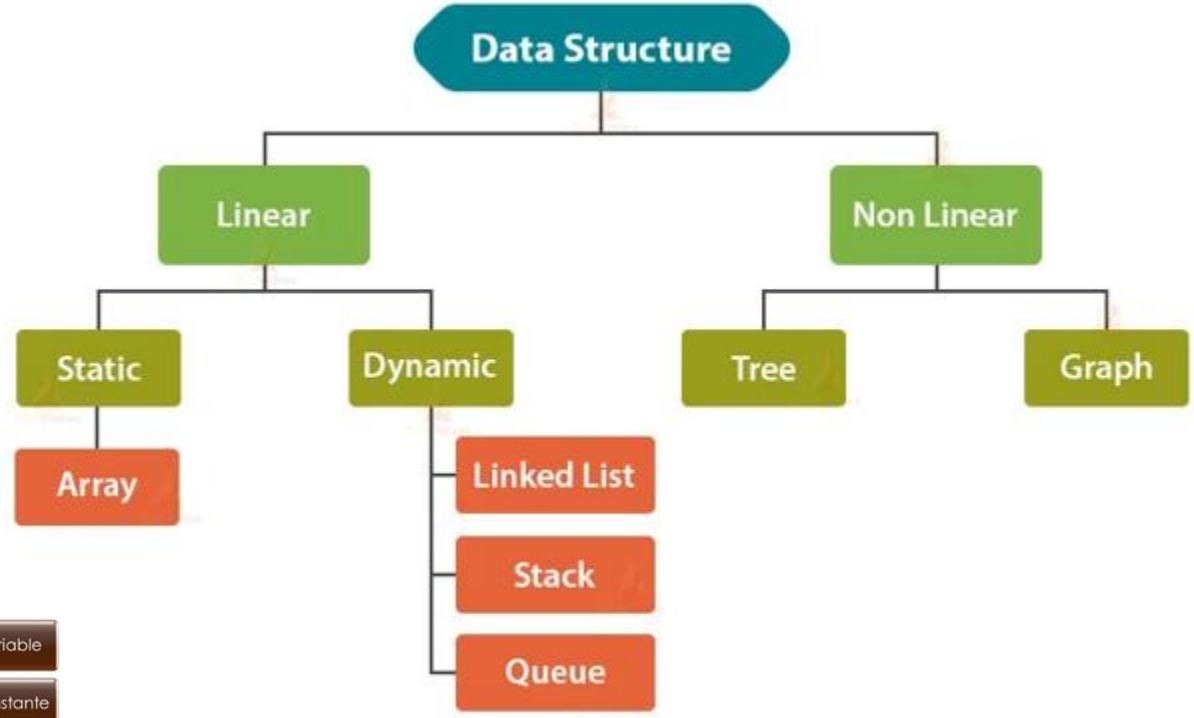
Por donde se almacenan

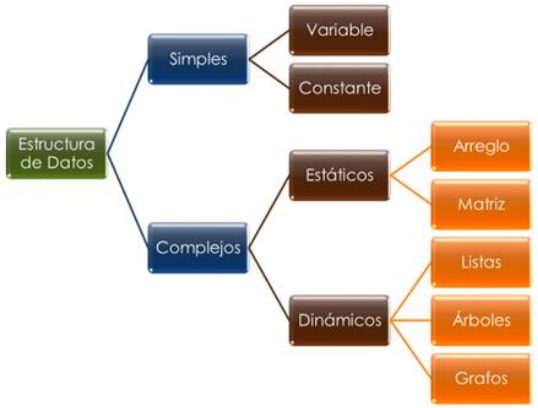
- Memoria interna
- Memoria externa

Introducción a las Estructuras de Datos

¿Cómo se clasifican las estructuras de datos?







Introducción a las Estructuras de Datos

Los lenguajes de programación proporcionan tipos primitivos de datos:

- **Tipos Simples:**

No pueden ser descompuestos en partes más pequeñas. Por ejemplo: Integer, Float, String, Boolean. Los tipos primitivos no son suficientes para resolver problemas complejos

- **Tipos Complejos:**

- **Están formados por múltiples componentes. Por ejemplo: List, Dictionary en Python. Estos son más útiles para resolver problemas, no obstante los programadores podemos definir nuestros propios tipos de datos (user-defined types).**

```
class CreditCard:
  """A credit card"""
  def __init__ (self, customer, idCard, limit):
    """Creates a new credit card object"""
    self._customer=customer #the name of the customer
    self._idCard=idCard #the id of the credit card
    self._limit=limit   #credit limit
    self._balance=0     #the initial balance is 0
```

EJERCICIO

Introducción a las Estructuras de Datos

¿Cuál es el mejor tipo básico o simple para representar si la luz está encendida o apagada?

a) Una cadena de caracteres con valores “Encendida” o “Apagada”

b) Un carácter, ‘E’ o ‘A’

c) Un booleano, cierto o falso

d) Un número 1 o 0

¿Qué es un tipo abstracto de datos?

- **Es un tipo definido por el usuario que especifica un conjunto de valores y un conjunto de operaciones para manipular dichos valores. Ejemplo: número complejo con las operaciones de suma, resta, multiplicación, y módulo**

Sea $z = a + bi$ un número complejo

Número complejo

$$z = \overbrace{5 + 4i}$$

Parte real — Parte imaginaria

$$(a + bi) + (c + di) = (a + c) + (b + d)i \quad \textbf{Suma}$$

$$(a + bi) - (c + di) = (a - c) + (b - d)i \quad \textbf{Resta}$$

$$(a + bi) + (c + di) = ac - bd + (ad + bc)i \quad \textbf{Multiplicación}$$

$$|z| = \sqrt{a^2 + b^2}$$ ***Módulo***

- **Un TAD debe ser independiente de su implementación.**
- **Un TAD se centra en qué debe hacer en lugar de cómo hacerlo.**

TIPO ABSTRACTO DE DATOS (TAD)

Introducción a las Estructuras de Datos

En el término Tipo Abstracto de Datos confluyen dos términos: tipo de datos y abstracción, hablaremos de la abstracción:

abstracción

Del lat. tardío *abstractio, -ōnis.*

1. f. Acción y efecto de abstraer o abstraerse.

abstraer Conjugar

Del lat. *abstrahĕre* 'arrastrar lejos', 'apartar, separar'.

Conjug. c. *traer.*

1. tr. Separar por medio de una operación intelectual un rasgo o una cualidad de algo para analizarlos aisladamente o considerarlos en su pura esencia o noción. U. t. c. intr. *Pensar es olvidar diferencias, es generalizar, abstraer.*

2. intr. Hacer caso omiso de algo, o dejarlo a un lado. *Centremos la atención en lo esencial abstrayendo DE consideraciones marginales.* U. t. c. prnl.

3. prnl. Concentrarse en los propios pensamientos apartando los sentidos o la mente de la realidad inmediata. *Aspirando suavemente su cigarro, se abstrajo DEL ruido de la disputa.*

4. prnl. Retirarse o recogerse, apartándose del trato social. *Abstraerse DE los negocios mundanos.*

Abstracción: Poner el foco de atención en los detalles importantes para un problema, descartando los irrelevantes

Student
NIA
email
marks
...

Patient
Birthdate
Height
Weight
ListOfAllergies
...

Introducción a las Estructuras de Datos

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)

TAD Número Complejo
Un número complejo tiene dos partes: ● La parte real (denotamos con a) ● La parte imaginaria (denotamos con b) El valor de un número complejo es: a+ib, donde i=$\sqrt{-1}$
Este TAD implica las siguientes operaciones: ● Obtener y modificar su parte real e imaginaria ● Suma y resta: ○ (a+ib) + (c+id) = (a+c) + (b+d)i ○ (a+ib) - (c+id) = (a-c) + (b-d)i ● Multiplicación: (a+ib) * (c+id) = (ac-bd) + (ad+bc)i ● Valor absoluto o módulo: |a+ib| = $\sqrt{a^2 + b^2}$ ● Comparar si es igual a otro # complejo

TAD # complejo (Especificación formal)

```
public interface numComplejo {

    public float getReal();

    public float getImag();

    public void setReal(float x);

    public void setImag(float x);

    public numComplejo suma(numComplejo obj);

    public numComplejo resta(numComplejo obj);

    public numComplejo multiplicacion(numComplejo obj);

    public float modulo();

    public boolean esigual(numComplejo obj);

}
```

TIPO ABSTRACTO DE DATOS (TAD)

Introducción a las Estructuras de Datos

- Una estructura de datos es una representación (implementación) de un TAD en un lenguaje de programación.
- La estructura de datos se centra en cómo almacenar y organizar los datos y en implementar las operaciones para manipular los datos.

- En Python, Java, las estructura de datos se implementan como clases.
- Un TAD puede tener varias implementaciones (estructuras de datos)
- Una estructura de datos (clase) debe implementar todas las operaciones definidas en su TAD (interfaz)

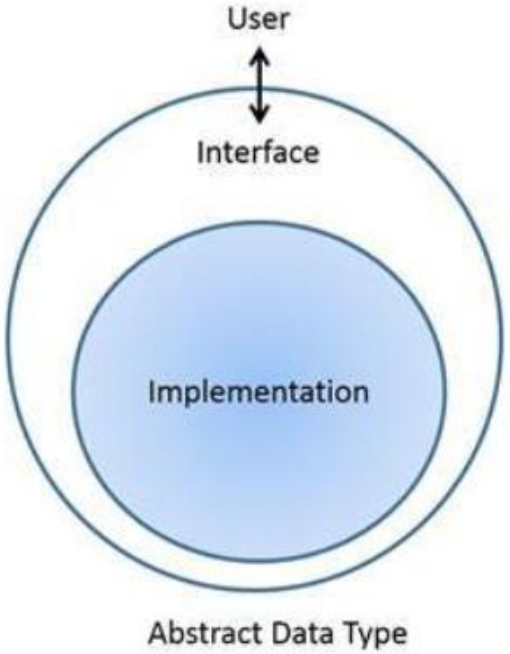
```java
public class numComplex implements numComplejo {
    private float real;
    private float imag;
    public numComplex(float r, float i) {
        real = r;
        imag = i;
    }
    @Override
    public float getReal() {
        return real;
    }
    @Override
    public float getImag() {
        return imag;
    }
    @Override
    public void setReal(float x) {
        real = x;
    }
    @Override
    public void setImag(float x) {
        imag = x;
    }
    @Override
    public numComplejo suma(numComplejo obj) {
        numComplejo result = new numComplex(real + obj.getReal(), imag + obj.getImag());
        return result;
    }
    @Override
    public numComplejo resta(numComplejo obj) {
        numComplejo result = new numComplex(real - obj.getReal(), imag - obj.getImag());
        return result;
    }
    @Override
    public numComplejo multiplicacion(numComplejo obj) {
        numComplejo result = new numComplex(real * obj.getReal() - imag * obj.getImag(),
                real * obj.getImag() + imag * obj.getReal());
        return result;
    }
    @Override
    public float modulo() {
        return (float) Math.sqrt(real * real + imag * imag);
    }
    @Override
    public boolean esigual(numComplejo obj) {
        return (real == obj.getReal() && imag == obj.getImag());
    }
}
```

UNMSM

Introducción a las Estructuras de Datos

Tipo abstracto de datos (TAD ó ADT en Inglés) : una especificación de una colección de datos y las operaciones que se pueden realizar en ellos.

- Describe *lo que* hace una colección, no *cómo* lo hace.

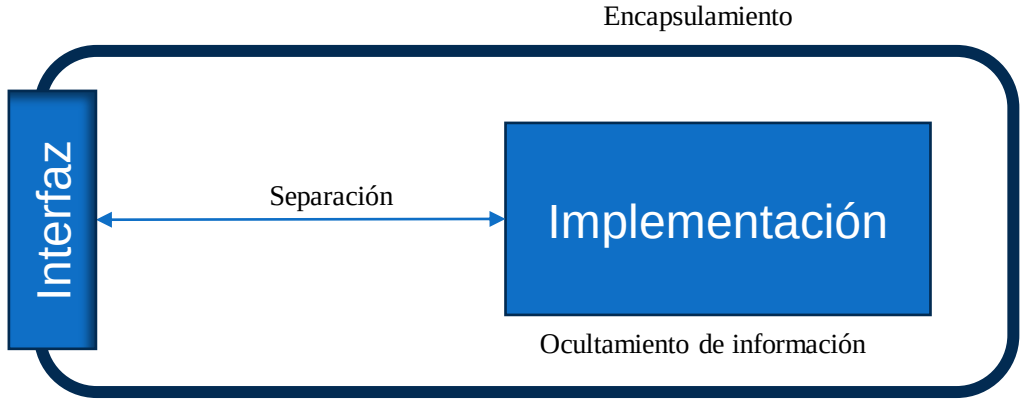


El marco de colección de Java especifica ADT con interfaces:

- `Collection`, `Deque`, `List`, `Map`, `Queue`, `Set`, `SortedMap`

Un ADT se puede implementar de múltiples maneras por clases:

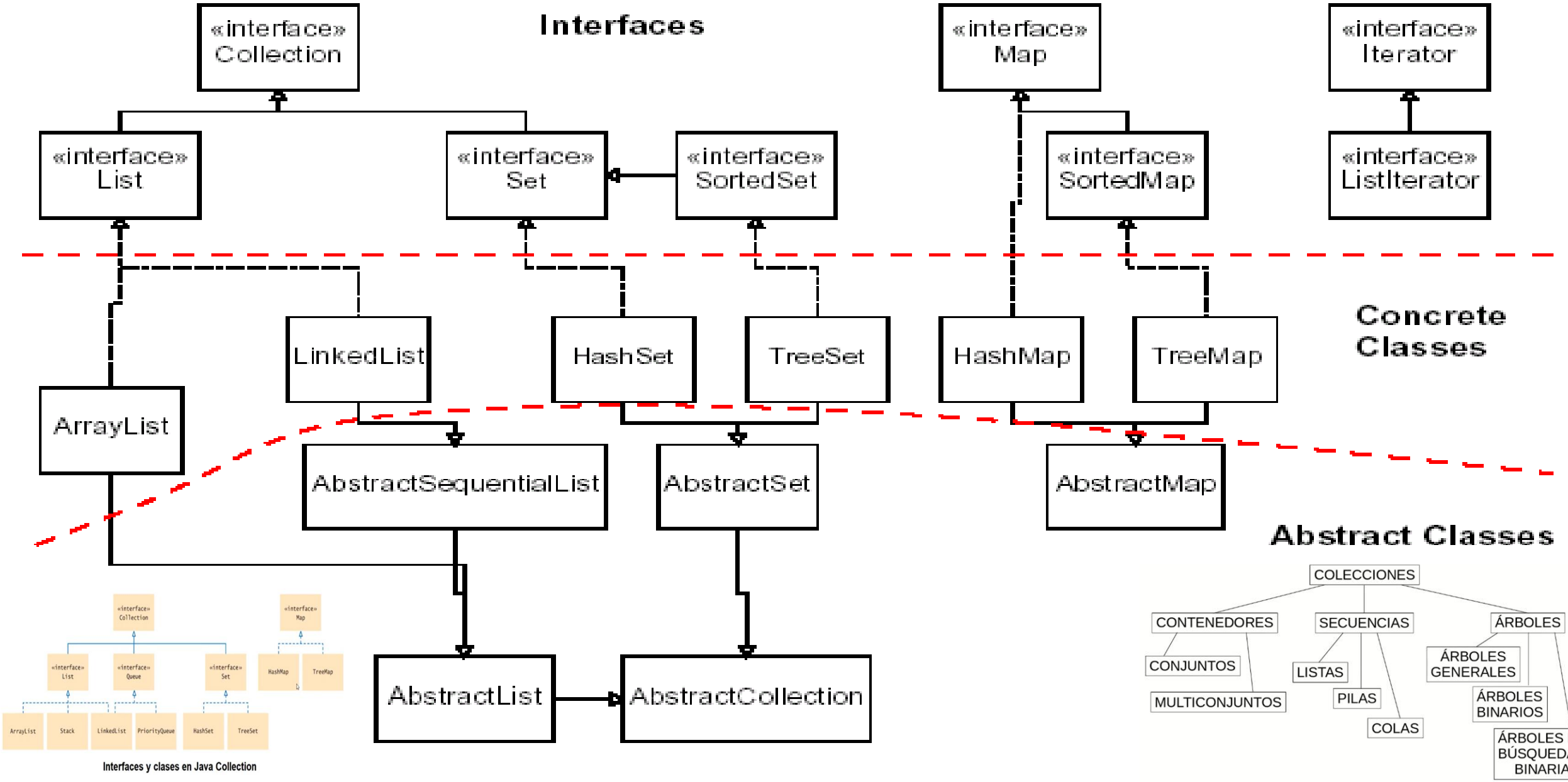
- `ArrayList` y `LinkedList` implementan `List`
- `HashSet` y `TreeSet` implementan `Set`
- `LinkedList` , `ArrayDeque` , etc. implementan `Queue`



ADT (Tipo abstracto de datos)

Un tipo abstracto de datos (ADT) es una descripción lógica de los datos y las operaciones que se permiten sobre ellos. ADT se define desde el punto de vista del usuario de los datos. ADT se preocupa por los posibles valores de los datos y la interfaz expuesta por ellos. ADT no se preocupa por la implementación real de la estructura de datos. Por ejemplo, un usuario quiere almacenar algunos números enteros y encontrar su valor medio. El ADT en este caso admitirá dos funciones, una para sumar números enteros y otra para obtener el valor medio. El ADT no habla sobre cómo se implementará exactamente. Pero la estructura de datos que representa el ADT si lo hará

Marco de colecciones Java



Interfaces y clases en Java Collection

Ejercicio – TAD Fecha

Una fecha representa un único día en nuestro calendario (por ejemplo, 4 de Agosto del 2023). El TAD se describe con una especificación no-formal. Elabore la especificación formal en Java (interface). Asimismo implemente la estructura de datos con una Clase Java y pruebe la misma

TAD Fecha (Especificación informal)

TAD Fecha
Una fecha representa un único día en nuestro calendario (por ejemplo, 4 de Agosto de 2023).
Podemos definir las siguientes operaciones para operar con fechas: ○ Date(day,month,year): crea una nueva instancia de tipo fecha. ○ day(): devuelve el día (valor numérico) de la fecha. ○ month(): devuelve el mes (valor numérico) de la fecha. ○ monthName(): devuelve el nombre del mes de la fecha. ○ year(): devuelve el año de la fecha. ○ numDays(otherDate): devuelve el número de días que hay de diferencia entre la fecha y la fecha pasada como parámetro otherDate. ○ isLeapYear(): devuelve True si el año de la fecha es bisiesto y False en otro caso. ○ compareTo(otherDate): compara la fecha con la fecha otherDate para determinar su orden lógico: ■ Si la fecha es anterior a otherDate, devuelve -1. ■ Si las dos fechas son iguales, devuelve 0. ■ Si la fecha es posterior a otherDate, returns 1. ○ str(): devuelve un string representando la fecha en el formato ‘dd/mm’/yyyy’

EJERCICIO

Introducción a las Estructuras de Datos

¿Cuál es la diferencia entre estructura de datos (ED) y tipo abstracto de datos (TAD)?

a) Las ED son representaciones lógicas y los TAD son representaciones físicas

b) Las ED son de almacenamiento permanente y los TAD son de almacenamiento durante la ejecución del programa

c) Los TAD son representaciones lógicas y las ED son representaciones físicas

d) Los TAD son de almacenamiento permanente y las ED son de almacenamiento durante la ejecución del programa

Un tipo abstracto de datos

a) Es la implementación de un tipo de datos en un lenguaje de programación.

b) Es la especificación de un nuevo tipo de datos definido por el usuario.

c) Únicamente los tipos primitivos son tipos abstractos de datos; los tipos complejos o definidos por los usuarios se denominan estructuras de datos.

Una estructura de datos

a) Es lo mismo que un tipo abstracto de datos, es decir, la definición de un nuevo tipo de datos, que es siempre independiente del lenguaje de programación.

b) Un tipo de abstracto de datos tiene siempre una única estructura de datos para representarlo.

c) Es la implementación de un tipo de datos en un lenguaje de programación.

La definición de un tipo abstracto de datos

a) Se debe hacer en un lenguaje de programación como Python.

b) Incluye el conjunto de datos y valores y las operaciones que se pueden realizar sobre esos valores.

c) Es lo mismo que su implementación

La abstracción

a) Consiste en el desarrollo de una clase de Python que representa un tipo abstracto de datos.

b) Debe considerar el máximo de información posible respecto a una entidad, sea o no de interés para el problema a resolver. En la fase de codificación, ya nos centraremos en los aspectos relevantes al problema.

c) Es el proceso de focalizarnos en los aspectos importantes de un problema, ignorando el resto de detalles.

Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

RESUMEN

Introducción a las Estructuras de Datos

- Una estructura de datos es un conjunto de elementos de información dotado de una organización.
- Las estructuras de datos pueden ser homogéneas/heterogéneas, lineales o no-lineales, estáticas/dinámicas/elásticas, estructuras de datos en memoria interna/externa.
- Un Tipo Abstracto de Datos (TAD) define qué valores y operaciones tendrá, pero no cómo implementarlos.
- Una Estructura de Datos es la implementación de un TAD (cómo). Un TAD puede tener diferentes implementaciones.
- En Java, las interfaces permiten definir formalmente TAD. En Java, las clases permiten implementar un TAD.

Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América