[Estructuras de Datos]

TDA COMPARABLE.

TDA COMPARATOR.

Copyright

TDA COMPARABLE
vs.
TDA COMPARATOR

Se recomienda que todo lo documentado en las siguientes secciones sea complementado a través de otras herramientas multimedia dispuestas en la siguiente explicación online.

Diferencia entre Comparable y Comparator

- Los conceptos **comparable** y **comparador** surgen a partir de un concepto más general que es la **comparación de tipos de datos**.
- Un **TDA** que **puede compararse**, se dice que es **comparable.**
- La forma en la que se **implementa** una comparación entre dos objetos instancia de un **TDA**, puede definirse en un **comparador.**

Comparable → Adjetivo → *Palabra que acompaña al sustantivo para* **expresar una cualidad.**

Comparador → Sustantivo → *Que tiene* **existencia real** *e independiente.*

Este ***rectángulo*** *es* ***comparable****, lo que implica* ***que permite ser comparado*** con otro elemento de su mismo tipo.

Un ***comparador*** *permite implementar la forma en la que se* ***comparan*** *dos* ***rectángulos:***

1) *Uno podría comparar dos rectángulos en relación a su perímetro.*
2) *Otro podría comparar dos rectángulos en relación a su área.*

TDA COMPARABLE

Se recomienda que todo lo documentado en las siguientes secciones sea complementado a través de otras herramientas multimedia dispuestas en la siguiente explicación online.

¿Qué define un TDA Comparable?

- Un tipo de dato se dice **comparable**, si permite ser **comparado** con otro de su mismo tipo.

Interface Comparable<E>
+ compareTo(E o): int

```
public interface Comparable<E> {
  public int compareTo(E o2);
}
```

En la materia utilizaremos el **TDA Comparable<E>** definido por **Java**.

La operación retorna un valor N tal que, considerando O1 el objeto que recibe el mensaje:

- *N = 0, si O1 es igual a O2.*
- *N > 0, si O1 es mayor a O2.*
- *N < 0, si O1 es menor a O2.*

TDA Comparable :: Ejemplo

```
public class Persona implements Comparable<Persona> {
  protected int dni, edad;
  protected float peso;

  public Persona(int d, int e, float p) {
    dni = d; edad = e; peso = p;
  }

  public int getDni() { return dni; }
  public int getEdad() { return edad; }
  public float getPeso() { return peso; }

  public int compareTo(Persona p1) {
    return dni - p1.getDni();
  }
}
```

*La clase **Persona** es un tipo de dato **Comparable**, por lo que deberá implementar un método **compare()** que permita comparar dos instancias de esta clase.*

*La operación retorna un valor N tal que, considerando p el objeto **Persona** que **recibe el mensaje**:*

- *N = 0, si p tiene igual dni que p1.*
- *N > 0, si p tiene mayor dni que p1.*
- *N < 0, si p tiene menor dni que p1.*

Todo dato **Comparable** permite ser **comparado** con otro dato de su mismo tipo, según un **criterio especificado**. Más allá de esto, ¿qué sucede, por ejemplo, si se desea comparar **Personas** por edad o por peso en lugar de por dni?

COMPARADORES

Se recomienda que todo lo documentado en las siguientes secciones sea complementado a través de otras herramientas multimedia dispuestas en la siguiente explicación online.

¿Qué es un comparador?

- A lo largo de la implementación de ciertas EDs o TDAs, resulta **importante comparar** datos para determinar si son **iguales**, **mayores** o **menores** que otros.
  - En la ED ***ABB*** es relevante la **relación de orden** entre dos o más **elementos**.
  - En el TDA ***Cola Con Prioridad*** es relevante la relación de orden entre las **claves** de sus entradas.
- La **comparación** entre **tipos de datos elementales** (p.e. *int*), o de **tipos de datos comparables** resulta directa.
- Sin embargo la comparación de tipos de datos **genéricos** no resulta evidente.

TDA Comparator

- Un **comparador**, es un tipo de dato que ofrece un método que permite **comparar dos elementos** del mismo tipo.

Interface Comparator<E>
+ compare(E o1, Eo2): int

```
public interface Comparator<E> {
  public int compare(E o1, E o2);
}
```

La operación retorna un valor N tal que:
- *N = o, si O1 es igual a O2.*
- *N > o, si O1 es mayor a O2.*
- *N < o, si O1 es menor a O2.*

En la materia utilizaremos el **TDA Comparator<E>** definido por **Java**.

TDA Comparator :: Ejemplos

```
public class Comp_pers_edad implements Comparator<Persona>{

  public int compare(Persona p, Persona p1) {
    return p.getEdad() – p1.getEdad();
  }
}
```

Esta comparador retorna un valor N tal que, considerando dos personas p1 y p2:

- *N = 0, si p tiene igual edad que p1.*
- *N > 0, si p tiene mayor edad que p1.*
- *N < 0, si p tiene menor edad que p1.*

Los bucles ifs anidados permiten retornar exactamente el mismo resultado que la siguiente expresión:

- *return Math.round(p.getPeso() - p1.getPeso());*

```
public class Comp_pers_peso implements Comparator<Persona>{

  public int compare(Persona p, Persona p1) {
    int to_return = 0;

    if (p.getPeso() < p1.getPeso()) {
      to_return = -1;
    }else {
      if (p.getPeso() > p1.getPeso()) {
        to_return = 1;
      }
    }
    return to_return;
  }
}
```

Esta comparador retorna un valor N tal que, considerando dos personas p y p1:

- *N = 0, si p tiene igual peso que p1.*
- *N > 0, si p tiene mayor peso que p1.*
- *N < 0, si p tiene menor peso que p1.*

Comparadores por defecto

- Se pueden desarrollar **comparadores por defecto** teniendo en cuenta elementos **comparables**.
- El **comparador por defecto** delega la responsabilidad de la operación *compare()*, en la operación *compareTo()* del tipo de dato **comparable**.

```
public class DefaultComparator<E extends Comparable<E>> implements Comparator<E> {

  @Override
  public int compare(E o1, E o2) {
    return o1.compareTo(o2);
  }
}
```

*Como el tipo de dato E **necesariamente** debe ser **Comparable**, es posible utilizar la operación **compareTo(o2)**.*

*Como la clase implementa un **Comparador**, deberá implementar la operación **compare(o1,o2)**.*

Fin de la presentación.