TEMA 2

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

- Son secuencias de caracteres de cualquier longitud
- Tipos:
  - Constantes (inmutables): cuyos contenido no varía (ej.: el nombre de una persona, de un mes, …).
    - Está implementadas bajo la clase **String** sin contemplar métodos que modifiquen el estado del objeto.
    - Cambiar el contenido del **String** representa cambiar de objeto
  - Variables (mutables): cuyos caracteres pueden variar (ej.: membrete de una carta, …).
    - Están implementadas bajo la clase **StringBuffer** con métodos que contemplan la modificación del objeto.

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

- CLASE: String ()
  - public String()
  - public String(char[])
  - public String(char[], int, int)
  - public String(int[], int, int)
  - public String(byte[], int, int, int)
  - public String(byte[], int)
  - public String(byte[], int, int)
  - public String(byte[])
  - public String(StringBuffer)
  - public String(StringBuilder)
  - String(int, int, char[])
  - public int length()
  - public boolean isEmpty()
  - public char charAt(int)
  - public int codePointAt(int)
  - public int codePointBefore(int)
  - public int codePointCount(int, int)
  - public int offsetByCodePoints(int, int)
  - void getChars(char[], int)
  - public void getChars(int, int, char[], int)
  - public void getBytes(int, int, byte[], int)
  - public byte[] getBytes()
  - public boolean contentEquals (StringBuffer)
  - public boolean equalsIgnoreCase (String)
  - public int compareTo(String)
  - public int compareToIgnoreCase (String)
  - public boolean regionMatches (int, String, int, int)
  - public boolean regionMatches (boolean, int, String, int, int)
  - public boolean startsWith(String, int)
  - public boolean startsWith(String)

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

- CLASE: String ()
  - public boolean endsWith(String)
  - public int hashCode()
  - public int indexOf(int)
  - public int indexOf(int, int)
  - public int lastIndexOf(int)
  - public int lastIndexOf(int, int)
  - public int indexOf(String)
  - public int indexOf(String, int)
  - static int indexOf(char[], int, int, char[], int, int, int)
  - public int lastIndexOf(String)
  - public int lastIndexOf(String, int)
  - static int lastIndexOf(char[], int, int, char[], int, int, int)
  - public String substring(int)
  - public String substring(int, int)
  - public CharSequence subSequence(int, int)
  - public String concat(String)
  - public String replace(char, char)
  - public boolean matches(String)
  - public String replaceFirst(String, String)
  - public String replaceAll(String, String)
  - public String[] split(String, int)
  - public String[] split(String)
  - public String toLowerCase()
  - public String toUpperCase()
  - public String trim()
  - public String toString()
  - public char[] toCharArray()
  - public static String valueOf(char[])
  - public static String valueOf(char[], int, int)
  - public static String copyValueOf(char[], int, int)
  - public static String copyValueOf(char[])
  - public static String valueOf(boolean)
  - public static String valueOf(char)
  - public static String valueOf(int)
  - public static String valueOf(long)
  - public static String valueOf(float)
  - public static String valueOf(double)

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

- CLASE: StringBuffer ()
  - public StringBuffer()
  - public StringBuffer(int)
  - public StringBuffer(String)
  - public int length()
  - public int capacity()
  - public void ensureCapacity(int)
  - public void trimToSize()
  - public void setLength(int)
  - public char charAt(int)
  - public int codePointAt(int)
  - public int codePointBefore(int)
  - public int codePointCount(int, int)
  - public int offsetByCodePoints(int, int)
  - public void getChars(int, int, char[], int)
  - public void setCharAt(int, char)
  - public StringBuffer append(String)
  - public StringBuffer append(StringBuffer)
  - public StringBuffer append(char[])
  - public StringBuffer append(char[], int, int)
  - public StringBuffer append(boolean)
  - public StringBuffer append(char)
  - public StringBuffer append(int)
  - public StringBuffer appendCodePoint(int)
  - public StringBuffer append(long)
  - public StringBuffer append(float)
  - public StringBuffer append(double)
  - public StringBuffer delete(int, int)
  - public StringBuffer
  - deleteCharAt(int)
  - public StringBuffer replace(int, int, String)
  - public String substring(int)
  - public CharSequence subSequence(int, int)
  - public String substring(int, int)
  - public StringBuffer insert(int, char[], int, int)
  - public StringBuffer insert(int, String)
  - public StringBuffer insert(int, char[])

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

- CLASE: StringBuffer ()
  - public StringBuffer insert(int, boolean)
  - public StringBuffer insert(int, char)
  - public StringBuffer insert(int, int)
  - public StringBuffer insert(int, long)
  - public StringBuffer insert(int, float)
  - public StringBuffer insert(int, double)
  - public int indexOf(String)
  - public int indexOf(String, int)
  - public int lastIndexOf(String)
  - public int lastIndexOf(String, int)
  - public StringBuffer reverse()
  - public String toString()

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Cadenas de caracteres

Literales String

- es la única clase cuyos objetos pueden presentarse en forma literal;
- su formato es una secuencia de caracteres de cualquier longitud encerrados entre comillas

```
"<carácter>…"
```

- son objetos de la clase **String** cuya evaluación devuelve la dirección del objeto al que representan;
- además, es la única clase que disfruta de un operador (+) para la concatenación de cadenas de caracteres y combinado con cualquier tipo.

```
int longitud = "caracteres".length();
String cadena = new String("caracteres");

boolean falso = cadena == "caracteres";
boolean cierto = cadena.equals("caracteres");
boolean tambien = "caracteres".equals(cadena);
StringBuffer buffe = newStringBuffer (cadena);

buffer.insert(0, "de ").insert(0, "cadena ");

boolean si = ("cadena de " + cadena).
             contentEquals(buffer);

String serie = (0+1)+", "+(1+1)+","+(2+1)+".";
```

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

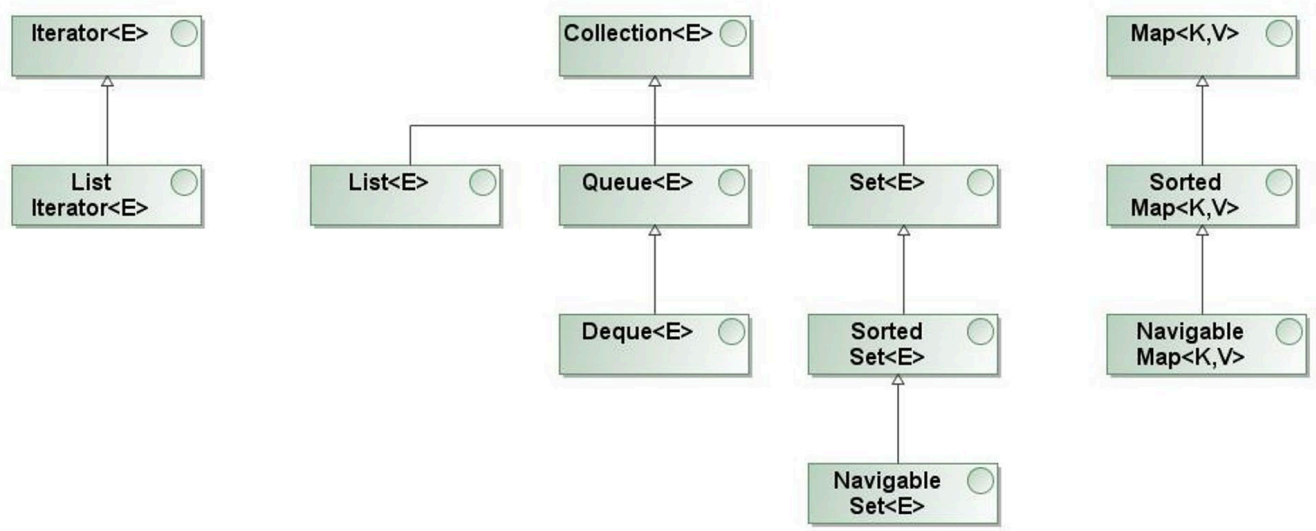
Programación **PARAMETRIZADA**

- Las clases parametrizadas se denominan clases genéricas; y los métodos parametrizados se denominan métodos genéricos.
- Los genéricos son útiles para definir clases cuyos atributos puedan ser de cualquier clase, y para definir métodos que puedan recibir argumentos y devolver resultados de cualquier clase.
- La mayoría de las estructuras básicas estándar (colas, listas, pilas, …) están ya implementadas como colecciones con este sistema y pueden usarse para mejorar el control y mantenimiento sin hacer uso de los vectores estándar (array).
- Estas ***colecciones*** ya tienen optimizados los procesos de inserción, ordenación, acceso y borrado por lo que es importante conocerlas y saber cual es la mas adecuada al problema y su necesidad para valorar que opción es la mas optima en cada caso.

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones

- Colecciones: Java proporciona en el paquete java.util un conjunto de clases para representar y manejar estructuras de datos.
- Provienen de tres grandes familias:

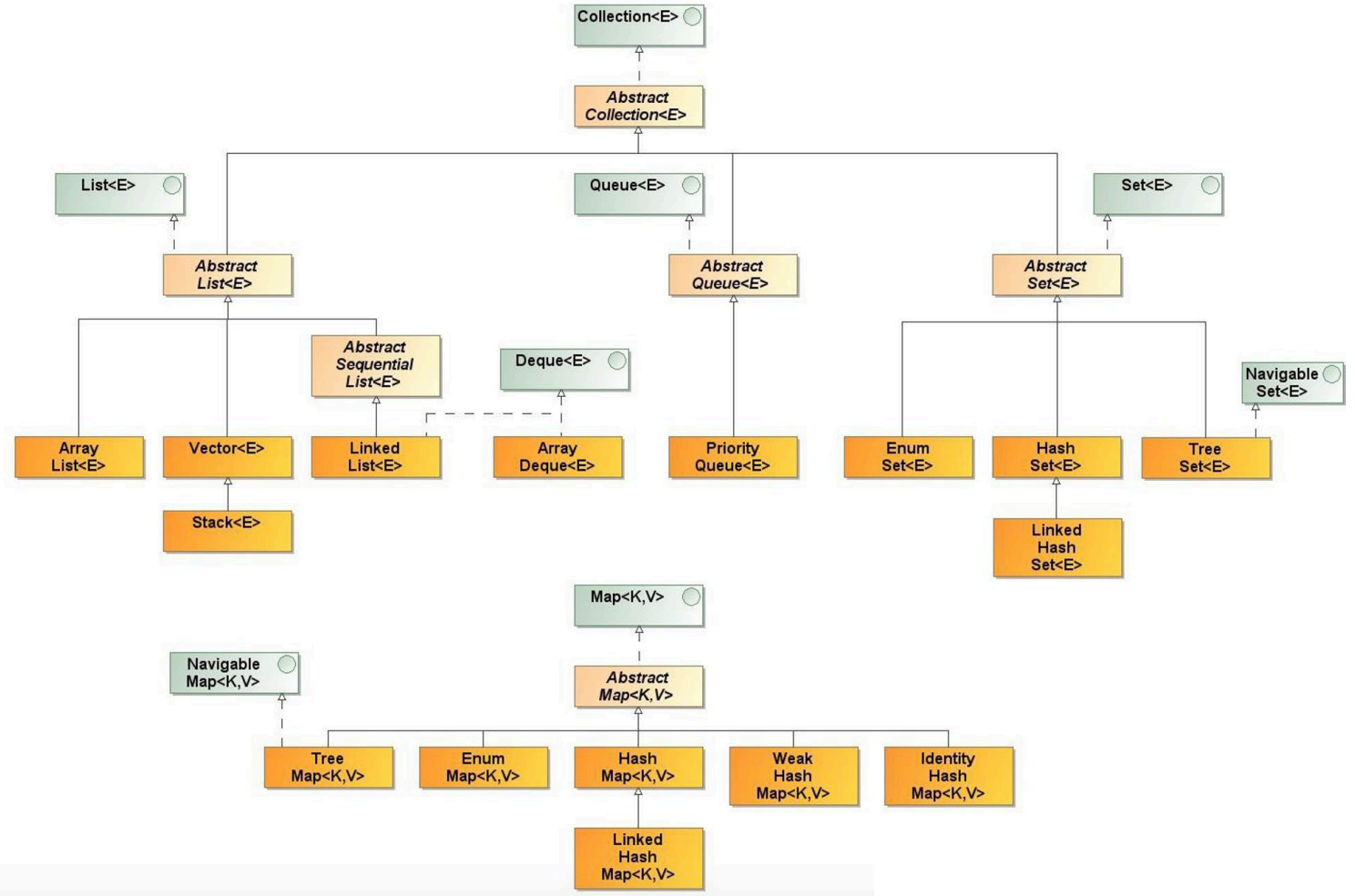


5.3 PROGRAMACIÓN PARAMETRIZADA

Colecciones

Array List<E>

Usables directente.



2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones Principales

- Clase ArrayList<E>: Implementación de lista que proporciona basada en vectores de elementos que autoajustan su tamaño según se añaden elementos.
- Clase LinkedList<E>: Implementación de lista que utiliza listas doblemente enlazadas. Además, implementa un Deque (**Double-Ended Queue**) o cola de dos extremos, por lo que puede usarse para tratar con colas y pilas en simultaneo.
- Clase Vector<E>: Implementación de un vector autoajustables, y se mantiene por razones de compatibilidad con código heredado. Además está sincronizada para evitar accesos concurrentes, lo que conlleva un cierto sobrecoste adicional.
- Calse Stack<E>: Representa una pila LIFO (*Last In, First Out*).
- Clase PriorityQueue<E>: Implementación de una pila de prioridades basada en prioridades que se establecen por el orden natural de sus elementos (***el objeto deberá ser Comparable***) o por un comparador de elementos que se suministra a la cola en su construcción (***en el constructor sobrecargado***) .

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones Principales

- Clase ArrayDeque<E>: Implementación de una cola de dos extremos mediante vectores autoajustables.
- Clase HashSet<E>: Implementación mediante una tabla hash (realmente un HashMap). No garantiza ningún orden al iterar sobre el conjunto.
- Clase LinkedHashSet<E>: Implementación mediante tabla hash y listas enlazadas (mediante LinkedHashMap), en la que los elementos se recorren en el mismo orden en el que se insertaron.
- Clase TreeSet<E>: Implementación mediante TreeMap, en la que los elementos se recorren según su orden natural.
- Clase HashMap<K,V>: Implementación mediante una tabla hash. No garantiza ningún orden al iterar sobre el mapa.
- Clase LinkedHashMap<K,V>: Implementación mediante una tabla hash y listas doblemente enlazadas. Los elementos se recorren en el orden de inserción de sus claves.

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones Principales

- Clase WeakHashMap<K,V>: Implementación mediante una tabla hash y claves débiles. Los elementos cuyas claves dejan de ser referenciadas se destruyen automáticamente.
- Clase IdentityHashMap<K,V>: Implementación mediante una tabla hash y con comparación de claves y valores mediante == en lugar de equals.
- Clase TreeMap<K,V>: Implementación mediante un árbol, en la que los elementos se recorren según su orden natural (o en el orden que proporciona su Comparator). Es la única clase que implementa la interfaz NavigableMap<K,V>

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones Principales

- Mas referencias:
  - https://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html
    - Base referencial de todas las colecciones principales, wrapped y creación de colecciones propias.
  - https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html
    - Vista general de todas colecciones y sus bases de extensión en la versión de java 8.
  - https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/package-summary.html
    - Vista general de todas colecciones y sus bases de extensión en la versión de java 21.
  - https://docs.oracle.com/en/java/javase/21/core/creating-sequenced-collections-sets-and-maps.html
    - Nuevas colecciones añadidas y explicadas en la versión 21.

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones principales (uso)

Ej.

```
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Map;

public class MultiMap<Clave ,Valor> {

    private Map<Clave, Collection<Valor>> map =
            new HashMap<Clave, Collection<Valor>>();
    …
```

```
public boolean add(Clave clave, Valor valor) {
    boolean add = false;
    Collection<Valor> coleccion = this.get(clave);  if
    (coleccion == null) {
        coleccion = new LinkedList<Valor>();
        coleccion.add(valor);  map.put(clave,
        coleccion);
        add = true;
    } else {
        Valor actual = null;
        Iterator<Valor> iterador = coleccion.iterator();  while
        (iterador.hasNext() && !valor.equals(actual)) {
            actual = iterador.next();
        }
        if (!valor.equals(actual)) {
            coleccion.add(valor);
            map.put(clave, coleccion);
            add = true;
        }
    }
    return add;
}
…
```

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

Colecciones

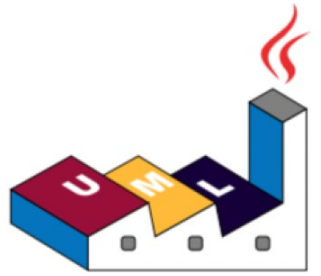
```
public Collection<Valor> get(Clave clave) {
    return map.get(clave);
}

public static void main(String[] args) {
    MultiMap<String, Integer> multimap =
        new MultiMap<String, Integer>();
    multimap.add("Enero", new Integer(1));
    multimap.add("Enero", new Integer(2));
    multimap.add("Enero", new Integer(1));
    multimap.add("Febrero", new Integer(1));
    multimap.add("Febrero", new Integer(1));
    multimap.add("Marzo", new Integer(1));
    multimap.add("Marzo", new Integer(3));
    multimap.add("Marzo", new Integer(2));
    System.out.println("Enero = " + multimap.get("Enero"));
    System.out.println("Febrero = " + multimap.get("Febrero"));
    System.out.println("Marzo = " + multimap.get("Marzo"));
    System.out.println("Abril = " + multimap.get("Abril"));
}
```

```
Enero = [1, 2]
Febrero = [1]
Marzo = [1, 3, 2]
Abril = null
```

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML

- Sirve para pintar las clases y ver su interacción.
- Existen múltiples softwares que ayudan en el pintado.
- Se usan como "*planos*" de software.



2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML

Ref: López Nores, M., García-Duque, J., Blanco-Fernández, Y., García-Duque, J., & García-Duque, J. (2005). *UML: Aplicaciones en Java y C++*. McGraw-Hill.

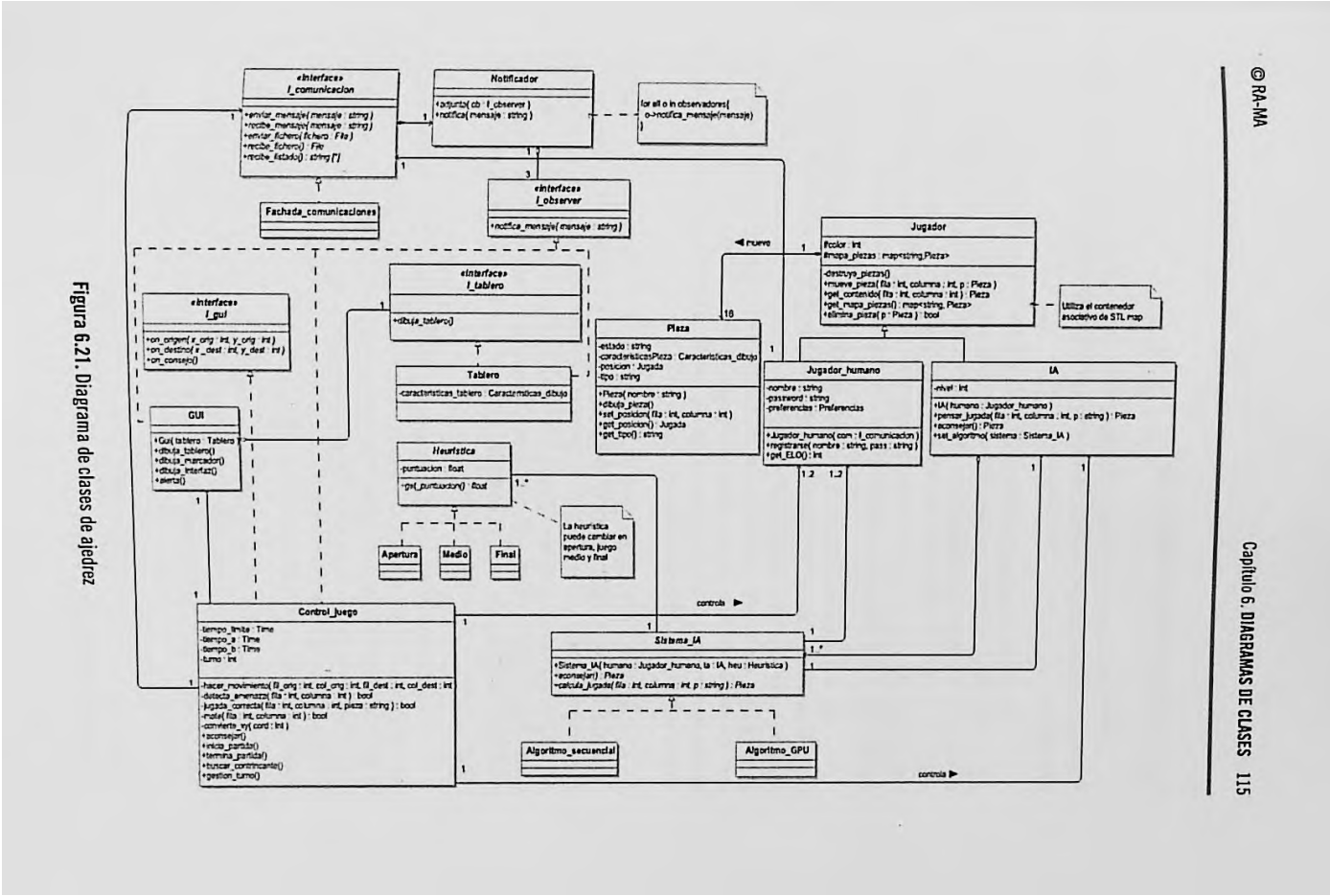


Figura 6.21. Diagrama de clases de ajedrez

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML

UML Aplicaciones en Java y C++

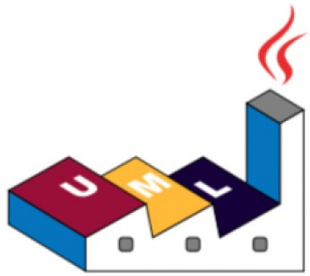
Acceso Biblioteca

OMG Standards Development Organization®

Web Oficial

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML



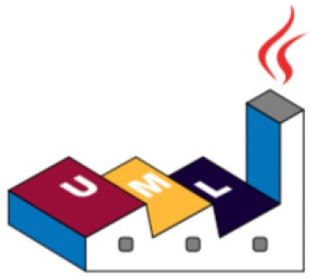
PlantUML

```
@startuml


' ====== CLASE BASE ======
class Persona {
  - String nombre
  # int edad
  + Persona(String nombre, int edad)
  + void mostrarInfo()
}

' ====== SUBCLASE ESTUDIANTE ======
class Estudiante {
  - int matricula
  + Estudiante(String nombre, int edad, int matricula)
  + void estudiar()
}

' ====== SUBCLASE PROFESOR ======
class Profesor {
  - String departamento
  + Profesor(String nombre, int edad, String departamento)
  + void impartirClase()
  {static} + int numeroProfesores
  {static} + void mostrarTotalProfesores()
}
```

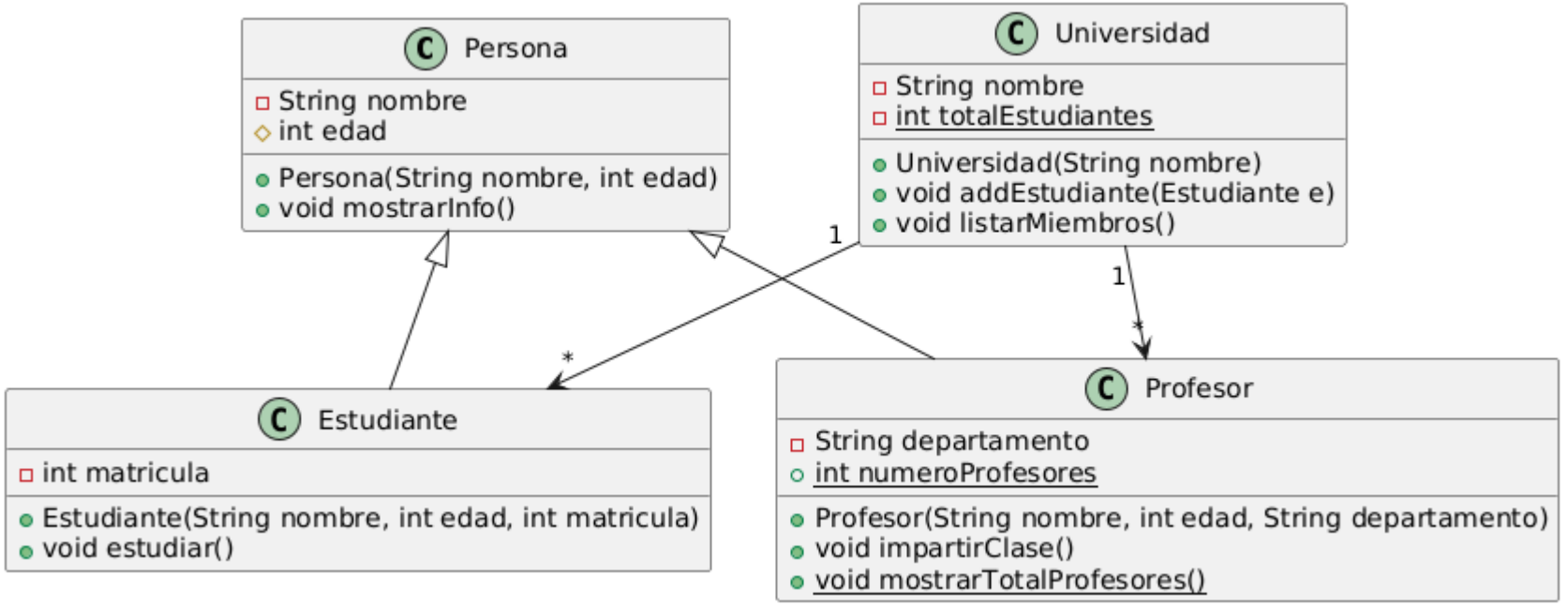
```
' ====== CLASE UNIVERSIDAD ======
class Universidad {
  - String nombre
  {static} - int totalEstudiantes
  + Universidad(String nombre)
  + void addEstudiante(Estudiante e)
  + void listarMiembros()
}

' ====== RELACIONES ======
Persona <|-- Estudiante
Persona <|-- Profesor
Universidad "1" --> "*" Estudiante
Universidad "1" --> "*" Profesor

@enduml
```

2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML

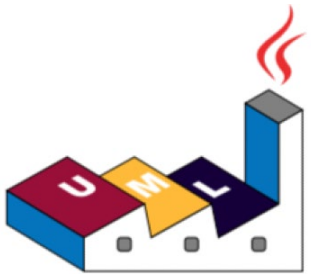


PlantUML



2.5 PROGRAMACIÓN BASADA EN OBJETOS AVANZADA

El lenguaje UML



https://plantuml.com/es/