Composición

Estructuras de Datos

Andrea Rueda

Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas

Composición

- Describe una relación de contenencia (has-a).
- Permite generar tipos de datos complejos a partir de tipos de datos más simples.
- Ejemplos:

  - Un **evento** tiene una **fecha** (día-mes-año) y una **hora** (horas-minutos-segundos).

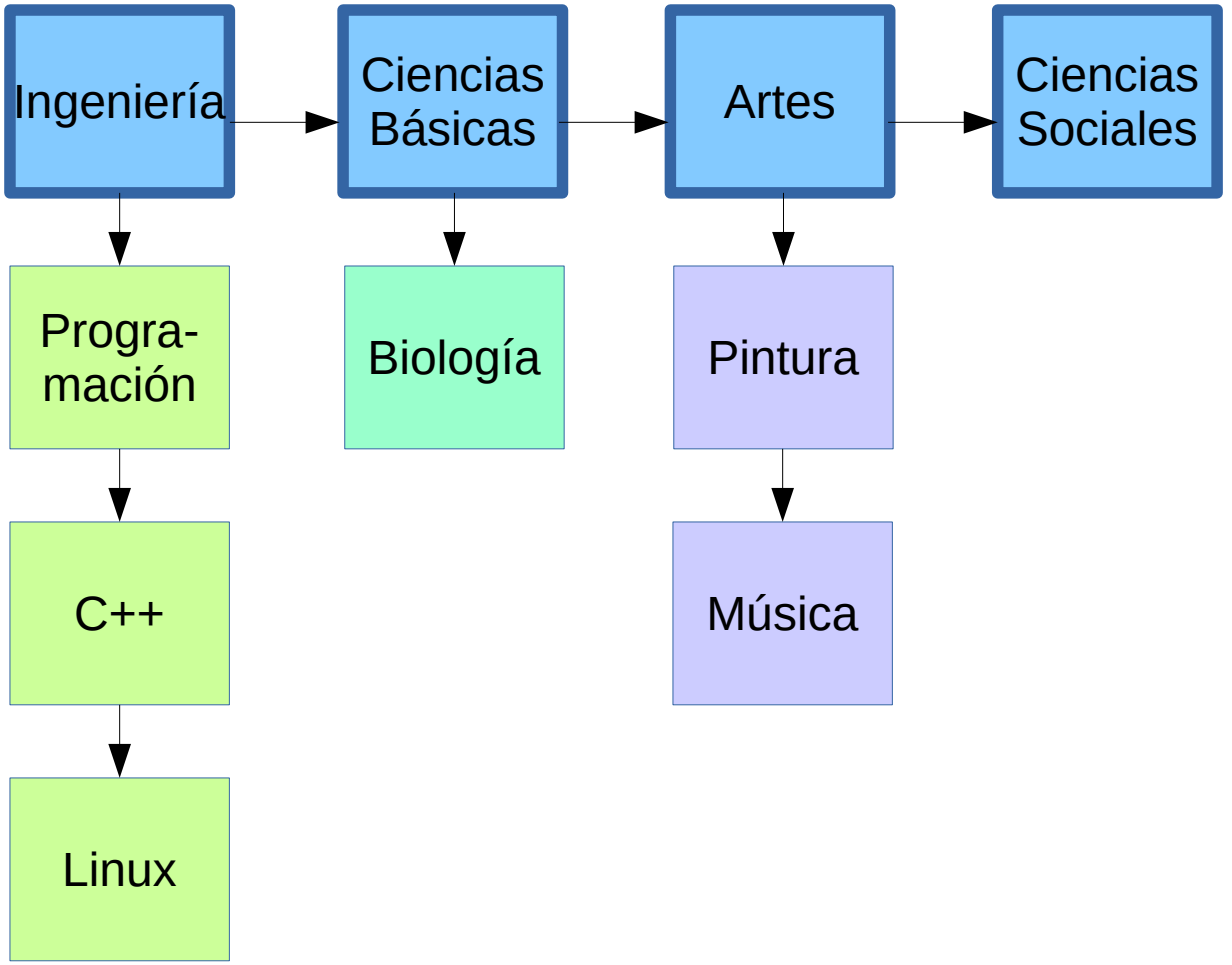
  - Una **universidad** tiene **facultades**, cada **facultad** tiene **carreras**.

¿Por qué composición?

- Cada tipo de dato simple puede enfocarse en una tarea particular.
- Cada tipo de dato simple puede reusarse en diferentes contextos.
- El tipo de dato compuesto puede dejar que cada tipo simple haga la mayor parte del trabajo, y solo se encarga de coordinar el flujo de información.

Composición

- Ejemplo: organización de libros en una biblioteca.



Composición

- Ejemplo: organización de libros en una biblioteca.

Libro
- nombre
- cantidad ejemplares

Composición

- Ejemplo: organización de libros en una biblioteca.

Libro
- nombre
- cantidad ejemplares

Área de Conocimiento
- nombre
- *lista* de Libros

Composición

- Ejemplo: organización de libros en una biblioteca.

Libro
- nombre
- cantidad ejemplares

Área de Conocimiento
- nombre
- *lista* de Libros

Biblioteca
- nombre
- *lista* de Áreas de Conocimiento

Composición

- Organización de libros en una biblioteca.

  TAD **Libro**

  Conjunto mínimo de datos:

  Comportamiento (operaciones) del objeto:

Composición

- Organización de libros en una biblioteca.

  TAD **Libro**

  Conjunto mínimo de datos:

  nombre, cadena de caracteres, título del libro.
  num_ejempl, entero, cantidad de ejemplares.

  Comportamiento (operaciones) del objeto:

  ObtenerNombre(), retornar título del libro.
  ObtenerNumEjempl(), retornar cantidad de ejemplares.
  FijarNombre(nNom), cambiar título a nNom.
  AgregarEjemplar(), incrementar cantidad en 1.
  EliminarEjemplar(), decrementar cantidad en 1.

Composición

- Organización de libros en una biblioteca.

  TAD **AreaConocimiento**

  Conjunto mínimo de datos:

  Comportamiento (operaciones) del objeto:

Composición

- Organización de libros en una biblioteca.

  TAD **AreaConocimiento**

  Conjunto mínimo de datos:

  nombre, cadena de caracteres, nombre del área.
  l_libros, lista de **Libro**, conjunto de libros del área.

  Comportamiento (operaciones) del objeto:

  ObtenerNombre(), retornar nombre del área.
  FijarNombre(nNom), cambiar nombre a nNom.
  ContarEjempl(), contar el total de libros del área.
  AgregarLibro(nLibro), agregar nLibro a la lista.
  EliminarLibro(nLibro), eliminar nLibro de la lista.

Composición

- Organización de libros en una biblioteca.

  TAD **Biblioteca**

  Conjunto mínimo de datos:

  Comportamiento (operaciones) del objeto:

Composición

- Organización de libros en una biblioteca.

  TAD **Biblioteca**

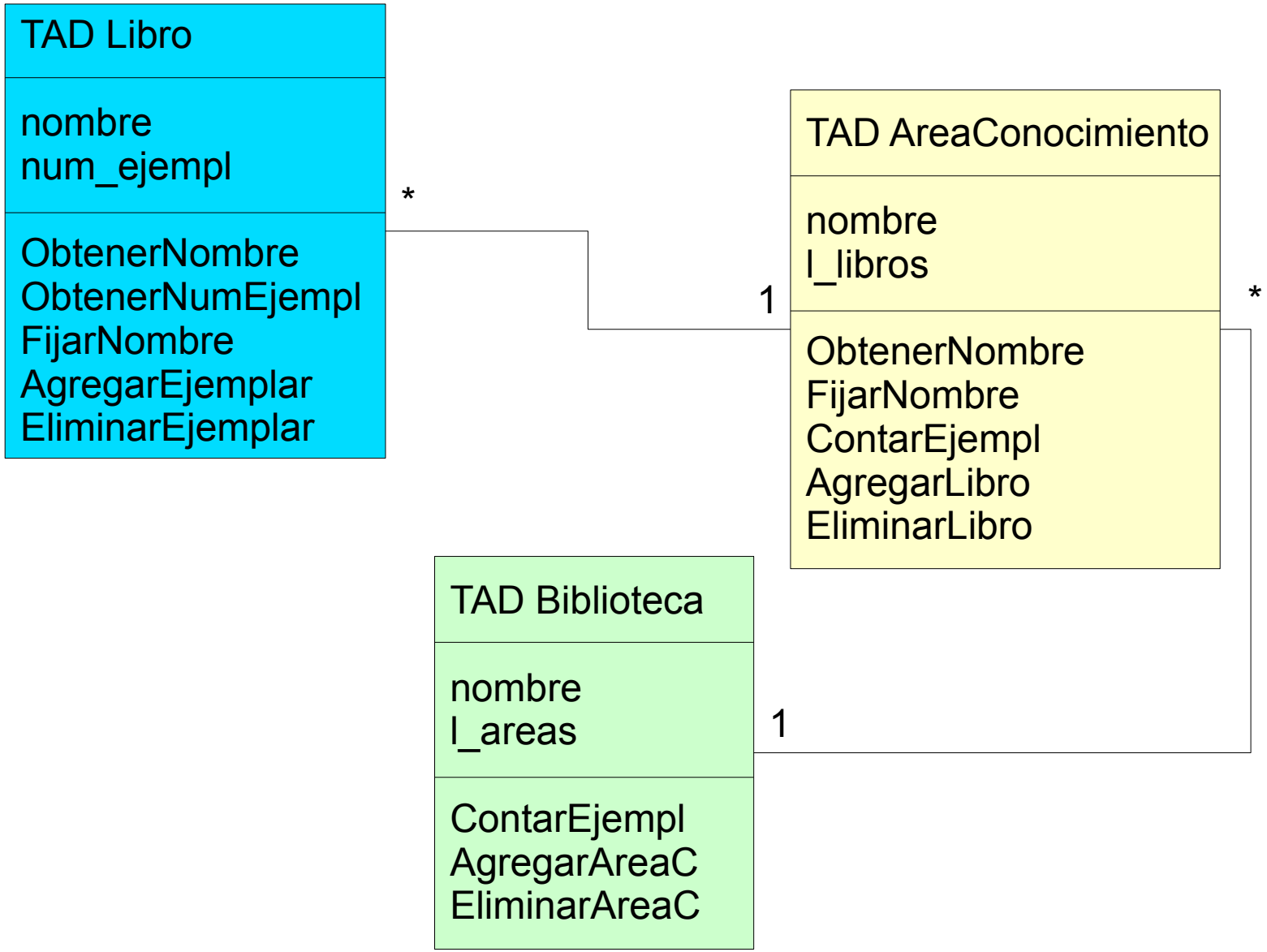
  Conjunto mínimo de datos:

  nombre, cadena de caracteres, nombre de la biblioteca.
  l_areas, lista de **AreaConocimiento**, conjunto de áreas.

  Comportamiento (operaciones) del objeto:

  ContarEjempl(), contar el total de libros de la biblioteca.
  AgregarAreaC(...), agrega un área de conocimiento.
  EliminarAreaC(...), eliminar un área de conocimiento.

Composición



Composición

- Organización de libros en una biblioteca.

```
class Libro {
  public:
   Libro();
   std::string ObtenerNombre();
   unsigned long ObtenerNumEjempl();
   void FijarNombre(std::string n_nombre);
   void AgregarEjemplar();
   void EliminarEjemplar();
  protected:
   std::string nombre;
   unsigned long num_ejempl;
};
```

https://repl.it/@Andrea_del_Pila/EDD-ejercicio-multilistas

Composición

- Organización de libros en una biblioteca.

```
class AreaConocimiento {
  public:
   AreaConocimiento();
   std::string ObtenerNombre();
   void FijarNombre(std::string n_nombre);
   void AgregarLibro(std::string n_libro);
   unsigned long ContarEjempl();
   bool EliminarLibro(std::string n_libro);
  protected:
   std::string nombre;
   std::list<Libro> l_libros;
};
```

https://repl.it/@Andrea_del_Pila/EDD-ejercicio-multilistas

Composición

- Organización de libros en una biblioteca.

```
class Biblioteca {
  public:
    Biblioteca();
    void AgregarAreaC( ... );
    long ContarEjempl() const;
    void EliminarAreaC( ... );
  protected:
    std::string nombre;
    std::list<AreaConocimiento> l_areas;
};
```

https://repl.it/@Andrea_del_Pila/EDD-ejercicio-multilistas

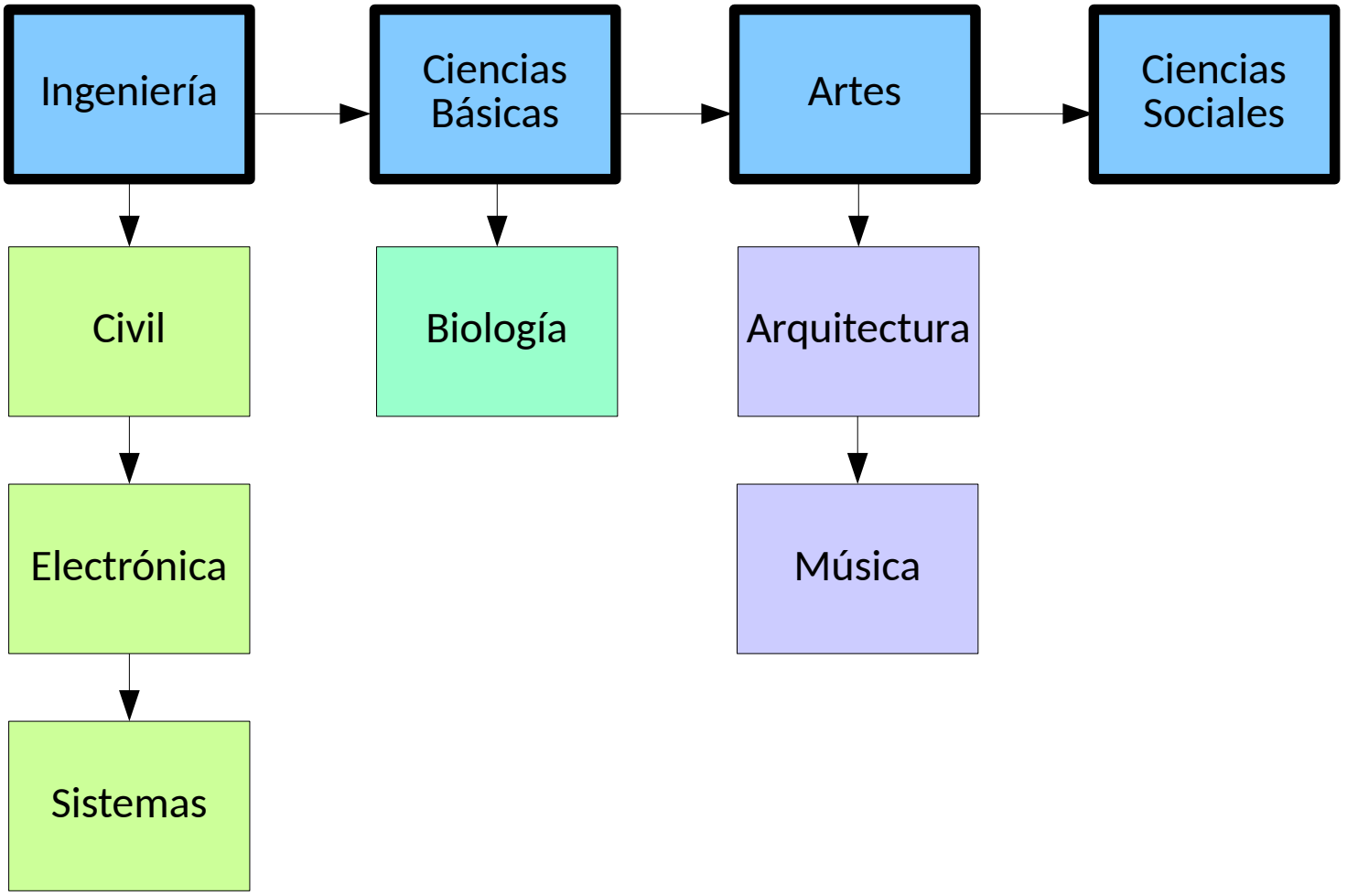
Composición

¡Multilistas!

```
typedef std::list<Libro> TArea;
typedef std::list<Area> TBiblio;

TBiblio lst = biblio.l_areas;
TBiblio::iterator lIt = lst.begin();

for ( ; lIt!=lst.end(); lIt++) {

  TArea slst = lIt->l_libros;
  TArea::iterator slIt = slst.begin( );

  for ( ; slIt!=slst.end(); slIt++)

    std::cout << slIt->nombreL << std::endl;

}
```

Ejercicio

- Carreras por facultades en una universidad:



Ejercicio

- Carreras por facultades en una universidad:

Carrera
- nombre
- cantidad estudiantes

Ejercicio

- Carreras por facultades en una universidad:

Carrera
- nombre
- cantidad estudiantes

Facultad
- nombre
- *lista* de Carrera

Ejercicio

- Carreras por facultades en una universidad:

Carrera
- nombre
- cantidad estudiantes

Facultad
- nombre
- *lista* de Carrera

Universidad
- nombre
- *lista* de Facultad

Ejercicio

- Completar los pasos faltantes:

  - Especificación de cada TAD

  - Diagrama de relación entre TADs

  - Implementación de cada TAD (cabecera y operaciones)

  - Programa principal para ingreso y visualización de datos de la Universidad

Referencias

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. Introduction to Algorithms, 3rd edition. MIT Press, 2009.
- L. Joyanes Aguilar, I. Zahonero. Algoritmos y estructuras de datos: una perspectiva en C. McGraw-Hill, 2004.