Diseño modular

- Técnica de descomposición/construcción de programas, basada en unidades lógicas de datos (módulos) y sus funcionalidades
- Facilita
  - Construir, analizar, probar y depurar por partes
  - Obtener código más legible, reusable, mantenible, etc.
  - Trabajar en equipo
- Necesario para programas grandes

Diseño modular

- Caso particular: programación orientada a objetos
  - módulo = clase
  - variable = objeto
  - operación = método
  - herencia y otras técnicas avanzadas de diseño

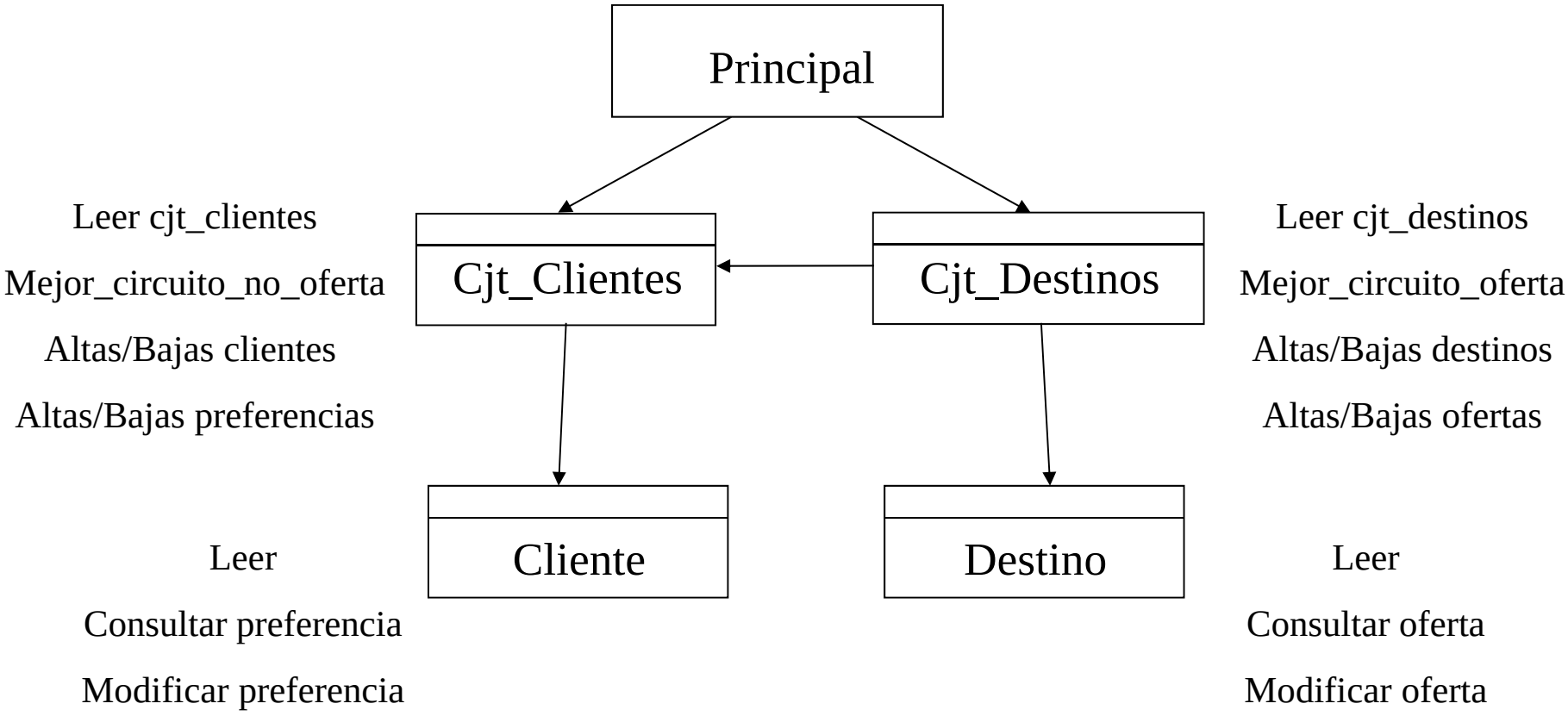
- Ejemplos de clases standard de c++:
  - `std::iostream` (lectura y escritura)
  - `std::vector` (contenedor indexado)

- Notación `objeto.método`:
  - `v.size()` no `size(v)`

Ejemplo: Agencia de viajes

- La agencia maneja un conjunto de clientes y un conjunto de destinos turísticos
- Cada cliente le informa a la agencia sobre sus preferencias
- Cada destino puede estar de oferta o no
- La agencia proporciona a sus clientes dos tipos de circuitos turísticos (viajes en que se visitan uno o más destinos) que respeten las preferencias:
  - Circuitos basados en las ofertas
  - Circuitos independientes de las ofertas

Diagrama modular: agencia de viajes



A $\rightarrow$ B = A usa B

Ejemplo: compras en una tienda

- Extraído de la asignatura IES
- Clases: tienda, caja, venta, producto, pago, etc.
- Diagrama clases: ver clases – IES.pdf

Ejemplo: Portal web de tests

- Extraído de un PFC
- Clases: usuario, administrador, test, sección de test, pregunta, tipo de preguntas, grupo preguntas, respuesta, grupo respuestas, etc
- Diagrama clases: ver clases – PFC.pdf

Clasificación de módulos

- Módulos de **datos**: tipo + operaciones
- Módulos **funcionales**: sólo operaciones (apenas se usarán en este curso)

Roles en el diseño modular

- **Especificación**: qué contiene el módulo y qué se puede hacer con ello
- **Uso**: integración del módulo en un diseño completo
- **Implementación**: cómo se representa el tipo y cómo se codifican las operaciones en un lenguaje de programación

Claves del diseño modular

- **Especificación:**
  - A qué clase pertenece cada funcionalidad
  - Relaciones entre clases
  - Parametrización; especificación pre/post
- **Implementación:**
  - Buena elección de componentes
  - Eficiencia en tiempo y espacio

Claves del diseño modular

• **Independencia de la especificación respecto de la implementación**:

- Abstracción de datos
- Componentes públicos y privados
- La clase se ha de poder usar conociendo solo los componentes públicos
- Los componentes privados se han de poder cambiar sin que afecte a la especificación