

Programación Avanzada

Implementación
Colecciones

[Contenido]

- n Introducción
- n Colecciones de Objetos
- n Colecciones Genéricas
- n Colecciones Concretas
- n Enfoque Completo
- n Realización de una Colección Genérica
- n Iteradores
- n Diccionarios
- n Búsquedas

[Introducción]

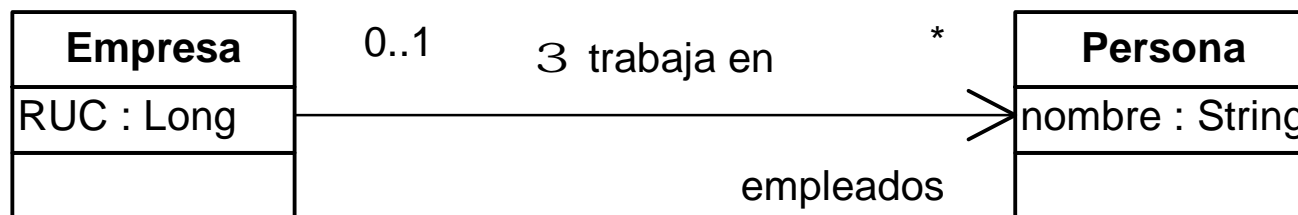
- n La implementación de asociaciones usualmente requiere del uso de colecciones para permitir links con muchos objetos
- n El tipo de los elementos de las colecciones depende de la clase correspondiente al extremo de asociación navegable
- n Por tratarlas de manera uniforme éstas comparten una misma estructura que puede ser reutilizada para generarlas

[Introducción (2)]

- n Se distinguen dos tipos de colecciones dependiendo de si los elementos contenidos poseen una clave que los identifique o no
- n La definición de las colecciones a utilizar en la implementación se estudiará incrementalmente
- n Se comenzará definiendo una **colección genérica** de elementos sin clave la cual será aumentada para
 - i Permitir iteraciones sobre sus elementos
 - i Soportar el uso de claves
 - i Soportar diferentes tipos de búsquedas
- n Dichas definiciones serán luego utilizadas para implementar **colecciones concretas**

Colecciones de Objetos

- Las colecciones de objetos son una herramienta fundamental para la implementación de muchas de las asociaciones presentes en un diseño



El pseudoatributo `empleados` de la clase `Empresa` introduce la necesidad de una colección de personas

[Colecciones de Objetos (2)]

- n Las colecciones deben permitir:
 - i Realizar iteraciones sobre sus elementos
 - i Realizar búsquedas de elementos por clave (en caso de que los elementos tengan una)
 - i Realizar búsquedas diversas
- n Las colecciones concretas difieren en el tipo de elementos que contendrán pero coinciden en el tipo de servicios que brindan

[Colecciones de Objetos (3)]

- n Desarrollar cada colección en forma íntegra cada vez que se necesita resulta poco práctico
- n Es posible definir una única vez una infraestructura común que sirva de base para todas las colecciones específicas:
 - i **Colecciones paramétricas** (templates): el tipo del elemento a almacenar es declarado como parámetro que será instanciado al generar la colección particular
 - i **Colecciones genéricas**: pueden almacenar directamente cualquier tipo de elemento

[Colecciones Genéricas]

- n Una colección genérica está definida de forma tal que pueda contener a cualquier tipo de elemento
- n Aspectos a considerar:
 - i ¿Cómo lograr que un elemento de una clase cualquiera pueda ser almacenado en la colección genérica?
 - i ¿Cómo se define la colección genérica?

Colecciones Genéricas

Genericidad de la Colección

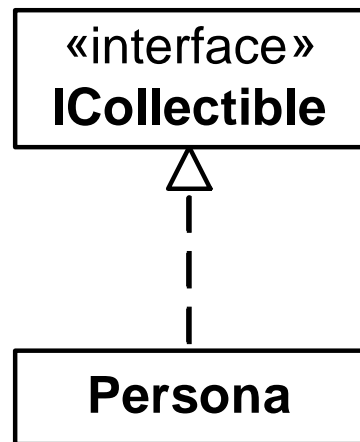
- n ¿Cómo lograr que un elemento de una clase cualquiera pueda ser almacenado en la colección genérica?
- n Se define la interfaz de marca `ICollection`
- n Cuando se desea que los elementos de una cierta clase puedan ser almacenados en una colección genérica se solicita que dicha clase realice la interfaz `ICollection`
- n De esta forma la colección genérica contendrá elementos “coleccionables” (es decir, que implementarán la interfaz `ICollection`)

Colecciones Genéricas

Genericidad de la Colección (2)

n Ejemplo:

- i La clase Persona debe realizar la interfaz de marca `ICollection` para poder agregar personas a una colección genérica



Recordar que una interfaz de marca no posee ninguna operación, por lo que no obliga a las clases que la implementan a presentar ningún servicio.

Colecciones Genéricas

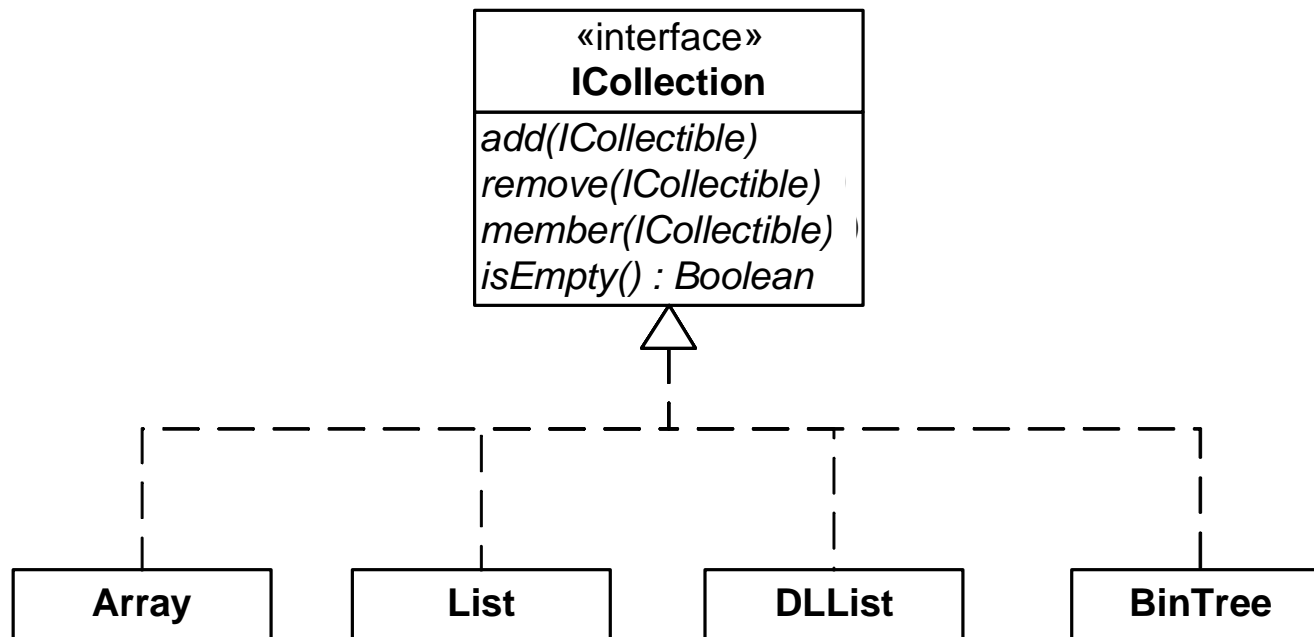
Encapsulamiento

- n ¿Cómo se define una colección genérica?
- n La noción de colección es independiente de su implementación
- n Se separa la especificación de la implementación:
 - i Se define una interfaz `ICollection`
 - i Una cierta colección genérica será una implementación que realice esta interfaz

Colecciones Genéricas

Realizaciones

- Podrán existir diferentes realizaciones (cada una con su estructura de datos particular) de la colección genérica

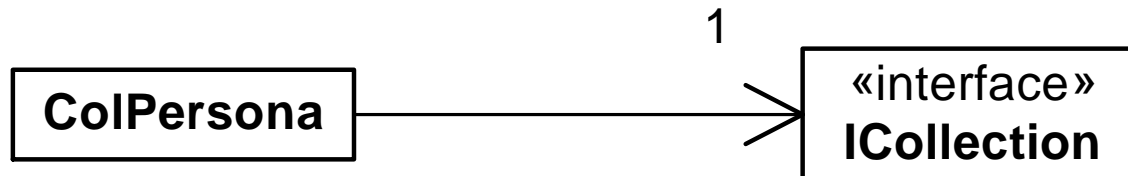


Colecciones Concretas

- n ¿Cómo se define una colección concreta a partir de una colección genérica?
- n Una colección concreta es aquella que:
 - i Presta los mismos servicios que la genérica
 - i Puede definir nuevos servicios
 - i Fija el tipo de los objetos a coleccionar
- n El pseudoatributo `empl eados` de la clase `Empresa` puede ser de tipo `Col Persona`
- n Una instancia de clase `Col Persona`:
 - i Encapsula a una colección genérica (que es quien contendrá efectivamente a las personas)
 - i Puede definir operaciones que actúen sobre las personas
 - i Asegura que a dicha colección genérica le sea agregado elementos de clase `Persona` únicamente

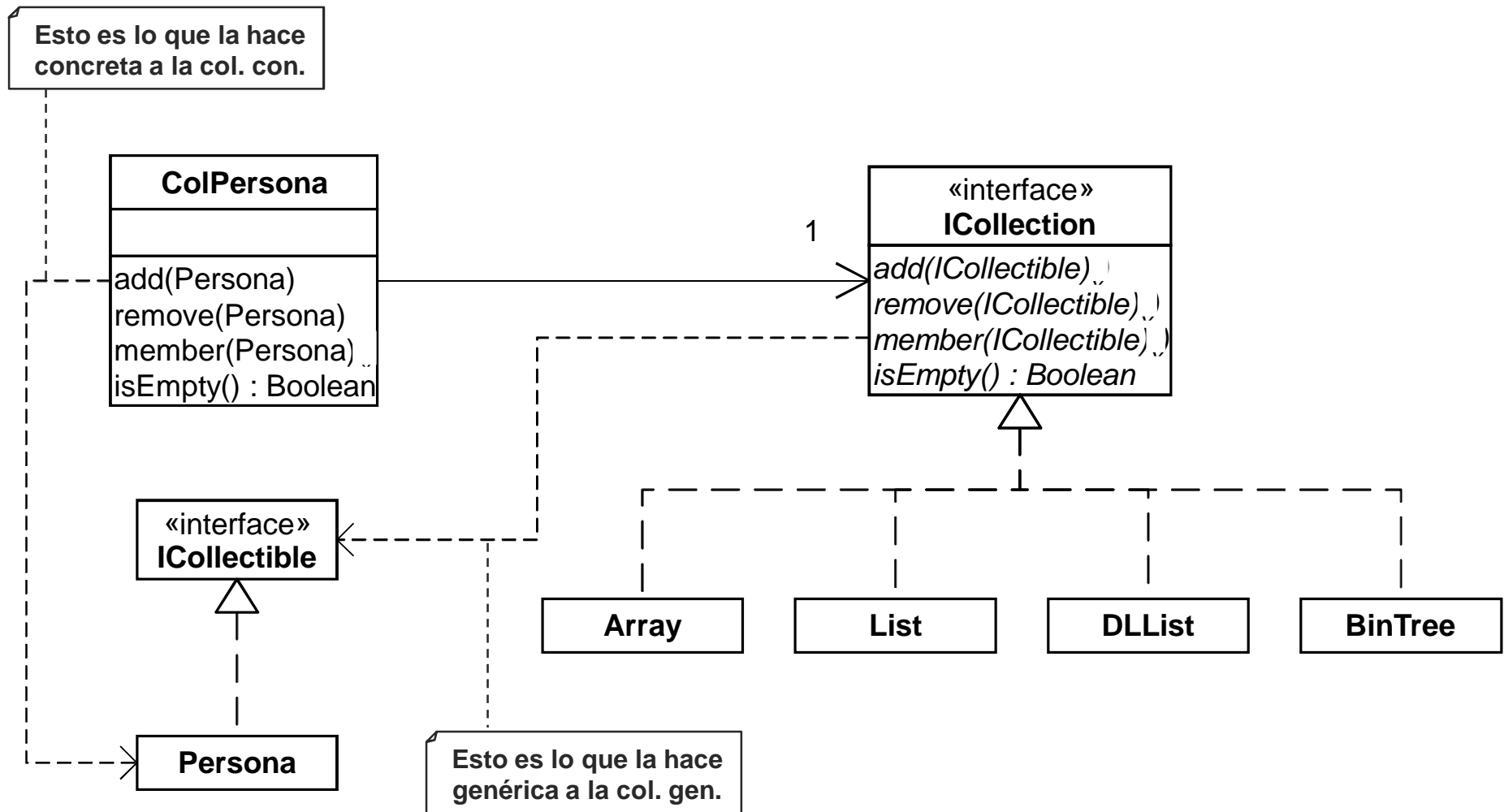
[Colecciones Concretas (2)]

- n Una colección concreta se define como un “wrapper” de una colección genérica
- n Ejemplo:



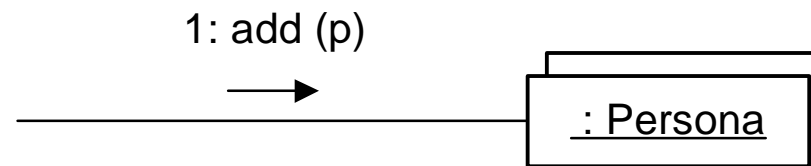
Cada instancia de **Col Persona** utiliza una realización concreta de **ICollection** para almacenar (únicamente) instancias de **Persona**

Enfoque Completo

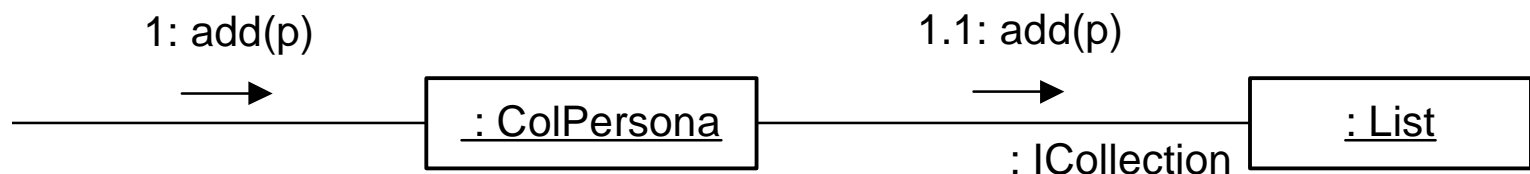


[Enfoque Completo (2)]

- n Interacciones de una colección concreta:
 - i Por ejemplo, lo diseñado de esta manera

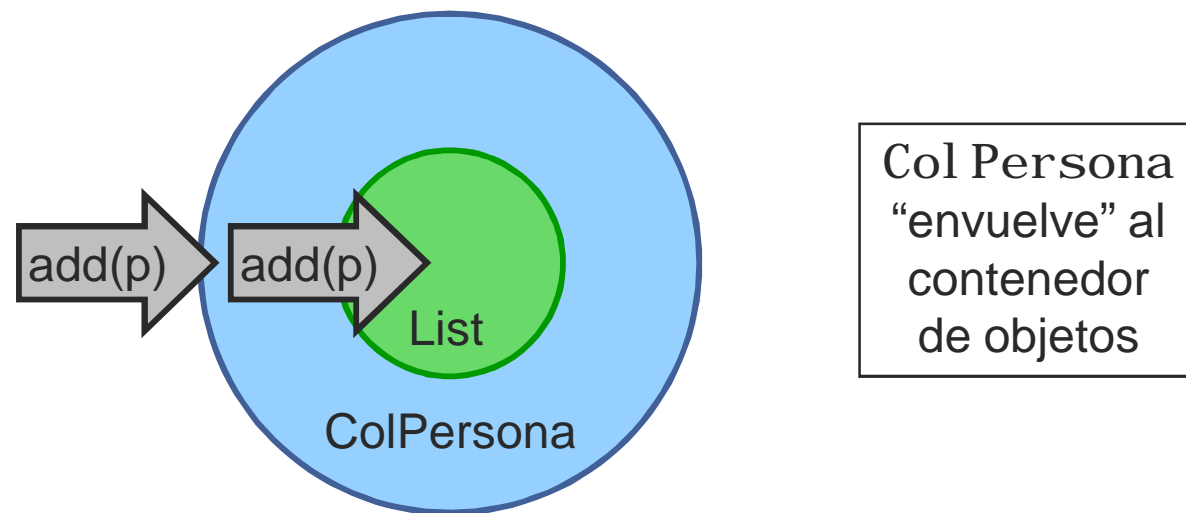


- i Podría en la práctica hacerse de esta otra



[Enfoque Completo (3)]

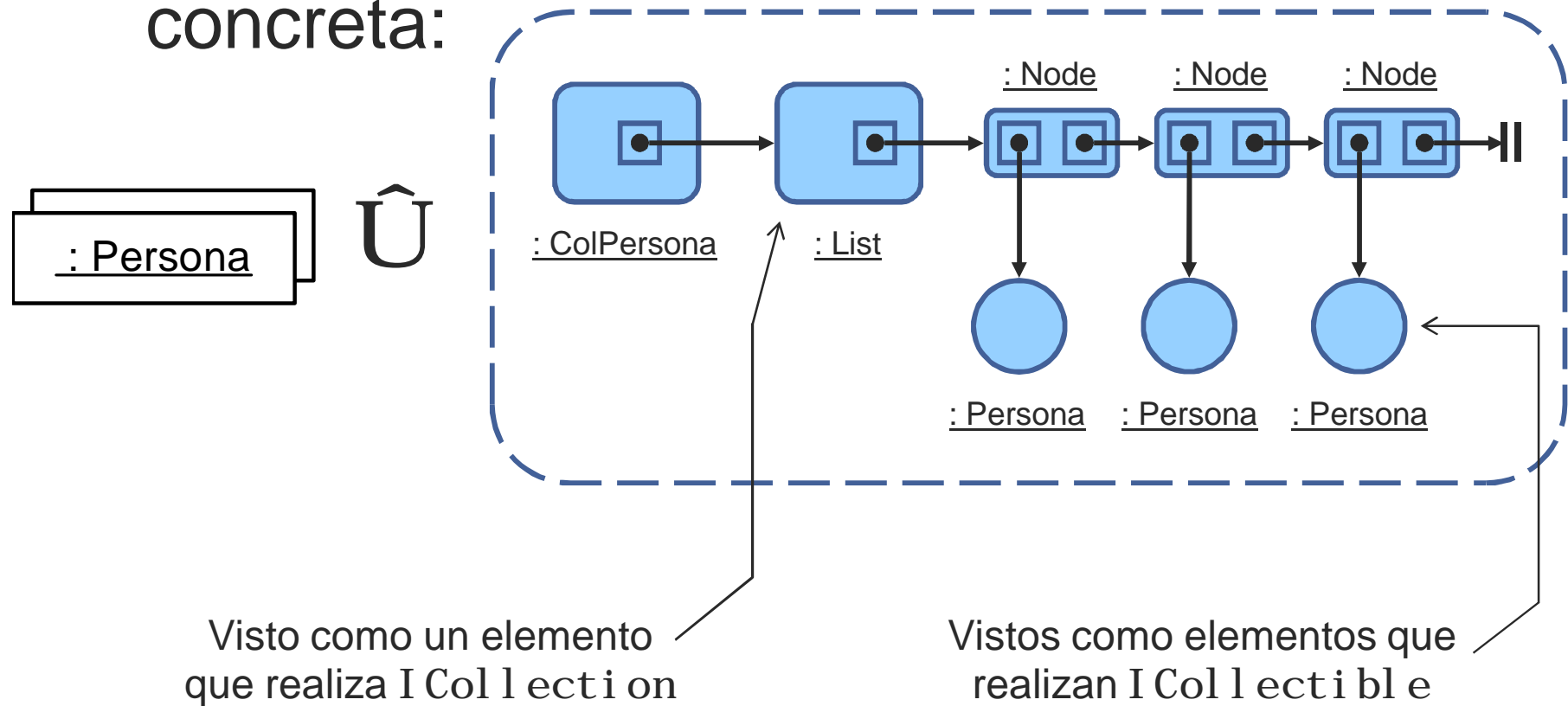
- n Interacciones de una colección concreta:
 - i Visto de otra manera



Una empresa que tenga una instancia de Col Persona nunca utilizará el contenedor de elementos (en este caso Li st) directamente.

[Enfoque Completo (4)]

n Estructura de una colección concreta:



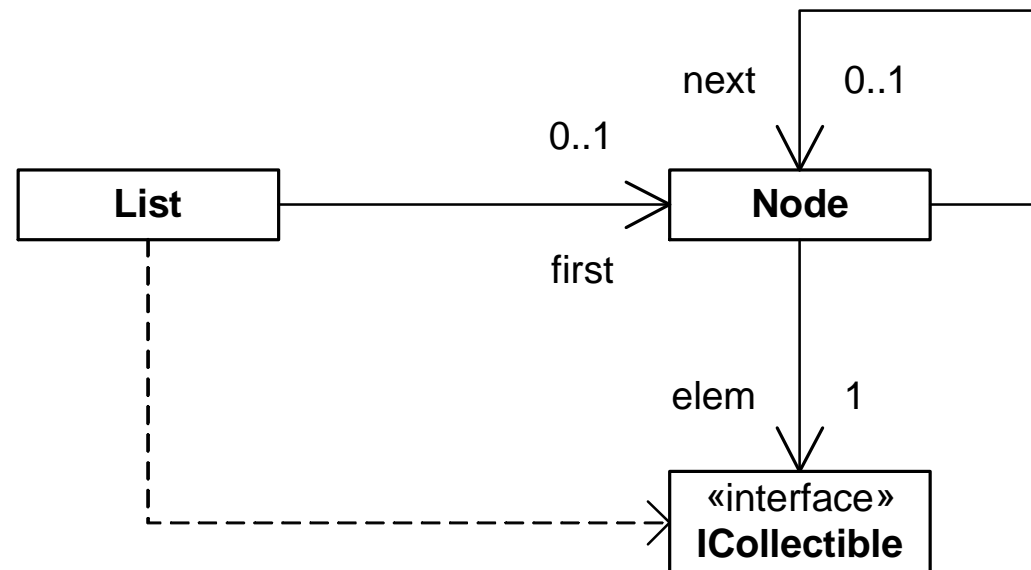
[Realización de una Col. Genérica]

- n La interfaz `ICollection` declara qué servicios debe proveer una colección
- n Es posible realizar dicha interfaz de diferentes maneras mediante diferentes estructuras de datos
- n Realizaciones posibles:
 - i Array o Vector
 - i Lista común o doblemente enlazada
 - i Arbol binario
 - i Etc.

Realización de una Colección Genérica

Lista Común

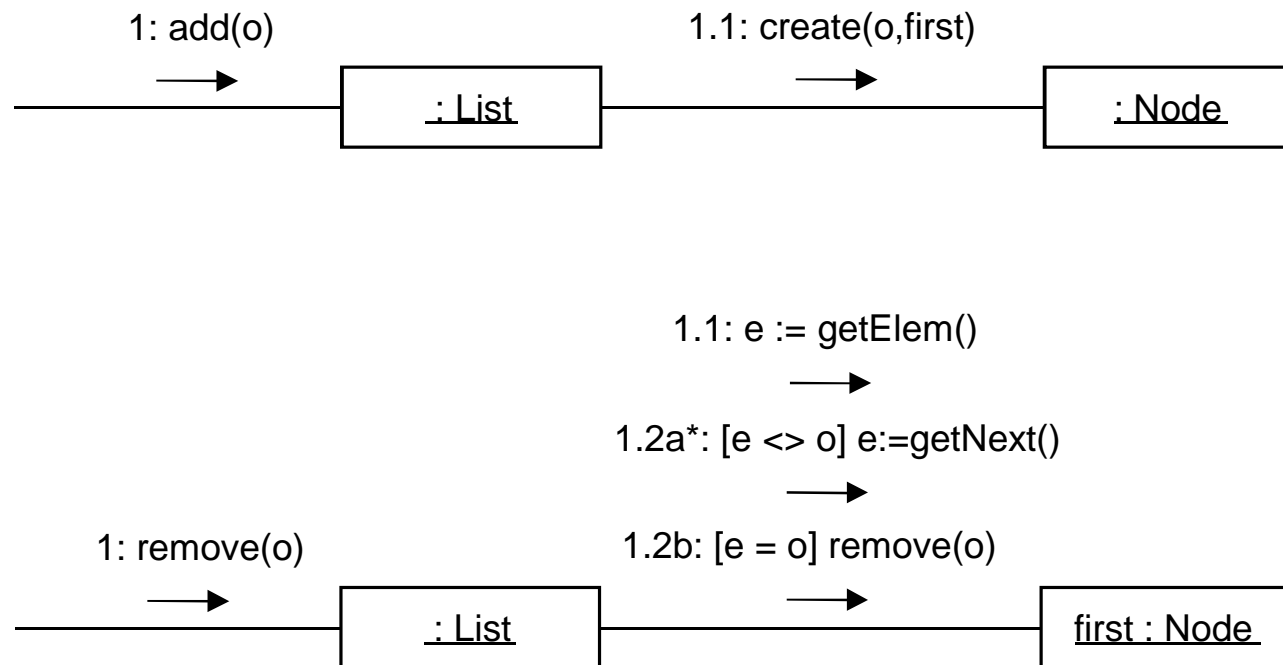
- El diseño de una lista común utilizando clases no difiere significativamente del diseño usual



Realización de una Colección Genérica

Lista Común (2)

- Las operaciones no se resuelven en forma completa en la clase `List`



[Iteradores]

- n Es muy común necesitar realizar iteraciones sobre los elementos de una colección
- n La interfaz `ICollection` es aumentada con la siguiente operación:

```
getIterator() : Iterator
```

- n A su vez la colección concreta `ColPersona` es aumentada con la operación:

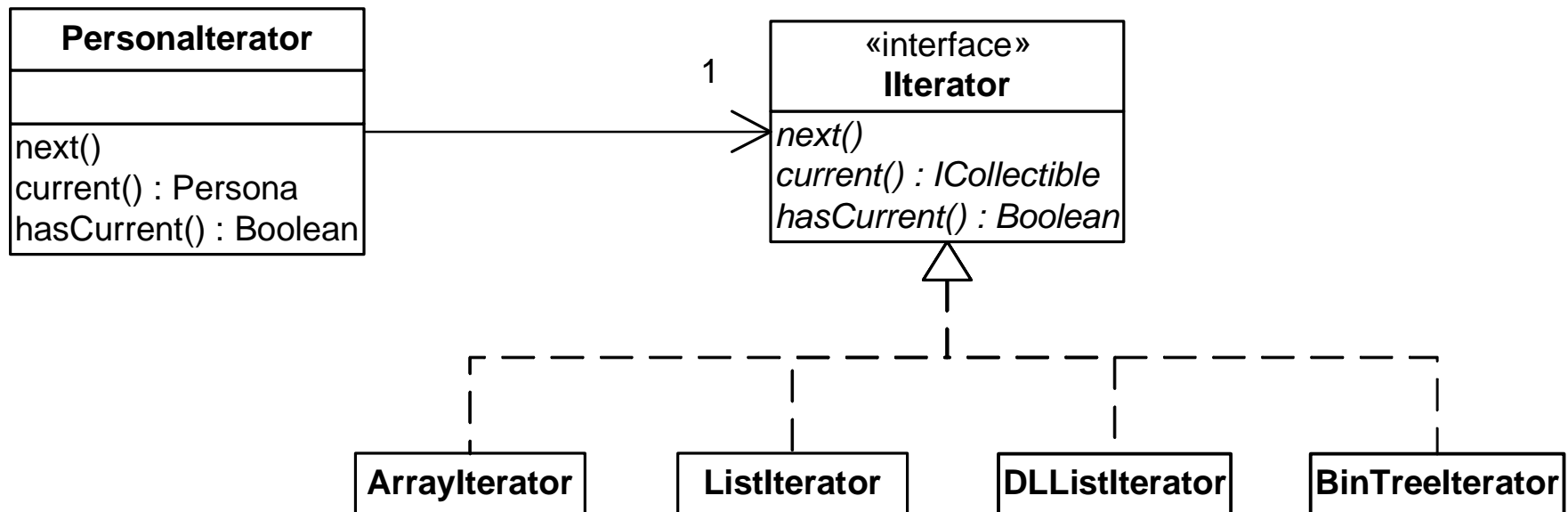
```
getIterator() : PersonalIterator
```

[Iteradores (2)]

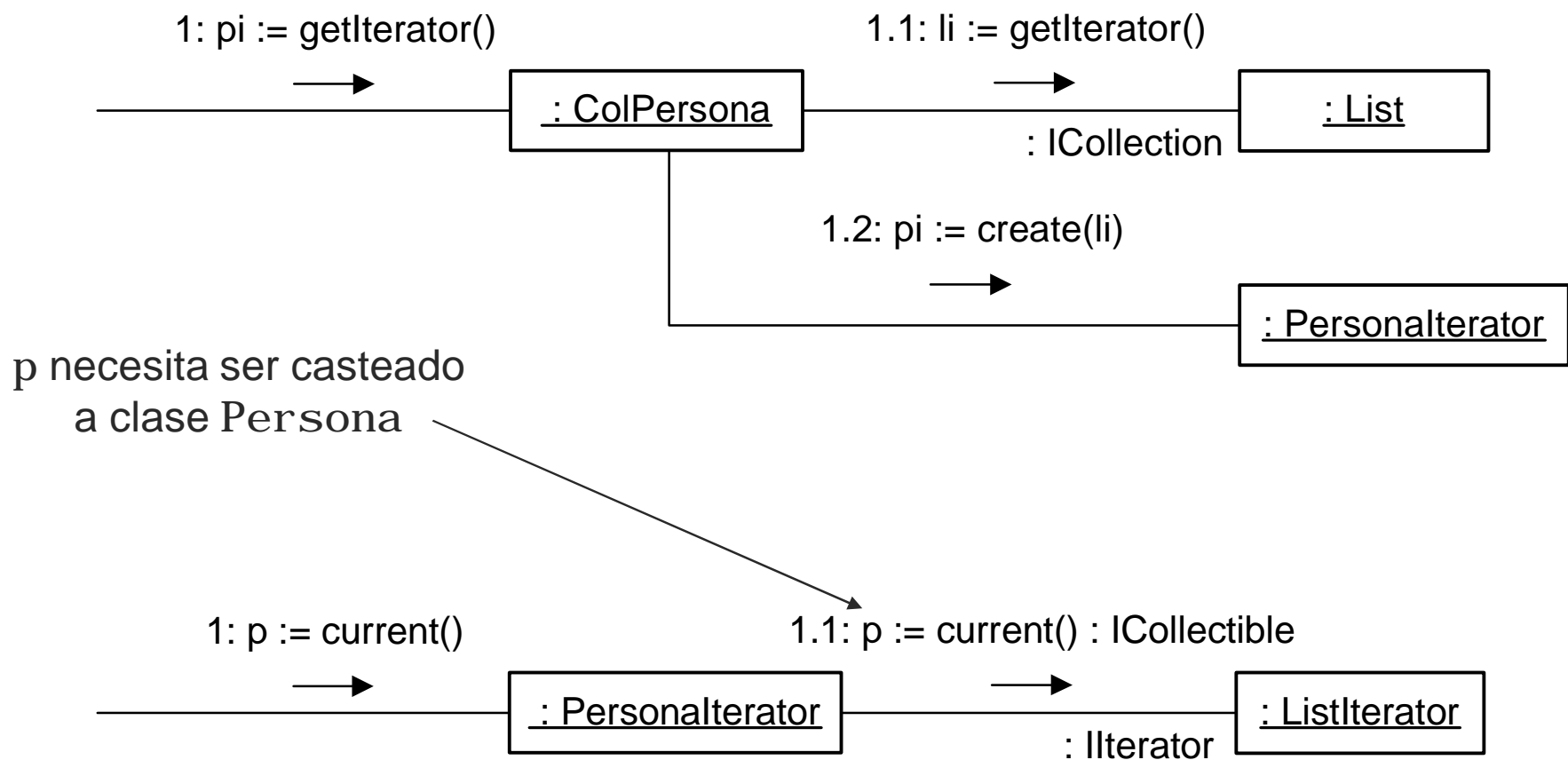
- n Un iterador (sugerido en el patrón de diseño Iterator) es un objeto que permite recorrer uno a uno los elementos de una colección
- n Un iterador es un observador externo de la colección (no en el sentido del patrón Observer)
- n Una colección puede tener diferentes iteradores realizando diferentes iteraciones simultáneamente

Iteradores Estructura

- Un iterador concreto como `PersonalIterator` encapsula a una realización de `IIterator`



Iteradores Interacciones



Iteradores

Uso de Iteradores

```
class Venta {  
    private:  
        ColLineasDeVenta lineas;  
    public:  
        float totalVenta();  
};  
  
float Venta::totalVenta() {  
    float total = 0;  
    LineaDeVentaIterator it = lineas.getIterator();  
  
    while (it.hasCurrent()) {  
        total = total + it.current()->subtotal();  
        it.next();  
    }  
    return total;  
}
```

Iteradores

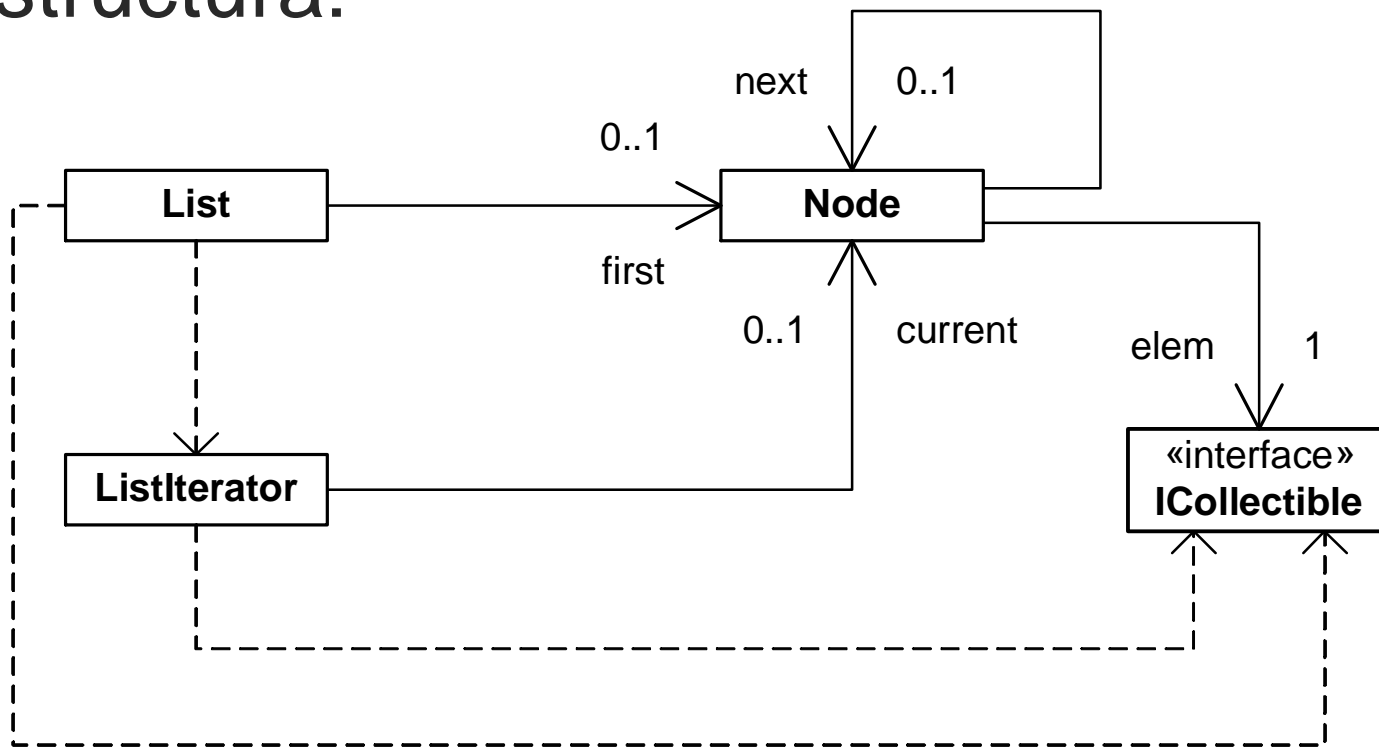
Realización de Iteradores

- n Como ejemplo de realización de iteradores se presenta el diseño de un iterador sobre listas comunes
- n La clase `ListIterator`:
 - i Realiza la interfaz `Iterator`
 - i Es encapsulado por un iterador concreto (como `PersonalIterator`)

Iteradores

Realización de Iteradores (2)

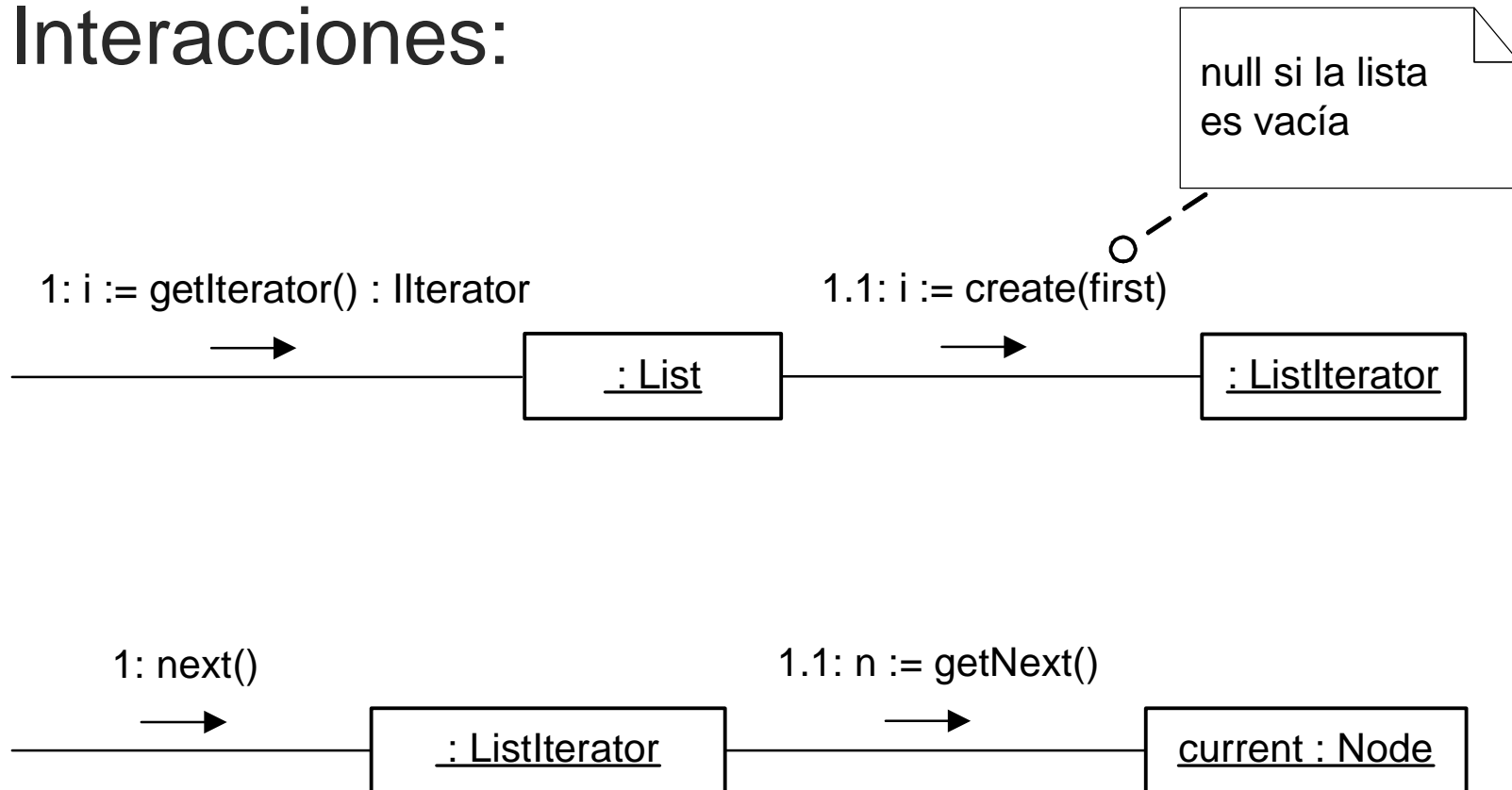
n Estructura:



Iteradores

Realización de Iteradores (3)

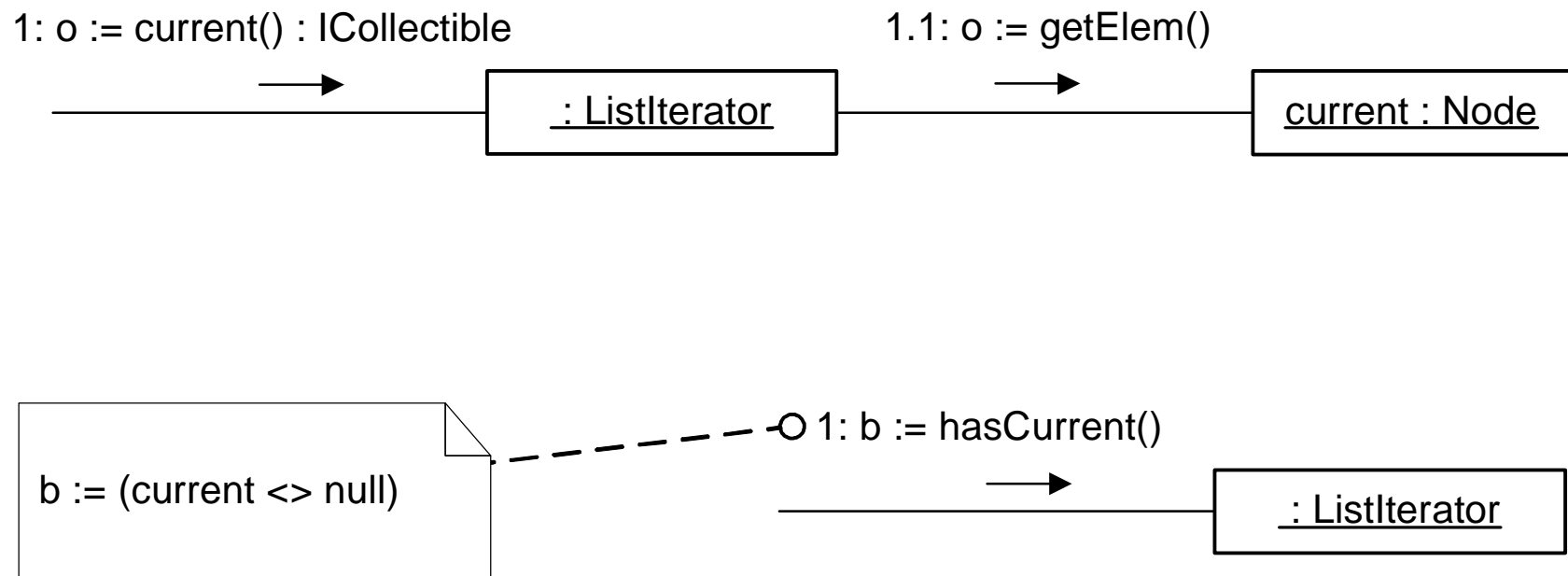
n Interacciones:



Iteradores

Realización de Iteradores (4)

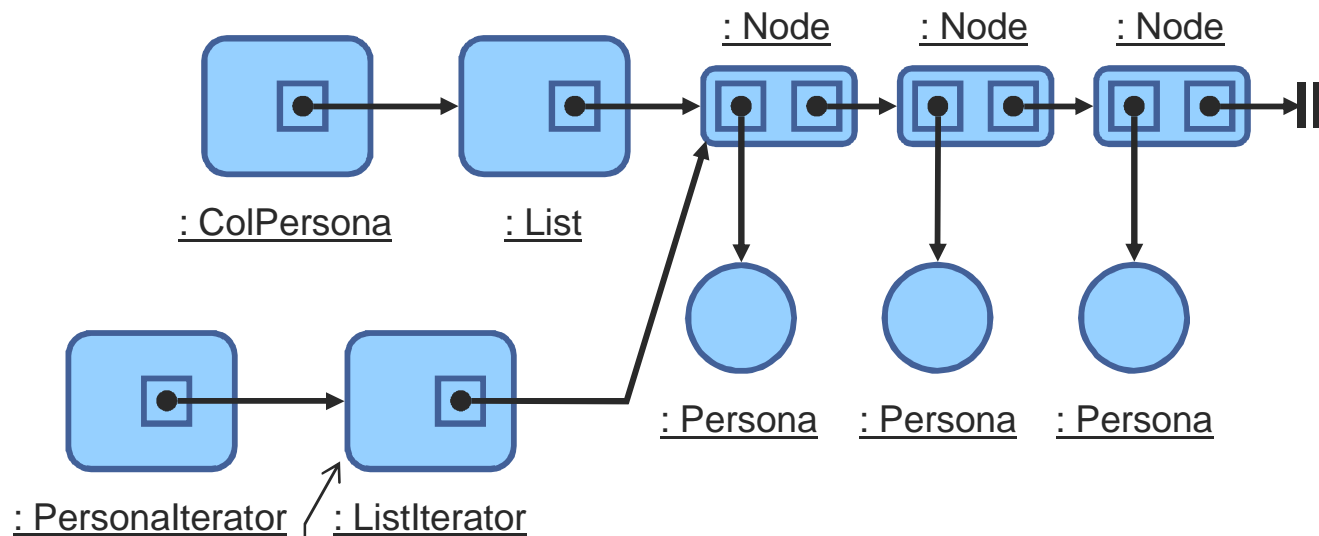
n Interacciones (cont.):



Iteradores

Realización de Iteradores (5)

n Ejemplo:



Visto como un elemento
que realiza **Iterator**

[Diccionarios]

- n Un diccionario es un tipo particular de colección en el cual se almacenan objetos que pueden ser identificados por una clave
- n Se define la interfaz `IDictionary` y se utiliza en forma análoga a la interfaz `ICollection`:
 - i Existirán diferentes realizaciones de `IDictionary`
 - i Las mismas contendrán elementos que realicen la interfaz `ICollection`
 - i Un diccionario concreto como `Dictionary` encapsulará una realización de `IDictionary`

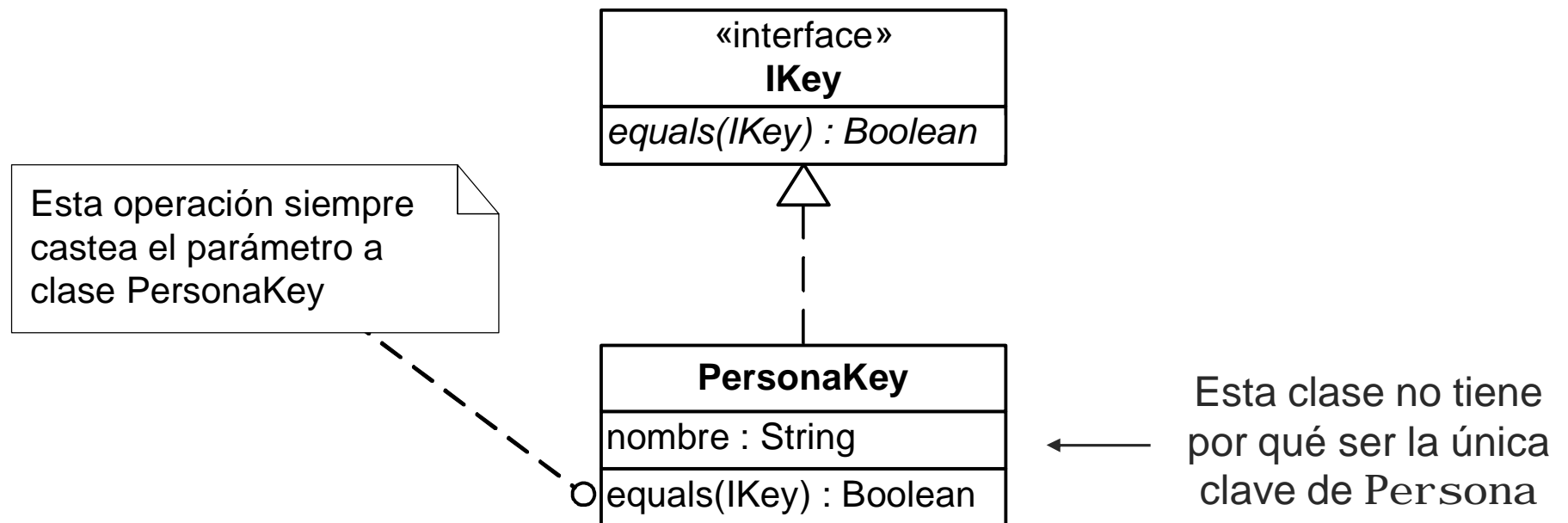
Diccionarios

Uso de Claves

- n Se está tratando la noción de diccionario genérico por lo que la clave que identifica a los elementos debe ser también genérica
- n Se define la interfaz IKey:
 - i Debe ser realizada por una clase que representa la clave de los elementos a incluir en el diccionario
 - i Contiene únicamente la operación `equals(IKey) : bool` ean utilizada para comparar claves concretas

Diccionarios

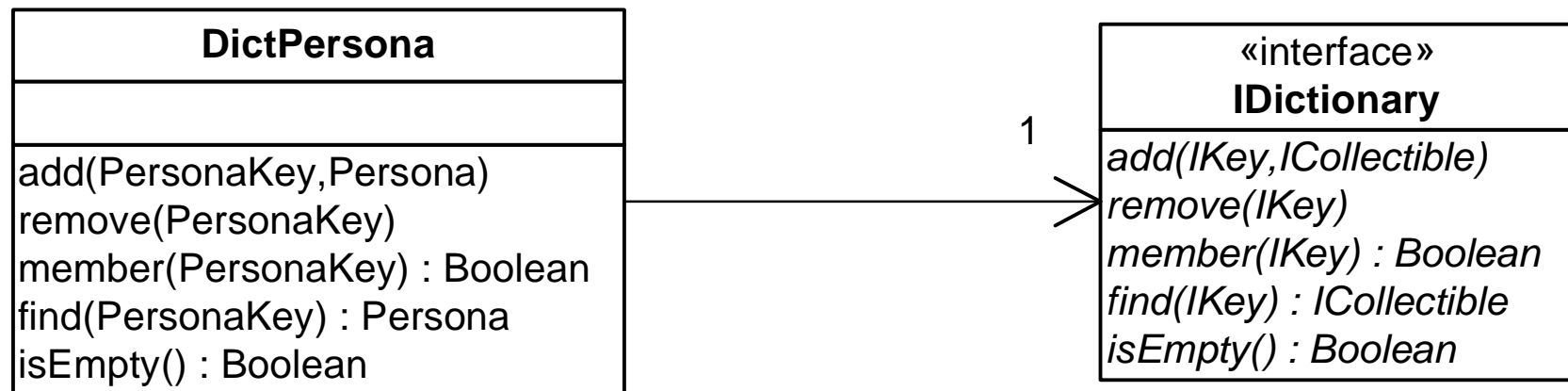
Uso de Claves (2)



Los atributos de la clave concreta son una combinación de atributos de la clase

Diccionarios

Diccionarios Concretos

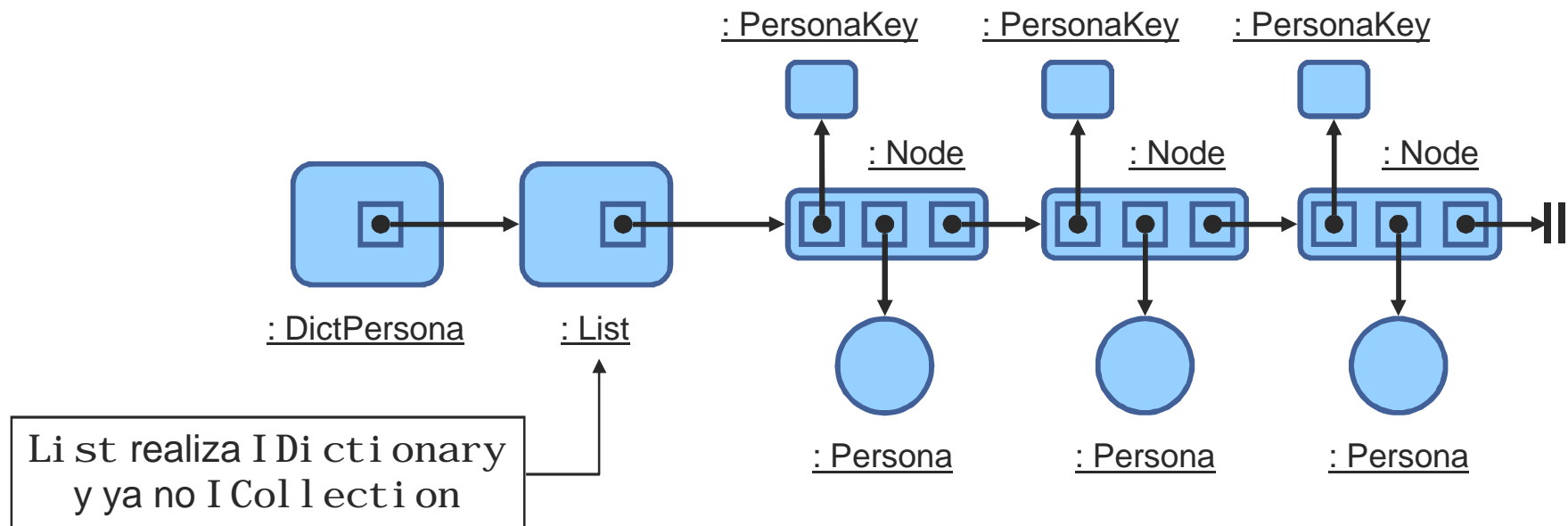


La clase **DictPersona** encapsula a una realización de **IDictionary** en forma análoga a como lo hace la clase **ColPersona** con la interfaz **ICollection**

Diccionarios

Diccionarios Concretos (2)

n Ejemplo:



La relación entre una persona y su clave es particular a cada diccionario concreto. La clave que le corresponde a una persona es la referenciada por el nodo que referencia a la persona (determinada al momento del `add()`).

Diccionarios

Iteraciones en Diccionarios

- n Un diccionario es una colección por lo que tiene sentido necesitar iterar sobre sus elementos
- n Se incorpora a la interfaz `IDictionary`:
 - i `GetEnumerator() : IEnumerator` que devuelve un iterador sobre los elementos contenidos en el diccionario
 - i `GetKeyIterator() : IEnumerator` que devuelve un iterador sobre las claves de los elementos contenidos en el diccionario

Diccionarios

Iteraciones en Diccionarios (2)

- n En diccionarios concretos (p.e. en el caso de `DictPersona`) las operaciones para realizar iteraciones son:
 - i `getPersonalIterator()` : `PersonalIterator` que devuelve un iterador sobre las personas del diccionario
 - i `getKeyIterator()` : `PersonaKeyIterator` que devuelve un iterador sobre las claves (`PersonaKey`) de las personas contenidas en el diccionario

Búsquedas

- n Las búsquedas por clave no son el único tipo de búsqueda que se suele requerir
- n Existe otro tipo de búsquedas que no involucran necesariamente una clave:
 - i Buscar todos los empleados menores de una cierta edad
 - i Buscar todos los empleados contratados antes de una fecha dada
- n Este tipo de funcionalidad es análogo al que proporciona la operación `select ()` de OCL

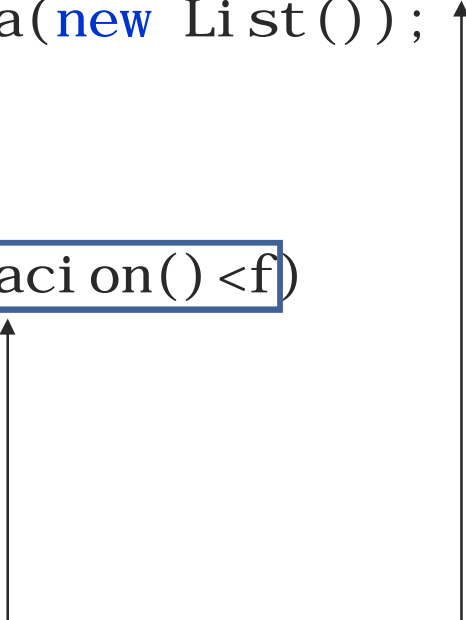
[Búsquedas (2)]

- n Dado que este tipo de búsquedas dependen de cada colección se implementan en las colecciones concretas
- n De esta forma se define una operación por cada búsqueda necesaria:
 - i Por ejemplo para buscar los empleados contratados antes de una fecha dada se incluye en Col Persona:

```
selectContratadosAntes(Fecha) : Col Persona
```


[Búsquedas (3)]

```
Col Persona * Col Persona: : selectContratadosAntes(Fecha f) {  
    Col Persona * result = new Col Persona(new List());  
    PersonaIterator it = getIterator();  
  
    while(it.hasCurrent()) {  
        if(it.current()->getFechaContratacion() < f)  
            result->add(it.current());  
        it.next();  
    }  
    return result;  
}
```



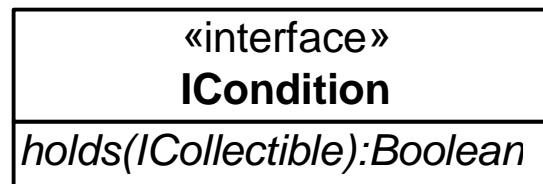
Notar que todas las variantes de select () de Col Persona serán exactamente iguales entre sí a menos de esta porción del código

[Búsquedas (4)]

- n Las operaciones de búsqueda de una colección concreta son muy similares entre sí
- n Incluso no solamente las correspondientes a una misma colección concreta: las búsquedas en todas las colecciones son similares salvo:
 - i La condición que determina la inclusión de un elemento en el resultado
 - i Los parámetros
 - i El tipo del iterador y la colección resultado

Búsquedas (5)

- n Sería posible incorporar a las interfaces `ICollection` e `IDictionary` respectivamente las operaciones:
 - i `select(ICondition) : ICollection`
 - i `select(ICondition) : IDictionary`
- n La interfaz `ICondition` se define como



en cada realización `holds()` indicará si un cierto objeto debe formar parte del resultado del `select()`

[Búsquedas (6)]

- n ¿Cómo manejar las diferencias mencionadas entre las diferentes implementaciones?
 - i El tipo del iterador sería `Iterator`
 - i El tipo del resultado sería `Collection` o `Dictionary` respectivamente
 - i A su vez la condición encapsula:
 - n El o los parámetros de la búsqueda (en sus atributos)
 - n El algoritmo que determina si un elemento de la colección debe pertenecer además al resultado (en el método asociado a `holds()`)

[Búsquedas (7)]

- n Una posible implementación de `select()` en una realización de `ICollection` sería:

```
ICollection * List::select(ICondition * cond) {  
    ICollection * result = new List();  
    Iterator * it = getIterator();  
  
    while(it->hasCurrent()) {  
        if(cond->holds(it->current()))  
            result->add(it->current());  
        it->next();  
    }  
    return result;  
}
```

[Búsquedas (8)]

- n De esta forma las clases que implementan `ICondition` son estrategias concretas que el `select()` utiliza para construir la colección resultado
- n En esta aplicación de Strategy se dan las siguientes correspondencias:
 - i `List` à `Context`
 - i `ICondition` à `Estrategia`
 - i `select()` à `solicitud()`
 - i `holds()` à `algoritmo()`

[Búsquedas (9)]

n Ejemplo de condición concreta:

```
class Persona : ICollectible {  
private:  
    String nombre;  
    int edad;  
public:  
    Persona();  
    String getNombre();  
    int getEdad();  
}
```

```
// CondEdad.h  
class CondEdad : ICondition {  
private:  
    int valorEdad;  
public:  
    CondEdad(int i);  
    bool holds(ICollectible *ic);  
}  
  
// CondEdad.cpp  
CondEdad::CondEdad(int i) {  
    valorEdad = i;  
}  
  
bool CondEdad::holds(ICollectible *ic) {  
    Persona * p = (Persona *)ic;  
    return (p->getEdad() == valorEdad);  
}
```