Especificación de Módulos

- Información sobre el contenido del módulo y cómo utilizarlo
  - Nombre del módulo
  - Si es de datos o funcional
  - Nombre del tipo definido, si es distinto del del módulo
  - Descripción del tipo
  - Especificación pre/post de sus operaciones

- Esto es lo que ve el usuario. **NO** se puede dar información sobre cómo está implementado

Especificación Pre/Post de operaciones

- **Cabecera**: Acción (`void`)/función, nombre, lista de parámetros y/o resultados
- **Precondición**: propiedades que han de cumplir los parámetros para que la operación se pueda ejecutar
- **Postcondición**: propiedades que han de cumplir los resultados y parámetros modificables al acabar la ejecución

- **Objetivo**: abstracción funcional = que las operaciones funcionen sin limitarse a valores concretos de sus datos

Ejemplos de especificación Pre/Post

```
int potencia (int a, int b)
/* Pre: a>=0, b>0 */
/* Post: el resultado es “a” multiplicado
  por sí mismo “b” veces */

void intercambiar (int &a, int &b)
/* Pre: cierto */
/* Post: “a” pasa a tener el valor
  original de “b”; “b” pasa a tener el
  valor original de “a” */
```

Ejemplos de especificación Pre/Post

```
bool busqueda (const vector<int> &v, int x)
/* Pre: cierto */
/* Post: el resultado indica si x está en v */

void int_vector(vector<int> &v, int i, int j)
/* Pre: 0<= i,j < v.size() */
/* Post: v[i] y v[j] tienen sus valores
  intercambiados respecto a los originales */
```

Operaciones de una clase

En programación orientada a objetos, se considera que cada objeto es propietario de las operaciones que se pueden aplicar sobre él.

Salvo excepciones, las llamadas se realizan de la forma

`objeto.metodo(resto de parámetros)`

El objeto sobre el que se aplica la operación no aparece en la cabecera de la misma, por eso recibe el nombre de **parámetro implícito.** Ejemplo `vector`

- `size_type size() const` cabecera
- `v.size()` llamada

Clasificación de las operaciones de una clase

- Constructoras y destructoras
- Modificadoras
- Consultoras
- Lectura y escritura

Clase Estudiant

(ver fichero Estudiant_esp)

- Atributo DNI
- Atributo Nota
- Operaciones constructoras y destructoras
- Operaciones modificadoras de nota
- Operaciones consultoras: DNI, nota y tiene_nota
- Operaciones de lectura y escritura

Constructoras y destructoras

```
Estudiant();

Estudiant(int n);

~Estudiant();
```

Las constructoras se ejecutan cuando se declara un estudiante nuevo:

```
Estudiant est;

Estudiant est(44444444);
```

La destructora se llama internamente para borrar objetos locales al salir de un ámbito de visibilidad

Constructoras y destructoras

Las constructoras también intervienen al crear objetos por defecto, por ejemplo al dimensionar vectores:

```
vector<Estudiant> vest (n);
```

Se hace una llamada a `Estudiant()` y se crean `n` copias del resultado.

Por último, intervienen cuando un objeto es pasado como parámetro por valor (por eso es mejor evitar esta situación mediante el paso por **referencia constante**).

Modificadoras

```
void afegir_nota(double nota);
void modificar_nota(double nota);
```

Cambian valores del parámetro implícito.

```
    est.modificar_nota(x);
```

si `est` y `x` cumplen la precondición, la nota de `est` pasa a ser `x`

Consultoras

```
int consultar_dni() const;
double consultar_nota() const;
bool te_nota()  const;
```

Retornan información del parámetro implícito.

```
    int x = est.consultar_dni();
```

`x` pasa a valer el DNI de `est`

Consultoras

```
static double nota_maxima();
```

Retorna información común a toda la clase; no tiene parámetro implícito

```
double x = nota_maxima();
```

`x` pasa a valer la nota más alta que se le puede poner a *cualquier* estudiante

Desde fuera de la clase se ha de indicar a qué clase pertenece

```
double x = Estudiant::nota_maxima();
```

Lectura y escritura

```
void llegir_estudiant();

void escriure_estudiant() const;
```

Comunican el parámetro implícito con el canal standard de entrada/salida

```
est.llegir_estudiant();

// est pasa a ser el estudiante leído

est.escriure_estudiant();

// est se ha escrito en el canal
```