**TECNOLOGÍA DE PROGRAMACIÓN** **Curso 2015-2016**

# Práctica 1

## 1.- Descripción general de la práctica

El objetivo general de la practica es aprender los aspectos básicos de la sintaxis de la orientación a objetos, implementando un sencillo tipo abstracto de datos. Adicionalmente, esto permitirá comparar ciertas diferencias entre los lenguajes C++ y Java.

## 2.- TAD en C++

En este apartado diseñarás, en base a la sintaxis de C++, el tipo abstracto de datos `agrupacion`. Para ello te proporcionamos como material adjunto a esta práctica los siguientes ficheros:

- `agrupacion-struct.h` – El tipo abstracto de datos `agrupacion` definido mediante *structs* de C++ y funciones (quizá lo recuerdes de asignaturas anteriores). Tiene un iterador que se recorre la estructura desde el último elemento introducido hasta el primero.
- `main-struct.cc` – Un programa principal que utiliza el tipo de datos definido en el fichero anterior para guardar y mostrar una secuencia de enteros por pantalla.
- `agrupacion-class.h` – Es el esqueleto de la definición del tipo de datos `agrupacion` definido mediante clases y métodos en C++. Deberás rellenar los huecos de este fichero en aquellos lugares donde pone “TODO”.
- `main-class.cc` – El programa principal que utiliza el tipo de datos definido en el fichero anterior. Debería compilar y funcionar correctamente una vez hayas implementado los métodos en el fichero anterior.
- `Makefile` – Fichero para ejecutar `make` y compilar todo lo anterior.

Deberás implementar una serie de métodos en `agrupacion-class.h` para que se comporte de forma equivalente a las funciones implementadas en `agrupacion-struct.h`. Los métodos que tienes que implementar están marcados en `agrupacion-class.h` con un comentario que empieza con “TODO:” seguido de instrucciones específicas en cada caso. Hay un total de 6.

Hay una serie de diferencias entre ambas implementaciones que es importante que tengas en cuenta:

- No existe función *iniciar*. Casi la totalidad de los lenguajes orientados a objetos permiten la definición de un **constructor**, que hace las veces de inicializador, que tiene el mismo nombre que la clase y que se ejecuta siempre que se crea en memoria el objeto correspondiente.
- Al contrario que en las funciones (en la que el parámetro que representa al TAD se declara específicamente) en una clase los métodos tienen un parámetro implícito *this* que es un puntero que apunta a la propia clase y que permite acceder a los atributos y métodos del propio objeto. En la mayoría de los casos se puede omitir.
- La implementación de los **iteradores** es diferente: el iterador (`const_iterator`) es una clase definida dentro de la clase global que

contiene los atributos y métodos exclusivos de los iteradores. Al separar el iterador de la propia estructura de datos, se pueden tener varios iteradores recorriéndose simultáneamente la propia estructura, y se pueden definir iteradores constantes.
- Para seguir el estándar definido por la **Standard Template Library** (STL) de C++, los iteradores tienen que tener definidos ciertos métodos de forma diferente: la inicialización del iterador es un constructor, el avance y el acceso al iterador están en métodos separados con nombres específicos y la comprobación de finalización se hace comparando iteradores al inicio y al final de la estructura de datos (ver el esqueleto que proporcionamos para conocer más detalles). Esto permite recorrerse la estructura de datos con un bucle específico del lenguaje, como se puede ver en el programa principal.

## 3.- TAD en Java

Como primera aproximación al lenguaje Java, vamos a diseñar el mismo tipo abstracto de datos que en el apartado anterior pero en el lenguaje Java. De nuevo, te proporcionamos material:

- `Agrupacion.java` – Es el esqueleto de la definición del tipo de datos, que deberás rellenar.
- `Main.java` – Es el programa principal que prueba el tipo de datos definido en la clase anterior.

Deberás implementar una serie de métodos en `Agrupación.java` para que se comporte de forma equivalente a la estructura de datos definida en C++ en el apartado anterior. El código a rellenar está marcado con comentarios que empiezan con “TODO:” seguido de instrucciones específicas en cada caso. Hay un total de 4.

Dado que es teóricamente tu primer contacto con el lenguaje Java, te enumeramos las diferencias principales con respecto a C++:

- Java es un lenguaje puramente **orientado a objetos**. Esto implica, entre otras cosas, que no existe el concepto de función: todo son métodos que pertenecen a alguna clase.
- Java es un lenguaje **compilado e interpretado**. Para compilar y ejecutar el ejemplo que te proporcionamos deberás utilizar los siguientes comandos:

```
javac Main.java
java Main
```

- Todas las clases y vectores en Java son representadas en memoria mediante **punteros**. Esto implica que no hay dos operadores “.” y “→”, sino que únicamente se utiliza el operador “.”. Afortunadamente, en Java hay recolección automática de basura, así que no te tienes que preocupar de la gestión de la memoria dinámica.
- Para los **iteradores**, utilizamos en el esqueleto que te proporcionamos la estructura de la biblioteca estándar de Java. Esto hace cierto uso de la herencia (palabra clave *implements*), mecanismo que es posible que todavía no hayamos visto en clase. De momento haremos acto de fé.
- Sorprendentemente, hay pocas diferencias más, aparte de las puramente

sintácticas.

## 4.- Entrega

Todos los archivos deben estar contenidos en un directorio de nombre **`practica1_<NIP>`** si la realización es individual, o **`practica1_<NIP1>_<NIP2>`** si es por parejas. La estructura de archivos debe ser la siguiente:

```
practica1_XXXXXX
     \---- java
          \--- Agrupacion.java
     \--- c++
          \--- agrupacion-class.h
```

El archivo `agrupacion-class.h` debe ser compilable y ejecutable con los archivos `Makefile` y `main-class.cc` que te proporcionamos sin ninguna modificación, mediante:

```
make
main-class
```

El archivo `Agrupacion.java` programa Java debe ser compilable y ejecutable con el archivo `Main.java` que te proporcionamos, sin ninguna modificacion, mediante:

```
javac Main.java
java Main
```

En caso de no compilar siguiendo estas instrucciones la evaluación de la práctica será de 0.

El directorio se comprimirá en un archivo con el mismo nombre y extensión `.zip`. El archivo comprimido a entregar no debe contener ningún fichero aparte de los fuentes que te pedimos: ningún fichero `.class`, `.o` (objeto), ejecutable, `Makefile` o ningún otro archivo de código fuente que te proporcionemos.

No se deben utilizar paquetes ni librerías ni ninguna infraestructura adicional fuera de las librerías estándar de los propios lenguajes.

El fichero `.zip` se entregará a través de Moodle. Si la realización es por parejas, sólo uno de los dos miembros de la pareja deberá entregar la práctica.