UNIVERSIDAD DE LA RIOJA

Asignatura:	**ESTRUCTURA DE LA INFORMACIÓN EN PROGRAMACIÓN**
Código Asignatura:	2012015
Titulación:	**Licenciatura en Matemáticas (Plan 2003)**
Departamento:	Matemáticas y Computación
Tipo de asignatura:	Obligatoria
Curso:	2º
Cuatrimestre:	PRIMERO
Conocimientos previos:	

CRÉDITOS: 6 **Teóricos**: 2.5
Prácticos: aula 0.5 laboratorio 3 campo

OBJETIVOS:

1. Conocer las nociones de objeto y clase tal y como se definen en Programación Orientada a Objetos (POO) y aplicarlas en la representación de información
   - 1.1 Comparar la representación de Tipos Abstractos de Datos mediante clases o registros
   - 1.2 Distinguir entre las nociones de estado y comportamiento de un objeto
   - 1.3 Aplicar el concepto de encapsulación de la información a distintos ejemplos. Conocer los modificadores de acceso "público" y "privado" en una clase y su comportamiento
   - 1.5 Representar clases, atributos y métodos por medio de diagramas UML
   - 1.5 Codificar en los lenguajes de programación Java y C++ los anteriores conceptos

2. Distinguir entre tipos de relaciones entre clases y utilizar estas relaciones en la definición de nuevas clases
   - 2.1 Definir ejemplos de clases que contienen objetos como atributos
   - 2.2 Conocer y definir ejemplos de relaciones de especialización / generalización
   - 2.3 Relaciones de herencia en POO. Definición y utilización de las mismas
   - 2.4 Aplicar la redefinición de métodos sobre ciertos casos de herencia
   - 2.5 Representar relaciones de herencia por medio de diagramas UML
   - 2.6 Codificar en Java y C++ los distintos conceptos y ejemplos estudiados

3. Conocer la noción de polimorfismo en POO y aplicarla en distintos casos de uso
   - 3.1 Diseñar casos de uso que permitan observar el comportamiento de métodos polimorfos
   - 3.2 Aplicar el polimorfismo en distintos escenarios y observar sus ventajas de uso en cuanto a reutilización de código
   - 3.4 Codificar en Java y C++ los distintos conceptos y ejemplos estudiados

4. Definir los conceptos de método abstracto y clase abstracta, observar su utilidad y diseñar clases que hagan uso de tales conceptos
   - 4.1 Utilizar métodos abstractos en clases de diseño propio
   - 4.2 Observar la necesidad del polimorfismo en la utilización de clases abstractas
   - 4.3 Aplicar clases completamente abstractas en el diseño de clases; observar cómo pueden ayudar en la definición de clases
   - 4.4 Representar clases abstractas e interfaces en diagramas UML
   - 4.4 Codificar en Java y C++ los distintos conceptos y ejemplos estudiados

5. Definir la noción de excepción, reconocer sus ventajas en el diseño de métodos y definir métodos que hagan uso de excepciones
   - 5.1 Definir excepciones propias
   - 5.2 Utilización de excepciones: distinguir entre crear una excepción, lanzarla y gestionarla, y las posibles formas de uso que ofrecen
   - 5.3 Introducir excepciones en programas para incrementar su fiabilidad

6. Diseñar y desarrollar soluciones a problemas que involucren los distintos conceptos introducidos en la asignatura
   - 6.1 Representar soluciones a problemas de programación por medio de diagramas UML
   - 6.2 Codificar las soluciones en Java y en C++

PROGRAMA TEÓRICO Y PRÁCTICAS DE AULA:

Tema 1. Nociones de clase y objeto en Programación Orientada a Objetos
   - 1.1 Representación de la información por medio de objetos
   - 1.2 Atributos o estado
   - 1.3 Métodos o comportamiento
   - 1.4 Abstracción de objetos en clases
   - 1.5 Necesidad y relevancia de los constructores de clase: constructor por defecto, constructores propios
   - 1.6 Métodos de acceso y modificación del estado de un objeto

1.7 Modificadores de acceso: relevancia y necesidad de los modificadores público y privado
1.8 Encapsulación de la información: distintas formas de representar una misma clase manteniendo su comportamiento
1.9 Introducción al lenguaje de especificación UML: utilización para representar clases y objetos
1.10 Lenguaje de programación C++: declaración de clases y construcción de objetos
1.11 Lenguaje de programación Java: declaración de clases y construcción de objetos

(9 horas)

Tema 2. Relaciones entre clases. Herencia entre clases
2.1 Comunicación entre distintas clases
2.2 Clases que contienen objetos como atributos: algunos ejemplos conocidos
2.3 Relaciones de especialización / generalización
2.4 Definición de la relación de herencia entre clases
2.5 Ventajas del uso de relaciones de herencia: reutilización de código
2.6 Redefinición de métodos en clases heredadas
2.7 Modificador de acceso "protegido": posibilidades de uso
2.8 Representación de relaciones de herencia en diagramas UML
2.9 Programación en Java y C++ de relaciones de herencia

(6 horas)

Tema 3. Definición y uso de métodos polimorfos
3.1 Definición de polimorfismo y ventajas de uso
3.2 Utilización de métodos polimorfos sobre ejemplos ya construidos
3.3 Programación de métodos polimorfos en C++: utilización de memoria dinámica y métodos virtual
3.4 Polimorfismo en Java

(3 horas)

Tema 4. Clases abstractas e interfaces.
4.1 Definición de métodos abstractos en POO. Algunos ejemplos de uso
4.2 Relación entre polimorfismo y métodos abstractos
4.3 Definición y ventajas de uso de clases completamente abstractas o interfaces
4.4 Representación en UML de interfaces y métodos abstractos
4.5 Implementación en C++ de métodos abstractos y clases abstractas
4.6 Implementación en Java de métodos abstractos e interfaces

(4 horas)

Tema 5. Excepciones en Java
5.1 Definición de excepciones en programación
5.2 Tipos de excepciones / errores y cómo tratarlos
5.3 Trabajando con excepciones: declaración, construcción, lanzamiento y gestión de excepciones
5.4 Programación de excepciones en Java. Utilización de excepciones de la librería y definición de excepciones propias

(3 horas)

PROGRAMA DE PRÁCTICAS EN LABORATORIO Y CAMPO:

Se realizará 15 sesiones prácticas de forma semanal de 2 horas de duración. El contenido de las sesiones será el siguiente (la temporalización de los contenidos podría no coincidir con las sesiones)

Práctica 1. Declaración y definición de clases en C++ y comparación con el uso de registros. Definición de atributos, métodos, constructores, modificadores de acceso y primeros programas usando objetos

Práctica 2. Declaración y definición de clases en Java. Utilización de objetos en programas. Entrada por teclado en Java y salida por consola y ficheros

Práctica 3. Ejemplos de encapsulación de la información. Distintas formas de representar el estado de una clase en C++ y Java

Práctica 4. Relaciones entre clases. Definición de nuevas clases utilizando objetos de clases ya existentes. Ejemplos básicos: vectores y listas de objetos en C++ y Java

Práctica 5. Relaciones de herencia entre clases en Java y C++. Definición y uso de las mismas. Redefinición de métodos

Práctica 6. Polimorfismo. Definición de métodos polimorfos en C++ y Java y utilización con métodos redefinidos. Diferencias de uso entre Java y C++

Práctica 7. Definición de métodos abstractos, clases abstractas y clases completamente abstractas en C++. Definición de métodos abstractos, clases abstractas e interfaces en Java.

Práctica 8. Declaración, construcción, y gestión de excepciones en Java. Ejemplos de uso.

Práctica 9. Utilización de la librería de Java. Excepciones, clases predefinidas, tipos de estructuras más comunes

Práctica 10. Diseño e implementación de un ejemplo de sistema de información por medio de las técnicas aprendidas de POO

SISTEMA Y CRITERIOS DE EVALUACIÓN:

La evaluación de la asignatura estará dividida en dos partes. Por una parte, la asistencia y aprovechamiento de las prácticas. Por otra, un examen de conocimientos al final del cuatrimestre.

La asistencia y aprovechamiento de las prácticas será un requisito indispensable para poder optar al examen final. La asistencia a prácticas debe ser superior al 80% de las mismas. El aprovechamiento de las prácticas se considerará superado siempre y cuando el alumno entregue las mismas en los plazos estipulados (estos plazos serán siempre anteriores a la fecha del examen, y estarán repartidos a los largo del cuatrimestre para promover el trabajo continuo del alumno).

El examen teórico al final del cuatrimestre estará destinado a valorar si el alumno ha alcanzado los objetivos anteriormente expuestos. Estará dividido en dos partes. La primera será de tipo test o preguntas cortas, y se centrará en aspectos teóricos de la asignatura. En la segunda parte los alumnos tendrán que diseñar soluciones a ciertos problemas de programación y codificarlas en los distintos lenguajes estudiados en la asignatura.

La nota final en la asignatura podrá contemplar otros aspectos, tales como la participación de los alumnos, la realización de tareas propuestas, ...

BIBLIOGRAFÍA BÁSICA:

B. Meyer, Construcción de software orientado a objetos, Prentice-Hall 1998.
B. Stroustrup, El lenguaje de Programación C++, Addison Wesley 2001.
B. Eckel, Piensa en Java (2ª edición), Prentice Hall 2002.
C. Muñoz, A. Niño, A. Vizcaíno, Introducción a la Programación con Orientación a Objetos, Prentice-Hall 2002.

MATERIALES PARA PRÁCTICAS:
Entorno de programación para C++ (dominio público) Dev-C++:
http://www.bloodshed.net/devcpp.html
Entorno de programación para Java (dominio público) JCreator:
http://www.jcreator.com

PROFESOR / PROFESORES RESPONSABLES:

Jesús María Aransay Azofra