6. Programación orientada a objetos

INTRODUCCIÓN

Este módulo de 228 horas pedagógicas inicia el aprendizaje práctico de los diferentes aspectos de programación orientada a objetos, simulando un ambiente de trabajo propicio para pruebas y desarrollo de aplicaciones, de acuerdo a las necesidades de la industria. El objetivo principal en esta etapa es que los y las estudiantes puedan resolver situaciones propias de un desarrollo bajo estos conceptos. Para ello, tendrán que concretar soluciones informáticas, ampliando la lógica resolutiva de casos de negocios empresariales con la programación Java.

La programación orientada a objetos es un paradigma que utiliza objetos como elementos fundamentales en la construcción de las soluciones de cualquier problema. Un objeto es una abstracción de algún hecho o ente del mundo real, con atributos que representan sus características o propiedades y métodos que emulan su comportamiento o actividad.

Así, se pretende que cada estudiantes desarrolle conocimientos suficientes para abordar el estudio de cualquier lenguaje OO, metodología de análisis y diseño OO de los sistemas gestores de bases de datos OO, y en general de cualquier materia basada en el modelo orientado a objetos.

Las clases son prácticas y teóricas. Para el desarrollo de ellas se usa el IDE NetBeans junto con MySQL y MySQLConnector/J como entorno de desarrollo, por el sentido pedagógico y de *software* libre para la educación, a diferencia de otras herramientas que son de uso personal. Además, se espera que los y las estudiantes apliquen sus conocimientos y habilidades previas en diagramación.

Se sugiere que el o la docente complemente sus exposiciones con código fuente como ejemplo para las diferentes soluciones informáticas. Para ello, se realizará la codificación en el laboratorio con un PC asignado para tal efecto.

Es importante que las actividades incorporen metodologías activas, centradas en los y las estudiantes y en el desarrollo de los Objetivos de Aprendizaje Genéricos, como el trabajo colaborativo y el cumplimiento de estándares, normativa vigente y de los protocolos asociados.

APRENDIZAJES ESPERADOS Y CRITERIOS DE EVALUACIÓN

MÓDULO 6 · PROGRAMACIÓN ORIENTADA A OBJETOS	228 HORAS	CUARTO MEDIO

OBJETIVOS DE APRENDIZAJE DE LA ESPECIALIDAD

OA 4
Construir aplicaciones computacionales basadas en programación orientada a objetos, de manera de cumplir con las exigencias técnicas y de los usuarios.

APRENDIZAJES ESPERADOS		CRITERIOS DE EVALUACIÓN	OBJETIVOS DE APRENDIZAJE GENÉRICOS
1.	Construye unidades de prueba para verificar el correcto funcionamiento de la codificación realizada, de acuerdo a exigencias técnicas de confiabilidad.	**1.1** Realiza pruebas para detectar problemas previos a la codificación de una unidad de *software* con una herramienta de *software* disponible en el mercado.	C
		1.2 Construye una unidad de prueba para detectar excepciones en un entorno de lenguaje con una herramienta compatible que permita desarrollar, configurar e implementar aplicaciones en función del lenguaje utilizado.	C
		1.3 Construye una unidad de prueba para detectar casos límites, en un entorno de lenguaje, con una herramienta compatible, propia del *software* utilizado como marco de trabajo.	C

APRENDIZAJES ESPERADOS		CRITERIOS DE EVALUACIÓN	OBJETIVOS DE APRENDIZAJE GENÉRICOS
2.	Detecta y corrige errores de codificación, analizando el comportamiento del código de programación, de acuerdo a especificaciones y manuales de referencia.	**2.1** Utiliza la herramienta de depuración de un entorno de desarrollo, para revisar la codificación de clases programadas.	C
		2.2 Revisa y corrige las variables, del programa desarrollado para la solución de un caso, indagando en las funciones internas de acuerdo a protocolos de revisión.	C
		2.3 Revisa las variables de una función de clases programadas, utilizando opciones del lenguaje empleado, para asegurar su correcto funcionamiento.	C
3.	Construye aplicaciones con clases predefinidas, utilizando herramientas del lenguaje de programación para resolver problemas complejos, de acuerdo a especificaciones técnicas.	**3.1** Usa clases complejas propias del marco de aplicación, que permiten establecer la conexión con la base de dato, para gestionar solicitudes del usuario.	C
		3.2 Utiliza clases propias del *software* de trabajo, que están asociadas a la inserción, modificación y eliminación, con sentencias del lenguaje estándar en uso, para resolver problemas de gestión de la BD, haciendo correcto uso de manuales de referencia.	B
		3.3 Almacena resultados de consultas, utilizando clases propias del *software* y que están orientadas específicamente a consultas, haciendo uso de la interfaz de programación.	H
		3.4 Agiliza la relación entre la aplicación y la base de datos, utilizando clases asociadas a un mapeador objeto-relacional, para configurar la conexión y mantener la persistencia de los datos, con independencia del entorno de desarrollo.	H
4.	Utiliza componentes reutilizables en un lenguaje de uso estándar, para ser aplicados en programas de diversa índole, de acuerdo a requerimientos y especificaciones técnicas.	**4.1** Desarrolla clases asociadas a la presentación que puedan ser reutilizadas, para satisfacer el requerimiento de visualización por parte de un usuario para aplicación Web, comprobando errores y calidad de comunicación.	C
		4.2 Utiliza clases asociadas a la capa de negocio de la aplicación que puedan ser reutilizadas, para resolver requerimiento principal, comprobando que tenga comunicación con la presentación y los datos.	C

EJEMPLO DE ACTIVIDAD DE APRENDIZAJE

NOMBRE DEL MÓDULO	**Programación orientada a objetos**
NOMBRE DE LA ACTIVIDAD DE APRENDIZAJE	Construcción de una unidad de prueba para verificar un resultado esperado
DURACIÓN DE LA ACTIVIDAD	4 horas
APRENDIZAJES ESPERADOS	CRITERIOS DE EVALUACIÓN QUE INCLUYE
1. Construye unidades de prueba para verificar el correcto funcionamiento de la codificación realizada, de acuerdo a exigencias técnicas de confiabilidad.	1.1 Realiza pruebas para detectar problemas previos a la codificación de una unidad de *software* con una herramienta de *software* disponible en el mercado.
METODOLOGÍAS SELECCIONADAS	Detección de fallas

DESCRIPCIÓN DE LAS TAREAS QUE REALIZAN DOCENTES Y ESTUDIANTES, Y LOS RECURSOS QUE SE UTILIZAN EN CADA UNA DE LAS SIGUIENTES ETAPAS

PREPARACIÓN DE LA ACTIVIDAD

Docente:

- Prepara el laboratorio con puestos de trabajo y *software* de trabajo instalado en el computador.
- Elabora una presentación (PPT) del caso de pruebas con uso de herramientas de depuración.
- Genera una guía con el planteamiento y los alcances del trabajo a realizar.

Recursos:

- Laboratorio con puestos de trabajo y energía disponible.
- Computador.
- NetBeans.
- MySQL.
- MySQLConnector/J.
- Java y otros *software* actualizados.
- Manuales de uso y referencia de *software* utilizado.

DESCRIPCIÓN DE LAS TAREAS QUE REALIZAN DOCENTES Y ESTUDIANTES, Y LOS RECURSOS QUE SE UTILIZAN EN CADA UNA DE LAS SIGUIENTES ETAPAS:	
EJECUCIÓN	**Docente:** › Explica y contextualiza la actividad de la clase. › Expone la importancia de las pruebas de caja negra sobre una aplicación o los componentes de la aplicación, ya sea de escritorio o web. › Muestra los criterios para crear un plan de pruebas, reconociendo la importancia que posee tanto en la planificación del proyecto sobre los requisitos funcionales y no funcionales como en la calidad esperada de la solución. › Propone un ejercicio para realizar pruebas con JUnit a partir de un plan de pruebas, ilustrando cada una de las alternativas y casos en que una aplicación puede generar errores, excepciones o situaciones de borde, aplicando la orientación a objeto obtenida por el estándar O.R.M. (Mapeo Objeto-Relacional). › Entrega guía de procedimiento de trabajo, los archivos a probar, los manuales y las guías de referencia de los *software* involucrados, en medio manual o digital, para construir una prueba de resultados. **Estudiantes:** › Observan la presentación realizada para realizar el procedimiento de la clase. › Revisan la guía y los manuales entregados. › Analizan el caso y las alternativas de uso del lenguaje para su solución. › Activan los programas de entorno necesarios para trabajar y desarrollan las sentencias en función del objetivo. › Explican la relación entre las sentencias utilizadas y el objetivo esperado de solución. › Reconocen errores y corrigen.
CIERRE	**Estudiantes:** › En un plenario dan cuenta de la experiencia de la actividad realizada y las dificultades más comunes. › Formulan conclusiones del trabajo. **Docente:** › Retroalimenta la actividad de clase, comentando las soluciones y aclarando los errores del procedimiento. › Profundiza el funcionamiento de los *assert* y los *fail* que ocurren dentro de una prueba unitaria, teniendo en cuenta cómo se compara la orientación a objetos con un resultado esperado. › Aclara que JUnit es también un medio de controlar las pruebas de regresión necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

EJEMPLO DE ACTIVIDAD DE APRENDIZAJE

NOMBRE DEL MÓDULO	**Programación orientada a objetos**
NOMBRE DE LA ACTIVIDAD DE APRENDIZAJE	Prueba de comportamiento de código con verificación paso a paso
DURACIÓN DE LA ACTIVIDAD	4 horas
APRENDIZAJES ESPERADOS	CRITERIOS DE EVALUACIÓN QUE INCLUYE
2. Detecta y corrige errores de codificación, analizando el comportamiento del código de programación, de acuerdo a especificaciones y manuales de referencia.	2.1 Utiliza la herramienta de depuración de un entorno de desarrollo, para revisar la codificación de clases programadas.
METODOLOGÍAS SELECCIONADAS	Detección de fallas

DESCRIPCIÓN DE LAS TAREAS QUE REALIZAN DOCENTES Y ESTUDIANTES, Y LOS RECURSOS QUE SE UTILIZAN EN CADA UNA DE LAS SIGUIENTES ETAPAS

PREPARACIÓN DE LA ACTIVIDAD

Docente:

- Prepara el laboratorio con puestos de trabajo y energía disponible.
- Prepara el *software* de trabajo instalado en computador.
- Elabora una presentación de caso para pruebas con uso de herramientas de depuración.
- Genera una guía con el planteamiento y los alcances del trabajo a realizar.

Recursos:

- Laboratorio con puestos de trabajo y energía disponible.
- Computador.
- NetBeans.
- MySQL.
- MySQLConnector/J.
- Java u otro *software* actualizado.
- PowerPoint.
- Manuales de uso y referencia de *software* utilizado.

DESCRIPCIÓN DE LAS TAREAS QUE REALIZAN DOCENTES Y ESTUDIANTES, Y LOS RECURSOS QUE SE UTILIZAN EN CADA UNA DE LAS SIGUIENTES ETAPAS	
EJECUCIÓN	**Docente:** › Explica y contextualiza la actividad a realizar en la clase. › Hace una presentación en PPT para profundizar las pruebas de caja blanca en la que explica cómo ocupar y conocer los diferentes comportamientos del código y analiza variables, instrucciones paso a paso y saltos de instrucciones. › Muestra la ejecución y utilización de las herramientas en el caso preparado y analiza los resultados. › Propone un ejercicio práctico en el que se analiza un método de una clase en un *debug*, creando un plan de resultados parciales para la prueba de caja blanca. › Entrega la guía de procedimiento de trabajo y manuales y guías de referencia del *software* involucrado, en medio manual o digital. **Estudiantes:** › Observan la presentación realizada con el contenido de la clase. › Revisan la guía y los manuales entregados. › Analizan el caso propuesto y el código y determinan los comandos a utilizar para la depuración. › Activan los programas de entorno necesarios para trabajar. › Utilizan los comandos necesarios de control y la verificación paso a paso. › Explican la relación entre las sentencias utilizadas y el objetivo esperado de prueba. › Reconocen los errores y corrigen.
CIERRE	**Estudiantes:** › Comentan la experiencia de la actividad y las dificultades encontradas. › Exponen sus conclusiones en cuanto al trabajo realizado. **Docente:** › Comenta con sus estudiantes en torno a los problemas encontrados y las alternativas de solución. › Concluye valorando la prueba de caja blanca y la importancia de experimentar cada uno de los métodos desarrollados, incentivando al y la estudiante a que justifique el desarrollo de su código con ejemplos descriptivos.

EJEMPLO DE ACTIVIDAD DE EVALUACIÓN

NOMBRE DEL MÓDULO	**Programación orientada a objetos**

APRENDIZAJE ESPERADO	CRITERIOS DE EVALUACIÓN	OBJETIVOS DE APRENDIZAJE GENÉRICOS A EVALUAR
1. Construye unidades de prueba para verificar el correcto funcionamiento de la codificación realizada, de acuerdo a exigencias técnicas de confiabilidad.	**1.1** Realiza pruebas para detectar problemas previos a la codificación de una unidad de *software* con una herramienta de *software* disponible en el mercado.	**c** Realizar las tareas de manera prolija, cumpliendo plazos establecidos y estándares de calidad, y buscando alternativas y soluciones cuando se presentan problemas pertinentes a las funciones desempeñadas.

Selección de cómo evaluar

DESCRIPCIÓN DE ACTIVIDADES DE EVALUACIÓN	INSTRUMENTOS DE EVALUACIÓN SELECCIONADOS
Construcción de una unidad de prueba para verificar un resultado esperado. Se evalúa a través de una actividad práctica de taller. El o la estudiante realiza la actividad, siguiendo el procedimiento establecido en protocolos, en el marco de las normas en ambiente de pruebas, con cuidado en las reglas de sintaxis y de las sentencias del lenguaje.	Escalas de apreciación.

Ejemplo de escala de apreciación

INDICADORES	CONCEPTOS			
	SIEMPRE	GENERALMENTE	OCASIONALMENTE	NUNCA
Aplica protocolo establecido.				
Usa correctamente las herramientas (PC, CD, *software*, manuales, guías).				
Mantiene en orden el espacio de trabajo.				
Prepara los materiales.				
Detecta los errores.				
Corrige los errores.				
Considera aspectos normativos de seguridad personal.				

BIBLIOGRAFÍA

Luque, I. (2002). *Bases de datos: Desde Chen hasta Codd con Oracle*. Ciudad de México: Alfaomega.

Deitel, M. y Deitel, J. (2010). *Java como programar*. São Paulo: Pearson Prentice Hall.

Eckel, B. (2008). *Piensa en Java*. Madrid: Prentice Hall.

Froufe, A. (2008). *Java 2: Manual de usuario y tutorial*. Madrid: RA-MA.

Gamma, E. (2006). P*atrones de diseño: Elementos de software orientado a objetos reutilizable*. Madrid: Pearson-Adison-Wesley.

Houlette, F. (2001). *Fundamentos de SQL*. Ciudad de México: McGraw-Hill.

Martín, A. y Amaya, R. (2009). *Curso de programación en Java-J2EE*. Madrid: Grupo Syncrom.

Moss, K. (1999). *Java servlets*. New York: McGraw-Hill.

Schildt, H. (2005). *La Biblia de Java 2 v5.0*. Madrid: Anaya Multimedia.

Sitios web recomendados

Java:
http://www.java.com/es/download

MySQL:
http://dev.mysql.com/doc/

(Los sitios web y enlaces sugeridos en este Programa fueron revisados en marzo de 2015).