DigitalHouse>
Coding School

Certified Tech Developer

The Ultimate Degree

Materia: Programación Orientada a Objetos

Fundamentación

El mundo se encuentra en constante cambio y evolución, al igual que el llamado mundo IT. En los años 70 y frente a la llamada crisis del software, empezó a idearse la premisa de que se puede abstraer la realidad mediante los llamados objetos, facilitando el proceso de creación de software al reflejar la realidad, naciendo de esta forma, el paradigma orientado a objetos. No fue sino hasta la década de los 90 en que se comenzó a popularizar dicho paradigma mediante la proliferación de los llamados, valga la redundancia, lenguajes de programación orientados a objetos.

Hoy en día, la programación orientada a objetos constituye uno de los pilares fundamentales para cualquier developer, es utilizada para la creación tanto de sistemas empresariales como también para aplicaciones web y móviles.

Además la preparación de programación orientada a objetos es una base indispensable para la inserción en equipos de trabajo, no importa dónde se trabaje, el paradigma está presente.

Objetivos de aprendizaje

El cursado de la materia permitirá que el alumno adquiera las bases y desarrolle la capacidad de programar trabajos en todas las ramas del desarrollo de software, desde la perspectiva del paradigma orientado a objetos. También le va a permitir comprender y analizar los diferentes desafíos que enfrentan los actuales equipos de trabajo al momento de desarrollar.

Los conceptos se aplicarán en el lenguaje de programación Java, uno de los más utilizados para desarrollos en empresas IT hoy en día y el conocimiento de patrones de diseño que resultan fundamentales al momento de diseñar un software.

Metodología de enseñanza-aprendizaje

Desde Digital House, proponemos un modelo educativo que incluye entornos de aprendizaje sincrónicos y asincrónicos con un enfoque que vincula la teoría y la práctica, mediante un aprendizaje activo y colaborativo.

Nuestra propuesta incluye clases en vivo con tu grupo de estudiantes y docentes, a los que podrás sumarte desde donde estés. Además, contamos con un campus virtual a medida, en el cual encontrarás las clases virtuales, con actividades, videos, presentaciones y recursos interactivos, para realizar a tu ritmo antes de cada clase en vivo.

A lo largo de tu experiencia de aprendizaje en Digital House lograrás desarrollar habilidades técnicas y blandas, como ser el trabajo en equipo, creatividad, responsabilidad, compromiso, comunicación efectiva y autonomía.

En Digital House utilizamos la metodología de “aula invertida”. ¿Qué quiere decir? Cada semana te vamos a pedir que te prepares para la que sigue, leyendo textos, viendo videos, realizando actividades, entre otros recursos. De esta forma, cuando llegues al encuentro en vivo, estarás preparado para abordar el tema de manera más rica.

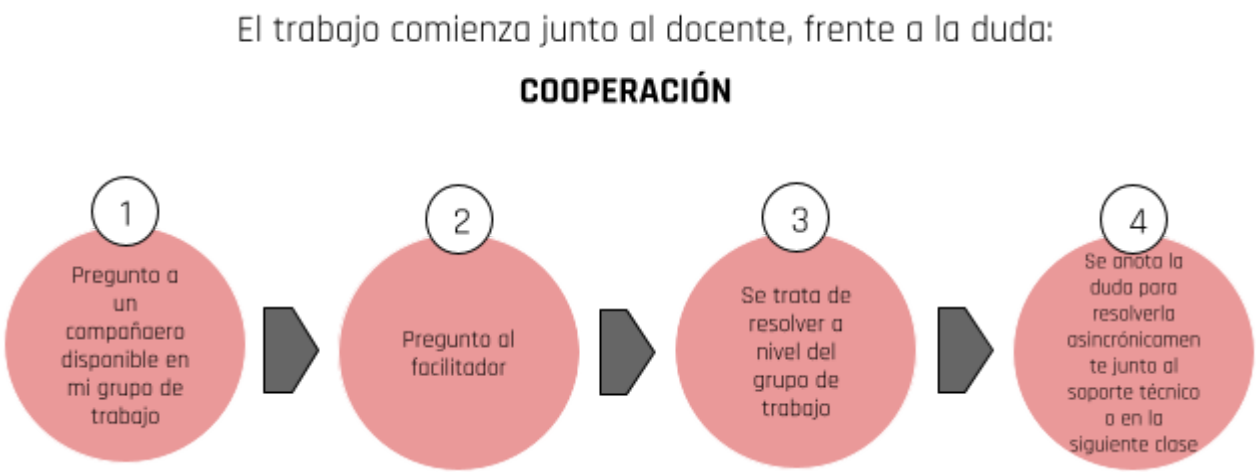
Utilizamos actividades y estrategias basadas en los métodos participativos y activos para ponerte en movimiento ya que uno solo sabe lo que hace por sí mismo. Por ese motivo, organizamos las clases para que trabajes en ella de verdad y puedas poner en práctica las distintas herramientas, lenguajes y competencias que hacen a la formación de un programador. Concebimos la clase como espacio de trabajo.

Una de las cuestiones centrales de nuestra metodología de enseñanza es el aprendizaje en la práctica. Por ese motivo, a lo largo de la cursada estarán muy presentes las ejercitaciones, es decir, la práctica de actividades de diversos tipos y niveles de complejidad que te permitirán afianzar el aprendizaje y comprobar que lo hayas asimilado correctamente. De esta forma, se logra un aprendizaje más significativo y profundo, la asimilación de los conocimientos de manera más eficaz y duradera, relacionar lo aprendido con la realidad de los programadores, fomentar la autonomía y el autoconocimiento, mejorar el análisis, la relación y la comprensión de conceptos, además, ayuda a ejercitar multitud de competencias.

El aprendizaje entre pares es uno de los elementos centrales de nuestra metodología, por eso, en cada clase te propondremos que trabajes en mesas de trabajo junto a tus compañeros y, a lo largo de la cursada, iremos variando la composición de los grupos para potenciar la cooperación. Lo que se propone es un cambio de mirada sobre el curso en cuestión, ya no se contempla al estudiante transitando su camino académico de manera individual, sino como parte de un equipo que resulta de la suma de las potencialidades de cada uno. La distribución en grupos de trabajo fomenta la diversidad y el aprovechamiento del potencial de cada integrante para mejorar el rendimiento del equipo.

La explicación recíproca como eje del trabajo cotidiano no solo facilita el aprendizaje de los compañeros, sino que sobre todo potencia la consolidación de conocimientos por parte de quien explica. Se promueve la responsabilidad, la autonomía, la proactividad, todo en el marco de la cooperación. Lo que lleva a resignificar la experiencia de aprendizaje y a que la misma esté vinculada con emociones positivas.

El trabajo cooperativo permite entablar relaciones responsables y duraderas, aumenta la motivación y el compromiso, además promueve un buen desarrollo cognitivo y social. La cooperación surge frente a la duda: si un estudiante tiene una pregunta, le consulta a algún miembro de su grupo asignado que esté disponible. Si la duda continúa, se convoca al facilitador. Si no lo resuelven, el facilitador pedirá a todos que se detengan para cooperar como equipo en la resolución del conflicto que ha despertado la duda. Así debatirán todos los integrantes de la mesa buscando la solución. Si aún así no pueden resolverlo, anotarán la duda que será abordada asincrónicamente por el soporte técnico o de forma sincrónica en la siguiente clase por parte del profesor.



Todos los días, finalizada la jornada, los estudiantes reconocerán a uno de los integrantes del grupo con quienes compartió ese día. El criterio para ese reconocimiento es la cooperación.

Cada grupo tendrá un facilitador que será elegido a partir de los reconocimientos y generando un sistema de rotación donde cualquiera pueda pasar por ese rol. El facilitador no es una figura estática, sino que cumple un rol dinámico y versátil. El facilitador es un estudiante que moviliza el alcance de los objetivos comunes del equipo poniendo en juego la cooperación. Es aquel que comparte con la mesa su potencial en favor del resto del equipo y que, por lo tanto, promueve la cooperación.

Información de la materia

- Modalidad 100% a distancia
- Cantidad de semanas totales: 9
- Clases virtuales en nuestro campus Playground: 54
- Cantidad de clases en vivo: 27

Requisitos y correlatividades

¿Qué materias tiene que tener cursada el alumno previamente?

- Programación imperativa

¿Qué materias cursa después?

- Backend

Modalidad de trabajo

Nuestra propuesta educativa está diseñada especialmente para esta modalidad 100% a distancia, mediante un aprendizaje activo y colaborativo siguiendo nuestro pilar de "aprender haciendo".
Los entornos de aprendizaje son tanto sincrónicos como asincrónicos, con un enfoque que vincula teoría y práctica, por lo que ambas están presentes en todo momento.
Contamos con un campus virtual propio en el cual vamos a encontrar actividades, videos, presentaciones y recursos interactivos con instancias de trabajo individual y en equipo para profundizar en cada uno de los conceptos.

Además, realizaremos encuentros online y en vivo con el grupo de estudiantes y docentes, a los que podremos sumarnos desde donde estemos a través de una plataforma de videoconferencias con nuestra cámara y micrófono para generar una experiencia cercana.

Metodología de evaluación

La evaluación formativa es un proceso continuo que genera información sobre la formación de nuestros estudiantes y de nosotros como educadores.
A su vez, se genera conocimiento de carácter retroalimentador, es decir, tiene una función de conocimiento ya que nos permite conocer acerca de los procesos de enseñanza y aprendizaje
También tiene una función de mejora continua porque nos permite saber en qué parte del proceso nos encontramos, validar si continuamos por el camino planificado o necesitamos tomar nuevas decisiones para cumplir los objetivos propuestos.
Por último, la evaluación desempeña un papel importante en términos de promover el desarrollo de competencias muy valiosas.
Nuestro objetivo es correrno s de la evaluación tradicional, donde muchas veces resulta un momento difícil, aburrido y tenso. Para ello, vamos a utilizar la gamificación, la cual es una técnica donde se aplican elementos de juego para que el contenido sea más atractivo, los participantes se sientan motivados e inmersos en el proceso, utilicen los contenidos de aprendizaje como retos que realmente quieren superar y aprendan del error.
A su vez, para registrar dicha formación, se utiliza un conjunto de instrumentos, para los cuales es fundamental utilizar la mayor variedad posible y técnicas de análisis.

Criterios de aprobación

- Realizar las actividades de Playground (80% de completitud)
- Asistencia a los encuentros sincrónicos (90% de asistencia)*
- Obtener un puntaje de 7 o más en la evaluación final.
- Obtener un puntaje de 7 o más en la nota final de la materia.

Contenidos

<u>Módulo 1:</u> Introducción a la programación orientada a objetos

Introducir el paradigma de la programación orientada a objetos.

- Establecer las similitudes y diferencias con un lenguaje de programación estructurado. Fundamentar sus ventajas.
- Explicar la necesidad de abstracción, desarrollar el concepto de encapsulamiento.
- Nombrar los conceptos de herencia y polimorfismo.
- Explicar la necesidad de tipos de datos para la definición de variables.
- Diferenciar entre una clase y un objeto.
- Explicar el concepto de clase, atributos, métodos. Constructor, métodos de acceso, métodos de propósito general.
- Demostrar el uso de UML para diagramas de clase.

Clase 1: ¿Qué es Java?

- Bienvenida de la materia
- Qué es JAVA
- Comparación con Javascript
- Entorno de desarrollo (IDE)
- Declaración de variables
- Tipos de datos propios de Java (primitivos)
- Operaciones con variables
- Método main
- Mostrar valor de una variable (System.out.println)
- Estructuras de control (if, switch, for, while)

Clase 2: Introducción a Java

- String, Integer, Float, Date
- Ingreso de datos con scanner
- Estructura de Datos Estatica (Array)
- Funciones

Clase 3: Cierre de la semana

Clase 4: Objetos y UML

- Conceptos de objeto y clases (POO)
- Atributo/comportamiento (responsabilidades)
- Encapsulamiento (público/privado)
- Diagrama UML
- Clases
- Variables y métodos de Clase

Clase 5: Clases

- Convenciones de nomenclatura (Camel case)
- Implementación de clases en Java
- Atributos y métodos
- Constructor
- Ocultar las variables: setters y getters
- Uso del this
- Creación de instancias

Clase 6: Cierre de la semana

Módulo 2: Programación Orientada a Objetos en Java

Diferenciar tipos de relaciones entre clases. Explicar variables y métodos de instancia y de clase, usos posibles. Trabajar los conceptos de herencia y polimorfismo. Utilizar clases abstractas e Interfaces. Colecciones, explicar qué son y cuales están disponibles en Java. Mostrar como realizar sobrecarga y sobreescritura. Trabajar con sobrecarga de equals y toString. Utilizar excepciones.

Clase 7: Relaciones entre clases

- Relaciones entre clases
- Representación en UML de relaciones
- Agregación
- Composición
- Implementación en Java

Clase 8: Herencia en UML

- Relaciones entre clases
- Herencia
- Alcance, protected
- Firma de un método
- Sobrecarga y sobreescritura

Clase 9: Cierre de la semana

Clase 10: Herencia en Java

- Herencia en Java
- Super
- Sobrecarga y sobreescritura en Java
- toString
- Instanceof , getClass. Casting
- equals

Clase 11: Clases abstractas

- Concepto de clase abstracta
- Uso de clases abstractas en el diseño
- Métodos abstractos
- Polimorfismo
- Redefinición de método
- Diferencia con clases concretas

Clase 12: Cierre de la semana

Clase 13: Interface

- Interface
- Concepto de interface (diferencia con herencia)
- Implements (uso de interfaces)
- Interface comparable

Clase 14: Práctica pre- evaluación

Clase 15: Evaluación

Clase 16: Colecciones

- Colecciones
- Diferentes tipos de colecciones
- Iterator
- Métodos generales, búsqueda, recorrer una colección
- Tipos en la definición de colecciones (genéricos)
- Array, diferencia con colecciones
- ArrayList, Set, Hash
- Streams API

Clase 17: Manejo de excepciones

- Concepto de excepción y de error
- Manejo del try/catch
- Throw y throws - Propagación de excepciones
- Excepciones de usuario
- Uso del finally

Clase 18: Cierre de la semana

Módulo 3: Patrones de diseño

Presentar patrones de diseño para lograr encontrar soluciones a problemas tradicionales.

Clase 19: Introducción a patrones de diseño

- Patrones
- Concepto de patrones de diseño
- Singleton
- Factory

Clase 20: Patrón State

- State
- Presentación en UML
- Ejemplo
- Implementación típica en Java

Clase 21: Cierre de la semana

Clase 22: Patrón Composite

- Composite
- Presentación en UML
- Ejemplo
- Implementación típica en Java

Clase 23: Práctica Pre- Evaluación

Clase 24: Evaluación

Clase 25: Patrón Observer

- Observer
- Presentación en UML
- Ejemplo
- Implementación típica en Java

Clase 26: Patrón Strategy

Clase 27: Cierre

Material de referencia

Deitel, P. y Deitel H. (2016). Como Programar en Java (10ª ed.). México: Pearson Educación.

Barker, J. (2005). Beginning Java Objects: from Concepts to Code (2ª ed.). Berkeley: Apress.

Fowler, M. y Scott, K. (1999). UML: Gota a gota. México: Pearson Educación.

Fontenla, C. M. y Paéz, N. (2011). Orientación a objetos con Java y UML (2ª ed. rev. y aum.). Buenos Aires: Nueva Librería.

Gamma, E., Helm, R., Johnson R. y Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Estados Unidos: Addison-Wesley Professional.