1. Datos Generales de la asignatura

Nombre de la asignatura:	Programación Orientada a Objetos
Clave de la asignatura:	CDD-2421
SATCA[1]:	2-3-5
Carrera:	Ingeniería en Ciencia de Datos

2. Presentación

Caracterización de la asignatura

La asignatura de Programación Orientada a Objetos (POO) contribuye significativamente al perfil de egreso al proporcionar a los estudiantes las habilidades necesarias para diseñar, desarrollar y mantener sistemas de software robustos y escalables utilizando los principios y conceptos fundamentales de POO. Esto permite a los egresados ser capaces de abordar problemas complejos de manera estructurada, utilizando un enfoque modular y orientado a objetos.

La asignatura de Programación Orientada a Objetos es crucial en la formación de estudiantes en ingeniería en Ciencia de Datos, ya que les brinda una base sólida en los principios de diseño de software moderno. POO promueve la reutilización de código, la modularidad, la extensibilidad y la facilidad de mantenimiento, lo que resulta fundamental en el desarrollo de aplicaciones complejas y de alta calidad.

La asignatura de Programación Orientada a Objetos se centra en los conceptos fundamentales de POO, incluyendo clases, objetos, herencia, polimorfismo, encapsulamiento, abstracción, y otros principios relacionados. Los estudiantes aprenden a diseñar y desarrollar sistemas de software utilizando lenguajes de programación orientados a objetos.

La asignatura de Programación Orientada a Objetos se relaciona estrechamente con otras asignaturas dentro del plan de estudios, como Estructuras de Datos y Algoritmos, Bases de Datos, Ingeniería de Software, entre otras. En estas asignaturas, se profundiza en temas específicos relacionados con POO, como el diseño de algoritmos eficientes utilizando estructuras de datos orientadas a objetos, la implementación de bases de datos relacionales utilizando modelos de objetos, y la aplicación de metodologías de desarrollo de software que hacen uso intensivo de los principios de POO.
Las competencias específicas que se desarrollan en la asignatura de Programación Orientada a Objetos incluyen la capacidad para diseñar e implementar programas modulares y reutilizables, la habilidad para entender y aplicar los principios de diseño de objetos, la destreza para utilizar herramientas y tecnologías relacionadas con POO, y la aptitud para colaborar en equipos de

[1] Sistema de Asignación y Transferencia de Créditos Académicos

desarrollo de software para crear soluciones efectivas a problemas complejos. Estas competencias son esenciales para el éxito profesional en el campo de la ingeniería en Ciencia de Datos.

Intención didáctica

Los contenidos de la asignatura de Programación Orientada a Objetos deben ser abordados de manera teórica y práctica. Esto implica una combinación de clases expositivas donde se presentan los conceptos fundamentales de POO, junto con sesiones de laboratorio donde los estudiantes tengan la oportunidad de aplicar esos conceptos en la práctica a través de ejercicios y proyectos.

El enfoque debe ser principalmente práctico, fomentando la resolución de problemas y la implementación de soluciones utilizando los principios de POO. Se deben proporcionar ejemplos concretos y casos de estudio que permitan a los estudiantes comprender cómo aplicar los conceptos teóricos en situaciones reales.

Los contenidos deben cubrir los principios básicos y avanzados de POO, desde la definición de clases y objetos hasta temas más avanzados como herencia, polimorfismo, encapsulamiento, diseño de patrones, etc. La profundidad de los temas debe ser suficiente para que los estudiantes comprendan los conceptos y puedan aplicarlos de manera efectiva en la resolución de problemas.

Se deben resaltar actividades como la resolución de problemas, la colaboración en equipo, la comunicación efectiva, el pensamiento crítico y la creatividad. Estas actividades ayudan a desarrollar competencias genéricas como el trabajo en equipo, la resolución de problemas, la comunicación oral y escrita, y el pensamiento analítico.

Al tratar los contenidos de la asignatura de Programación Orientada a Objetos, se están desarrollando competencias genéricas como la capacidad para trabajar en equipo, la habilidad para resolver problemas de manera efectiva, la comunicación clara y precisa, el pensamiento crítico y analítico, y la capacidad para adaptarse a nuevas tecnologías y entornos.

El docente debe desempeñar un papel activo y facilitador en el desarrollo de la asignatura. Esto implica proporcionar orientación y apoyo a los estudiantes, diseñar actividades prácticas que fomenten el aprendizaje activo, evaluar el progreso de los estudiantes de manera continua y proporcionar retroalimentación constructiva, y estar al tanto de las tendencias y avances en el campo de la programación orientada a objetos para mantener actualizado el contenido del curso. Además, el docente debe fomentar un ambiente de aprendizaje colaborativo y motivador que inspire a los estudiantes a explorar y profundizar en los temas de la asignatura.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Instituto Tecnológico Superior de Alvarado del 21 al 23 agosto de 2023.	Representante del Instituto Tecnológico Superior de Alvarado.	Propuesta inicial.
Tecnológico Nacional de México 30 octubre 2023	Representante del Instituto Tecnológico de: Querétaro y del Instituto Tecnológico Superior de Alvarado.	Presentación de la propuesta de la carrera de Ingeniería en Ciencia de Datos.
Instituto Tecnológico de Querétaro Campus Norte del 19 al 22 de marzo 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Puebla, Querétaro, Tehuacán. Instituto Tecnológico Superior de Alvarado. CENIDET. Representante de Ciencias Básica de los Institutos de: Celaya, Morelia y CIIDET.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Ciencia de Datos.
Tecnológico Nacional de México del 22 al 24 de abril del 2024	Representante del Instituto Tecnológico de Querétaro e Instituto Tecnológico Superior de Alvarado.	Contraste y ajuste de las asignaturas de Ingeniería en Ciencia de Datos con respecto a las de Ing. en Inteligencia Artificial, Ing. en Desarrollo WEB e Ing. en Ciberseguridad
Tecnológico Nacional de México del 27 al 31 de mayo del 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Querétaro. Instituto Tecnológico Superior de Alvarado. CENIDET.	Consolidación curricular de la carrera de Ingeniería Ciencia de Datos

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplica la programación orientada a objetos para resolver problemas reales y de ingeniería en Ciencia de Datos.

5. Competencias previas

Aplica algoritmos y lenguajes de programación para diseñar e implementar soluciones a problemáticas del entorno.

6. Temario

No.	Temas	Subtemas
1	Introducción al paradigma Orientado a Objetos.	1.1. Definición del paradigma Orientado a Objetos. 1.2. Programación Orientada a Objetos Vs. Programación Estructurada. 1.3. Clases y Objetos. 1.4. Pilares de la programación Orientada a Objetos: Abstracción, Encapsulamiento, Herencia y Polimorfismo. 1.5. Lenguaje de modelado unificado: diagrama de clases. 1.6. Introducción a los principios SOLID. 1.6.1. Principio de Única Responsabilidad (S). 1.6.2. Principio de Abierto Cerrado (O). 1.6.3. Principio de Sustitución de Liskov (L). 1.6.4. Principio de Segregación de Interfaz (I). 1.6.5. Principio de Inversión de Dependencias (D).A.
2	Clases y objetos.	2.1. Declaración de clases: modificadores de acceso (public, private, protected), atributos y métodos. 2.2. Instanciación de una clase. 2.3. Referencia al objeto actual. 2.4. Métodos: declaración, mensajes, paso de parámetros, retorno de valores. 2.5. Constructores y destructores declaración, uso y aplicaciones. 2.6. Sobrecarga de métodos. 2.7. Sobrecarga de operadores: Concepto y utilidad, operadores unarios y binarios.
3	Herencia.	3.1. Diferenciación entre clases abstractas, clases concretas e Interfaces. 3.2. Clasificación de Herencia: herencia simple y herencia múltiple. 3.3. Reutilización de miembros heredados. 3.4. Referencia al objeto de la clase base. 3.5. Constructores y destructores en clases derivadas. 3.6. Redefinición de métodos en clases derivadas.

4	Polimorfismo.	4.1. Definición de Polimorfismo. 4.2. Diferenciación entre Polimorfismo a nivel de objeto y a nivel de clase. 4.3. Clases abstractas: definición, métodos abstractos, implementación de clases abstractas, modelado de clases abstractas 4.4. Interfaces: definición, implementación de interfaces, herencia de interfaces 4.5. Variables polimórficas (plantillas): definición, uso y aplicaciones 4.6. Reutilización de código
5	Patrones de diseño.	5.1. Definición 5.2. Tipos de patrones de diseño: creacionales, estructurales y de comportamiento. 5.3. Ejemplos de patrones de diseño creacionales. 5.4. Ejemplos de patrones de diseño estructurales. 5.5. Ejemplos de patrones de diseño de comportamiento.
6	Flujos y archivos	6.1. Definición 6.2. Clasificación: Archivos de texto y binarios 6.3. Operaciones básicas y tipos de acceso 6.4. Manejo de objetos persistentes 6.5. Manejo de archivos JSON.

7. Actividades de aprendizaje de los temas

1. Introducción al paradigma de la programación orientada a objetos	
Competencias	Actividades de aprendizaje
Específica(s): Aplica los conceptos del paradigma de programación orientada a objetos para modelar situaciones de la vida real. *Genéricas:* ● Capacidad de análisis y síntesis. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	● Investigar en diversas fuentes los conceptos principales del paradigma orientado a objetos para elaborar un resumen. ● Identificar ejemplos de la vida real donde se manifiesten dichos conceptos y comentarlos en clase. ● Redactar una definición propia de los conceptos de forma simple y entendible. ● Diseñar un diagrama de clases aplicados a distintos problemas utilizando un software adecuado e identificar las herramientas de representación utilizadas; elaborar reporte.

2. Clases y objetos	
Competencias	Actividades de aprendizaje
Específica(s): Aplica los conceptos de clases y objetos en el desarrollo de programas para solución de problemas de acuerdo con el paradigma orientado a objetos. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	● Realizar actividades grupales para que el alumno identifique mediante la abstracción las características y comportamientos de objetos de su entorno. ● Diseñar diagramas de clases relacionados a objetos de su entorno considerando únicamente la identificación de los atributos del objeto e implementar las clases en un lenguaje de programación orientado a objetos. Considerar modificadores de acceso públicos para exponer y comprender la vulnerabilidad de los datos. ● Diseñar diagramas de clases protegiendo los datos con modificadores de acceso privado o protegido y agregar métodos públicos para obtener acceso seguro a los mismos. ● Crear clases que reúnan los miembros necesarios para resolver un problema y así implementar el encapsulamiento, utilizando un lenguaje de programación orientado a objetos. ● Identificar el tiempo de vida de las variables al instanciar un objeto. ● Identificar la estructura de un método y crear una aplicación que permita su uso en la resolución de problemas específicos. ● Crear aplicaciones que contengan métodos sobrecargados y probar la utilidad de dichos métodos, utilizando un lenguaje de programación orientado a objetos. ● Elaborar reporte de prácticas.
3. Herencia	
Competencias	Actividades de aprendizaje
Específica(s): Aplica relaciones de herencia en clases derivadas para reutilizar los miembros de una clase base en el desarrollo de aplicaciones. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita.	● Elaborar un cuadro sinóptico en el que se muestre las definiciones de herencia y su clasificación. ● Identificar los atributos y comportamientos propios de una especie que comparten los animales pertenecientes a ella. ● Identificar los atributos y comportamientos propios de una categoría de objetos que compartan todos sus miembros. ● Crear aplicaciones que manejen el concepto de herencia implementando redefinición de constructores y métodos, utilizando un lenguaje de programación orientado a objetos. ● Elaborar un reporte de prácticas.

● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	
4. Polimorfismo	
Competencias	Actividades de aprendizaje
Específica(s): Aplica el concepto de polimorfismo para la definición de clases abstractas e interfaces que permitan reutilización de código. Genéricas: ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	● Realizar un cuadro comparativo entre clases abstractas, clases concretas e interfaces. ● Identificar clases base que no requieren ser instanciadas o que carezcan de sentido para ello por ser abstractas y discutirlo en clase. ● Investigar en fuentes de información los conceptos y reglas para implementar clases abstractas en un programa y hacer un resumen. ● Crear un programa donde se maneje herencia de interfaces para especializar los comportamientos que las clases podrán implementar. ● Crear un programa donde se declare variables miembro de tipo clase abstracta o interfaz para que en tiempo de ejecución se inicialice con diferentes subtipos o implementaciones de las mismas, y se demuestre así, toda la flexibilidad del polimorfismo al cambiar el comportamiento de un objeto en tiempo de ejecución. ● Elaborar reporte de prácticas.
5. Patrones de diseño	
Competencias	Actividades de aprendizaje
Específica(s): Aplica los patrones de diseño creacionales, estructurales y de comportamiento para la reutilización del código y resolución de problemas. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas).	● Realizar un cuadro comparativo entre los diferentes tipos de patrones de diseño. ● Investigar ejemplos de los 3 tipos de patrones de diseño que son: creacionales, estructurales y de comportamiento. ● Proponer una solución a un problema real utilizando un patrón de diseño. ● Proponer una solución a un problema real utilizando un patrón de diseño estructural.

● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	● Proponer una solución a un problema real utilizando un patrón de diseño de comportamiento. ● Elaborar reporte de prácticas.
6. Flujos y archivos	
Competencias	Actividades de aprendizaje
Específica(s): Aplica la clasificación de archivos y operaciones básicas sobre éstos para manipular su información. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación. ● Habilidad para manejo de equipo de cómputo. ● Capacidad para trabajar en equipo. ● Solución de problemas.	● Investigar en fuentes de información los conceptos y metodologías para manipular archivos de texto y binarios y hacer un resumen. ● Crear un programa que maneje un archivo de texto y sus operaciones básicas. ● Crear un programa que maneje un archivo binario y sus operaciones básicas. ● Crear un programa que maneje un archivo con estructura JSON. ● Elaborar el reporte de prácticas.

8. Práctica(s)

- Crear un programa que instancie y use un objeto predefinido por el lenguaje para practicar el envío de mensajes, el uso de parámetros y la recepción de su respuesta. Sugerencia: objeto de clase String.
- Implementar clases para instanciar objetos que modelan sus contrapartes de la vida real usando tipos de datos simples y objetos como parámetros y valores de retorno, así como métodos sin valores de retorno.
- Intercambiar clases de objetos entre compañeros para usar sus miembros con valores o situaciones erróneas que evidencian la necesidad de protegerlos con modificadores de acceso. Modificar el código fuente aplicando los distintos niveles de acceso para experimentar y descubrir (aprender) el impacto de cada uno de ellos.
- Implementar la clase Persona con los atributos nombre y edad; un constructor, un destructor, y al menos el método para mapear el ciclo de vida de una persona con el de un objeto.
- Implementar la clase Calculadora que realice al menos las cuatro operaciones básicas de la aritmética sobrecargando métodos para cada tipo de dato numérico del lenguaje de los parámetros.
- Implementar la clase Matriz que sobrecargue los operadores +, -, * y / para este tipo de dato definido por el usuario.
- Programar una aplicación sobre figuras geométricas que implemente la clase base Figura Geométrica de la cual hereden sus miembros las clases derivadas y que éstas solo especialicen sus características o comportamientos.
- Implementar constructores y destructores a las clases base y derivadas de la aplicación sobre figuras geométricas para experimentar y comprender su funcionamiento cuando está implicada la herencia.
- Modificar la clase Figura Geométrica para convertirla en abstracta y programar al menos un método abstracto que todas las clases derivadas deberán implementar con su propio comportamiento.
- Programar la interfaz Vehículo con un conjunto de métodos abstractos que todo vehículo de la vida real debería tener. Programar varias clases que implementen la interfaz anterior y definan el comportamiento particular de sus métodos.
- Especializar la interfaz Vehículo en al menos dos subinterfaces (vehículo terrestre o vehículo aéreo) que agreguen comportamientos abstractos que las clases deberán implementar.
- Programar clases que generen excepciones comunes como referencias nulas o desbordamientos numéricos para estudiar su naturaleza, comportamiento, prevención y lanzamiento.
- Implementar aplicaciones que almacenen y recuperen información de diferentes tipos de datos simples a través de un archivo de texto para persistir información.
- Programar una clase que tome un objeto de cierto tipo y lo persista en un archivo de texto para ser recuperado posteriormente restableciendo el estado que tenía antes de ser persistido (serializarlo).

9. Proyecto de asignatura

El objetivo del proyecto que planteé el docente que imparta esta asignatura, es demostrar el desarrollo y alcance del(los) logro(s) formativo(s) de la asignatura, considerando las siguientes fases:

- **Fundamentación:** marco referencial (teórico, conceptual, contextual, legal) en el cual se fundamenta el proyecto de acuerdo con un diagnóstico realizado, mismo que permite a los estudiantes lograr la comprensión de la realidad o situación objeto de estudio para definir un proceso de intervención o hacer el diseño de un modelo.
- **Planeación:** con base en el diagnóstico en esta fase se realiza el diseño del proyecto por parte de los estudiantes con asesoría del docente; implica planificar un proceso: de intervención empresarial, social o comunitario, el diseño de un modelo, entre otros, según el tipo de proyecto, las actividades a realizar los recursos requeridos y el cronograma de trabajo.
- **Ejecución:** consiste en el desarrollo de la planeación del proyecto realizada por parte de los estudiantes con asesoría del docente, es decir en la intervención (social, empresarial), o construcción del modelo propuesto según el tipo de proyecto, es la fase de mayor duración que implica el desempeño de los saberes, habilidades y destrezas a desarrollar.
- **Evaluación:** es la fase final que aplica un juicio de valor en el contexto laboral-profesión, social e investigativo, ésta se debe realizar a través del reconocimiento de logros y aspectos a mejorar se estará promoviendo el concepto de “evaluación para la mejora continua”, el desarrollo del pensamiento crítico y reflexivo en los estudiantes.

10. Evaluación por competencias

La evaluación debe hacerse diagnóstica, formativa y sumativa. De igual manera, para fortalecer la parte actitudinal, se recomienda guiar al estudiante hacia la introspección para utilizar la autoevaluación y la coevaluación.

En el caso de las actividades de aprendizaje se sugiere el uso de estrategias metacognitivas como: mapas mentales, mapas conceptuales, reportes de prácticas, exposiciones en clase, ensayos, resúmenes, observación y cuestionarios, cuadros comparativos, informes.

Mientras que para verificar el nivel del logro de las competencias del estudiante se recomienda utilizar: el portafolio de evidencias, listas de cotejo, rúbricas, matrices de valoración, exámenes, guías de observación, además de estrategias en las que se logren las competencias blandas.

11. Fuentes de Información

1. Sierra, K., & Bates, B. (2018). Head First Java (2nd ed.). O'Reilly Media.
2. Deitel, P., & Deitel, H. (2019). Java: How to Program (11th ed.). Pearson.
3. Freeman, E., Robson, E., & Bates, B. (2020). Head First Design Patterns: A Brain-Friendly Guide (2nd ed.). O'Reilly Media.
4. Schildt, H. (2018). Java: The Complete Reference (11th ed.). McGraw-Hill Education.
5. Budd, T. (2002). Object Oriented Programming. Third edition: Addison Wesley.
6. Ceballos F. (2003). Programación Orientada a Objetos con C++. 3ra. Edición: Editorial Ra-Ma.
7. Ceballos J. (2012) Microsoft C# -Curso de Programación. España: Alfaomega.
8. Clark D., (2013). Beginning C# Object-Oriented Programming, Apress.
9. Craig I. (2002). The Interpretation of Object-Oriented Programming Languages. 2nd Edition: Springer. London.
10. Dean J. y Dean R. (2009) Introducción a la programación con Java.: McGraw Hill.
11. Deitel P., Deitel H. (2013). Cómo programar en java. 9a. Edición. Pearson.
12. Leonardo Bermón Angarita. (2021). Ejercicios de programación orientada a objetos con Java y UML. 1ra. Edición. Universidad Nacional de Colombia.
13. Doyle, B (2013) C# Programming: From Problem Analysis to Program Design. Cengage Learning.
14. Groussard, T. (2009) Visual Basic.NET (VB.NET) - Programe con Visual Studio 2008. España: Eni Ediciones.
15. Groussard, T. (2011) Recursos Informáticos C#4 Los fundamentos del lenguaje- desarrollar con visual studio 2010. España: Eni Ediciones.
16. Gutiérrez F., Duran F., Pimentel E. (2007). Programación Orientada a Objetos con JAVA: Ediciones Paraninfo, S.A.
17. Harvey M. (2008). Como programa en Java. México: Prentice Hall.
18. Holmes B., Joyce D. (2001). Object-Oriented Programming with JAVA. Jones and Barlett: Canada. Publishers Inc.
19. Joyanes L. (2011). Fundamentos de programación: Algoritmos, Estructuras de Datos y Objetos. 3ra. Edición. Mac-Graw Hill.
20. Joyanes, L. y Zahonero, I. (2011) Programación en Java 6. España: McGraw Hill.
21. Larman G. (2002). UML y Patrones 2/E: Pearson Educacion.
22. Marrer G. (2009). Fundamentals of Programming with Object Oriented Programming. Python Edition. Ebook Edition.Muñoz C., Nino A. Vizcaino A. (2002). Introducción a la Programación con Orientación a Objetos: Pearson Educacion.
23. Schildth H. (2002). Fundamentos de programación en Java: Mac-Graw Hill 20 Solana Aroa (2009). Programación con C# 4.0. Madrid, España.
24. Velez Serrano, J.F., Peña Abril, A., Gortázar Bellas, F. Sánchez Calle, A. (2010) Diseñar y Programar, Todo Es Empezar: Una Introducción a la Programación Orientada a Objetos Usando UML Y JAVA (EBOOK).
25. VV.AA. (2004). C/C++ Y Java. Cómo programar (4ª Ed). México.
26. VV.AA., (2006) Introducción a la Programación Orientada a Objetos: Universidad de Alicate. Servicio de Publicaciones.
27. Warnes D., Kolling M.(2007). Programación Orientada a Objetos con Java. 3ra Edición: Prentice- Hall.
28. BlueJ, (2014). A free Java Development Environment designed for beginners. Disponible en internet en www.bluej.org. Fecha de acceso: 13 de Febrero de 2014.

29. Greenfoot, (2014). Teach and learn java programming. Disponible en internet en www.greenfoot.org. Fecha de acceso: 13 de Febrero de 2014.
30. Jeroo, (2014). Jeroo. Disponible en internet en http://home.cc.gatech.edu/dorn/jeroo. Fecha de acceso: 13 de Febrero de 2014.
31. Oracle (2014). Java platform – Standard edition 7 API specification. Disponible en internet en http://docs.oracle.com/javase/7/docs/api/. Fecha de acceso: 13 de Febrero de 2014.
32. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.