

|  | | * Herramienta fundamental para Redes * Herramientas para Ciencias de la computación |
| --- | --- | --- |
| Requisitos previos: | | Programación Básica |
| **II. PROGRAMACION DEL CONTENIDO** | | |
| **OBJETIVO GENERAL** | | |
| Presentar al estudiante la conceptualización y aplicación del paradigma orientado a objetos, enfatizando e los elementos conceptuales propios de este que permitan plantear y aplicar modelos bien formados utilizand un lenguaje de programación orientado a objetos. | | |
| **OBJETIVOS ESPECÍFICOS** | | |
| 1. Determinar los tipos de aplicación y las situaciones en las que se debe aplicar el paradigma orientado a objetos. 2. Comprender, interpretar y analizar el cambio de enfoque en el modo de resolver problemas que supone el uso del paradigma orientado a objetos respecto a otros paradigmas. 3. Aplicar los conceptos del paradigma de programación orientada a objetos tales como: polimorfismo,encapsulamiento, herencia, sobrecarga, funciones virtuales, etc., usando como lenguaje de programación C# o Java 4. Manejar adecuadamente conceptos tales como: recursividad, objetos transientes, residentes y persistentes; generalización y generacidad; clases plantillas; asociación, agregación y composición. 5. Identificar problemas de: portabilidad, efectos colaterales y transparencia referencial. 6. Comprender la enorme importancia de crear software fiable, reutilizable y mantenible. 7. Dominar estrategias básicas de reutilización como son el uso de librerías o paquetes de software. 8. Aplicar el modelo orientado a objetos en programación de dispositivos de cómputo. | | |
| **COMPETENCIAS DE FORMACIÓN:** | | |
| Competencias que compromete la asignatura: | El estudiante está en capacidad de pensar ordenadamente para modelar una solución a un problema, en donde se debe analizar e implementar dicha solución aplicando el paradigma de programación orientado a objetos | |
| Competencias específicas de la asignatura: | * El estudiante entiende el concepto de paradigma y sus implicaciones en el modo de resolver problemas. * Conoce y entiende el proceso de evolución de los distintos paradigmas de programación. * Entiende el tipo de problemas de desarrollo software que solucionan un uso correcto del paradigma orientado a objetos. * Conoce el modo en que el paradigma orientado a objetos ayuda a mejorar las capacidades de reutilización del software. * Entiende los conceptos de clase, atributo, operación, interfaz y objeto. * Entiende el mecanismo de paso de mensajes. * Comprende el modo en que se deben implementar los caminos de comunicación entre clases para permitir el paso de mensajes entre ellas. * Entiende y es capaz de implementar los distintos tipos de relaciones que se pueden establecer a nivel de objeto entre dos clases: asociaciones, agregaciones y composiciones. * Entiende el concepto de estado de un objeto. * Entiende la relación entre diagramas de clase y el código de implementación de dichos diagramas. * Entiende el mecanismo de abstracción de la herencia. * Es capaz de plantear jerarquías de herencia bien definidas. * Comprende los costes de la herencia. * Diferencia claramente cuándo usar herencia y cuándo optar por composición.  Entiende el concepto y la utilidad del polimorfismo. * Entiende la diferencia entre ligadura estática y ligadura dinámica en los | |

n o

|  | lenguajes de programación.   * Entiende la relación a nivel de implementación entre herencia y polimorfismo. * Identifica los distintos tipos de polimorfismo: sobrecarga, sobreescritura, variables polimórficas y genericidad. * Entiende las relaciones entre los distintos tipos de polimorfismo. * Entiende los mecanismos de gestión de errores que ofrecen algunos lenguajes de programación (C# o Java). * Entiende el concepto de concurrencia. * Entiende el concepto de persistencia. |
| --- | --- |
| Competencias Transversales a las que contribuye la asignatura: | * El alumno tiene la capacidad de discernir que tecnología debe utilizar para la resolución de problemas particulares. * Comunica ideas de manera clara de forma oral o escrita. * Actúa estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos. |
| Programa sintético: | 1. **Introducción al paradigma Orientado a Objetos**    1. El progreso de la abstracción    2. El paradigma orientado a objetos    3. Lenguajes orientados a objetos    4. Metas del paradigma orientado a objetos 2. **Fundamentos de la programación orientada a objetos**    1. Clases    2. Atributos    3. Operaciones (métodos)    4. Encapsulación y ocultamiento de la información.    5. Modularidad de Meyer.    6. El concepto de interfaz    7. El concepto de objeto    8. Metaclases    9. El diseño de aplicaciones OO    10. Relaciones entre clases y relaciones entre objetos    11. Documentación del código 3. **Herencia y polimorfismo**    1. Introducción a la Herencia    2. Herencia Simple    3. Herencia Múltiple    4. Herencia de Interfaz    5. Herencia de Implementación    6. Beneficios y costes de la herencia    7. Elección de la técnica de reutilización    8. Polimorfismo y reutilización    9. Sobrecarga    10. Polimorfismo en jerarquías de herencia    11. Variables Polimórficas    12. Genericidad 4. **Gestión de errores y otras características**    1. Gestión de errores    2. Concurrencia    3. Persistencia       1. Persistencia con serialización       2. Persistencia con archivos    4. Recogiendo la basura 5. **Sockets** |

| **RESULTADOS DE APRENDIZAJE** | |
| --- | --- |
| * Entender el concepto de paradigma y sus implicaciones en el modo de resolver problemas. * Entender el proceso de evolución de los distintos paradigmas de programación. * Conocer el modo en que el paradigma orientado a objetos ayuda a mejorar las capacidades de reutilización del software. * Entender los conceptos de clase, atributo, operación, interfaz y objeto. * Entender el mecanismo de paso de mensajes. * Comprende el modo en que se deben implementar los caminos de comunicación entre clases para permitir el paso de mensajes entre ellas. * Entender y es capaz de implementar los distintos tipos de relaciones que se pueden establecer a nivel de objeto entre dos clases: asociaciones, agregaciones y composiciones. * Entender el concepto de estado de un objeto. * Entender la relación entre diagramas de clase y el código de implementación de dichos diagramas. * Entender el mecanismo de abstracción de la herencia. * plantear jerarquías de herencia bien definidas. * Comprender los costes de la herencia. * Diferenciar claramente cuándo usar herencia y cuándo optar por composición. Entiender el concepto y la utilidad del polimorfismo. | |
| **III. ESTRATEGIAS** | |
| **Metodología Pedagógica y Didáctica:**   * Asistencia a clases expositivas y de discusión  Elaboración y lectura de paper (documentación). * Se debe procurar incentivar el trabajo de grupo más que el trabajo individual. (se recomienda trabajar en grupos de dos o tres estudiantes) * Implementación y prueba de prototipos (programas) en laboratorio de computación        |  | Horas | |  | Horas  profesor/semana | Horas Estudiante/semana | Total Horas Estudiante/semestre | Créditos | | --- | --- | --- | --- | --- | --- | --- | --- | | **Tipo de** **Curso** | TD | TC | TA | (TD + TC) | (TD + TC +TA) | X 16 semanas |  | | Asignatura | 4 | 2 | 3 | 6 | 9 | 144 | 3 |   ***Trabajo Presencial Directo (TD)*** : trabajo de aula con plenaria de todos los estudiantes.  ***Trabajo Mediado\_Cooperativo (TC)***: Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.  ***Trabajo Autónomo (TA):*** Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio, etc.) | |
| **IV. RECURSOS** | |
| **Medios y Ayudas:**   * Aula normal con pizarrón para sesiones de cátedra y para sesiones de discusión. * Disponibilidad para acceder a proyector multimedia. * Laboratorio de computación, para las sesiones de laboratorio. * IDE’s para desarrollar en java (Eclipse, Netbeans, …) * Página web para publicar material didáctico, guías de ejercicios, soluciones, tareas, etc.  Acceso al material bibliográfico recomendado. * Asignación de una persona que tenga las plenas competencias del curso (monitor) para asesorar a los estudiantes en dudas durante las sesiones del laboratorio de computación. | |
| **BIBLIOGRAFÍA** |  |
| TEXTOS GUÍA |
| * Bertrand Meyer. Construcción de Software Orientado a Objetos. Prentice Hall. * Bruce Eckel. Thinking Java. Pretince Hall * Guía de certificación de java. Sun Microsystem. * Francisco Javier Ceballos Sierra, Microsoft C#. Lenguaje y aplicaciones, 2ª edición  Harvey M. Deitel y Paul J. Deitel, C# Como Programar, segunda edición. * Alfredo Weitzenfeld, Ingenieria de Software orientada a Objetos con UM. Java e Internet. |
| TEXTOS COMPLEMENTARIOS |
| * Agustín Froufe Quintas. Java 2 Manual de usuario y tutorial. Alfaomega. * Francisco Javier Ceballos Sierra , Enciclopedia de Microsoft Visual C#, 3ª edición |
| REVISTAS |
|  |
| DIRECCIONES DE INTERNET |
| http://msdn.microsoft.com/es-es/library/kx37x362(VS.80).aspx http://msdn.microsoft.com/es-es/vcsharp/default.aspx http://www.mygnet.net/manuales/java//guia\_java.1691 |

| **V. ORGANIZACIÓN / TIEMPOS** |
| --- |
|  |

**Espacios, Tiempos, Agrupamientos:**

Se recomienda trabajar una unidad cada cuatro semanas, trabajar en pequeños grupos de estudiantes, utilizar Internet para comunicarse con los estudiantes para revisiones de avances y solución de preguntas (esto considerarlo entre las horas de trabajo cooperativo).

| **PROGRAMA SINTÉTICO SEMANAS ACADÉMICAS** | | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **1. Introducción al paradigma Orientado a**  **Objetos** | **x** | **x** | **x** | **x** |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.1. El progreso de la abstracción | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.2. El paradigma orientado a objetos | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.3. Lenguajes orientados a objetos |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.4. Metas del paradigma orientado a objetos |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2. Fundamentos de la programación orientada a**  **objetos** |  |  |  |  | **x** | **x** | **x** | **x** | **x** | **x** |  |  |  |  |  |  |
| 2.1. Clases |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2.2. Atributos |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2.3. Operaciones (métodos) |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2.4. Encapsulación y ocultamiento de la información. |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |
| 2.5. Modularidad de Meyer. |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |
| 2.6. El concepto de interfaz |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |
| 2.7. El concepto de objeto |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |
| 2.8. Metaclases |  |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |
| 2.9. El diseño de aplicaciones OO |  |  |  |  |  |  |  | x |  |  |  |  |  |  |  |  |
| 2.10. Relaciones  entre clases y relaciones entre objetos |  |  |  |  |  |  |  |  | x |  |  |  |  |  |  |  |
| 2.11. Documentación del código |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |  |
| **3. Herencia y polimorfismo** |  |  |  |  |  |  |  |  |  |  | **x** | **x** | **x** | **x** | **x** | **x** |
| 3.1. Introducción a la Herencia |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| 3.2. Herencia Simple |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| 3.3. Herencia Múltiple |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| 3.4. Herencia de |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |

| Interfaz |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3.5. Herencia de Implementación |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| 3.6. Beneficios y costes de la herencia |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| 3.7. Elección de la técnica de reutilización |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |
| 3.8. Polimorfismo y reutilización |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |
| 3.9. Sobrecarga |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |
| 3.10. Polimorfismo en jerarquías de herencia |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |
| 3.11. Variables Polimórficas |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |
| 3.12. Genericidad |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |
| **4. Gestión de errores y otras características** |  |  |  |  |  |  |  |  |  |  |  |  |  | **x** | **x** | **x** |
| 4.1. Gestión de errores |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |
| 4.2. Concurrencia |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |
| 4.3. Persistencia |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |
| **5. Sockets** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |

| **VI. EVALUACIÓN** | | | |
| --- | --- | --- | --- |
| **PRIMERA NOTA** | **TIPO DE EVALUACIÓN** | **FECHA** | **PORCENTAJE** |
| Prueba oral/escrita para el grupo que el docente elabora.  Informes de conceptos basado en análisis (Paper's) Pruebas orales/escritas rápidas (Quizes) | Semana 4 ó 5 | 10 % |
| **SEGUNDA NOTA** | Prueba escrita conjunta (para todos los grupos de la asignatura), elaborada por los docentes que imparten la asignatura.  Pruebas orales/escritas rápidas (Quizes) | Semana 14 ó 15 | **20 %** |
| **TERCERA NOTA** | Guías de ejercicios resueltas  Informes de conceptos basado en análisis (Paper's)  Pruebas orales/escritas rápidas (Quizes) Prueba escrita | Varias fechas | 10 % |
| **CUARTA NOTA** | Informe de desempeño en laboratorio | Varias fechas | 20 % |
| **PROYECTO** | Informe de desempeño y sustentación de un prototipo funcional que evalúe las competencias exigidas. | Semana 16 | 10 % |
| **EXAMEN FINAL** | Prueba escrita conjunta (para todos los grupos de la asignatura), elaborada por los docentes que imparten la asignatura. |  | **30 %** |
| ASPECTOS A EVALUAR DEL CURSO | | | |
| * Claridad y entendimiento de los conceptos. * Que se haya identificado correctamente el problema y que el modelo lo represente adecuadamente. * Que la solución diseñada resuelva el problema. * Apego a la formalidad y estándares requeridos. * Que el análisis de corrección sea exhaustivo. * Que el prototipo corresponda al modelo diseñado y no presente errores de sintaxis. * La asistencia a las clases magistrales y a los laboratorios. * El esfuerzo y dedicación en la resolución de problemas. * Que la documentación permita reconocer la forma en que se ha abordado el problema y la estructura del programa implementado. * En las pruebas escritas se consideran en forma parcial los aspectos considerados en proyectos de programación bajo problemas que requieren un menor tiempo de desarrollo y en una modalidad que no requiere uso del computador, así como la comprensión conceptual. | | | |