UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE AREQUIPA

VICE RECTORADO ACADEMICO

SILABO

CODIGO DEL CURSO: 1002116

1.1 DATOS GENERALES

FACULTAD: Ing. De Producción y Servicios					
DEPARTAMENTO: Ingeniería de Sistemas e Informática		**ESCUELA:** Ingeniería de Sistemas			
ASIGNATURA: Objetos y Abstracción de Datos					
PRE REQUISITO : 1001208 - Introducción a la Programación Orientada a Objetos					
CREDITOS: 4	**Año: 2012-I**	**HORAS 6 (seis)**			
	Semestre: III	**T: 2**	**TP: 2**	**P: 2**	**S:**

PROFESOR: Ernesto Cuadros-Vargas

TITULO: **GRADO ACADEMICO: Doctor**

HORARIO Total Semanal: 14 Hrs.	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
	13:00-15:00	13:00-15:00	13:00-15:00, 18:00-20:00			
AULA:	101	101	Lab			

1.2 EXPOSICION DE MOTIVOS

Este es el tercer curso en la secuencia de los cursos introductorios a la informática. En este curso se pretende cubriros conceptos señalados por la *Computing Curricula* IEEE(c)-ACM 2001, bajo el enfoque *functional-first*.
El paradigma orientado a objetos nos permite combatir la complejidad haciendo modelos a partir de abstracciones de los elementos del problema y utilizando técnicas como encapsulamiento, modularidad, polimorfismo y herencia.
El dominio de estos temas permitirá que los participantes puedan dar soluciones computacionales a problemas de diseño sencillos del mundo real.

1.3 OBJETIVOS:

Introducir al alumno a los fundamentos del paradigma de orientación a objetos, permitiendo asimilar los conceptos necesarios para desarrollar un sistema de información.

1.4 CONTENIDO TEMATICO:

Capítulo 1: DS/Gráfos y Árboles. (7 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Ilustrar con ejemplos la terminología básica de teoría de grafos y algunas de las propiedades y casos especiales de cada una. • Mostrar diferentes métodos de recorrido en árboles y grafos. • Modelar problemas en Ciencias de la Computación usando grafos y árboles. • Relacionar grafos y árboles con estructura de datos, algoritmos y conteo.	• Arboles. • Grafos no dirigidos. • Grafos dirigidos. • Arboles de expansión. • Estrategias de recorrido. [1]	7	9-17Abr	11%

Capítulo 2: PF/Construcciones fundamentales. (5 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Analizar y explicar el comportamiento de programas simples involucrando las estructuras de programación fundamental cubiertas por esta unidad. • Modificar y extender programas cortos que usan condicionales estándar, estructuras de control iterativas y funciones. • Diseñar, implementar, probar y depurar un programa que use cada una de las siguientes estructuras fundamentales de programación: cálculos básicos, entrada y salida simple, estructuras estándar condicionales e iterativas y definición de funciones. • Escoger la estructura apropiada condicional e iterativa para una estructura de programación dada.	• Sintaxis básica y semántica de un lenguaje de más alto nivel. • Variables, tipos, expresiones y asignaciones. • Entrada y salida simple. • Estructuras de control condicionales e iterativas. • Funciones y paso de parámetros. • Descomposición estructurada. [3]	5	17-26Abr	20%

• Aplicar técnicas de descomposición estructurada o funcional para dividir un programa en pequeñas partes. • Describir los mecanismos de paso de parámetros.				

Capítulo 3: PF/Algoritmos y Resolución de Problemas. (5 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Discutir la importancia de los algoritmos en el proceso de solución de problemas. • Identificar las propiedades necesarias de un buen algoritmo. • Crear algoritmos para resolver problemas simples. • Usar pseudocódigo o un lenguaje de programación para implementar, probar y depurar algoritmos para resolver problemas simples. • Describir estrategias útiles para depuración.	• Estrategias para la solución de problemas. • El rol de los algoritmos en el proceso de solución de problemas. • Estrategias de implementación para algoritmos. • Estrategias de depuración. • El Concepto y propiedades de algoritmos. [3]	5	30Abr-3May	28%

Capítulo 4: PF/Estructuras de Datos. (9 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Describir la representación de datos numéricos y de caracteres. • Entender como la precisión y el redondeo puede afectar los cálculos numéricos. • Discutir la representación y uso de tipos de datos primitivos y estructuras de datos incorporadas en el lenguaje. • Describir aplicaciones comunes para cada estructura de datos e la lista de temas. • Implementar estructuras de datos definidas por el usuario en un lenguaje de alto nivel.	• Representación de datos numéricos. • Rango, precisión y errores de redondeo. • Arreglos y registros. • Cadenas y procesamiento de cadenas. • Representación de caracteres. • Administración del almacenamiento en tiempo de ejecución. • Punteros y referencias. • Estructuras enlazadas. • Estrategias de implementación para pilas, colas y tablas *hash*. • Estrategias de implementación para grafos y árboles. • Estrategias para escoger la es-	9	7-15May	

• Comparar implementaciones alternativas de estructuras de datos considerando su desempeño. • Escribir programas que usan cada una de las siguientes estructuras de datos: arreglos, registros, cadenas, listas enlazadas, pilas, colas y tablas de *hash*. • Comparar y contrastar los costos y beneficios de las implementaciones dinámicas y estáticas de las estructuras de datos. • Escoger la estructura de datos apropiada para modelar un problema dado.	tructura de datos correcta. [3]			

Capítulo 5: PF/Recursividad. (2 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Describir el concepto de recursividad y dar ejemplos de su uso. • Identificar el caso base y el caso general de un problema definido recursivamente. • Comparar soluciones iterativas y recursivas para problemas elementales tal como factorial. • Describir la técnica dividir y conquistar. • Implementar, probar y depurar funciones y procedimientos recursivos simples. • Describir como la recursividad puede ser implementada usando una pila. • Discutir problemas para los cuales el *backtracking* es una solución apropiada. • Determinar cuándo una solución recursiva es apropiada para un problema.	• El concepto de recursividad. • Funciones matemáticas recursivas. • Funciones recursivas simples. • Estrategias de dividir y conquistar. • *Backtracking* recursivo. [3]	2	21May	46%

Capítulo 6: PF/Programación Orientada a Eventos. (2 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Explicar la diferencia entre programación orientada a eventos y programación por línea de comandos. • Diseñar, codificar, probar y depurar programas de manejo de eventos simples que respondan a eventos del usuario. • Desarrollar código que responda a las condiciones de excepción lanzadas durante la ejecución.	• Métodos para la manipulación de eventos. • Propagación de eventos. • Manejo de excepciones. [3]	2	22May	49%

Capítulo 7: AL/Análisis Básico de Algoritmos. (3 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Determinar la complejidad de tiempo y espacio de algoritmos simples.	• Análisis asintótico de límites en los casos promedio y superior. • Identificar las diferencias entre el comportamiento entre el mejor, mediano y peor caso.	3	28-29May	54%

Capítulo 8: AL/Algoritmos Fundamentales. (3 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Implementar los algoritmos cuadráticos más comunes y los algoritmos de ordenamiento *O*(*NlogN*). • Diseñar e implementar una función de (*hash*) apropiada para una aplicación. • Diseñar e implementar un algoritmo de resolución de colisiones para tablas de *hash*. • Discutir la eficiencia computacional de los principales algoritmos de ordenamiento, búsqueda y (*hashing*). • Discutir otros factores, además de la eficiencia computacional, que influyen en la	• Algoritmos numéricos simples. • Búsqueda secuencial y binaria. • Algoritmos cuadráticos de ordenamiento (Selección, inserción). • Algoritmos de tipo *O*(*NlogN*) (Quicksort, heapsort, mergesort). • Tabla de (*hash*) incluyendo estrategias de solución para las colisiones. • Arboles de búsqueda binaria. • Representación de grafos (Listas y Matrices de adyacencia). • Recorridos por amplitud y profundidad. • El algoritmo del camino más corto (algoritmos de Dijkstra y Floyd). • Cerradura transitiva (algoritmo de Floyd). • Árbol de expansión mínima	3	29May-4Jun	59%

elección de los algoritmos, tales como tiempo de programación, mantenimiento y el uso de patrones específicos de aplicación en los datos de entrada. • Resolver problemas usando los algoritmos de grafos fundamentales, incluyendo búsqueda por amplitud y profundidad; caminos más cortos con uno y múltiples orígenes, cerradura transitiva, ordenamiento topológico y al menos un algoritmo de árbol de expansión mínima. • Demostrar las siguientes capacidades: evaluar algoritmos, seleccionar una opción de un rango posible, proveer una justificación para tal elección e implementar el algoritmo.	(algoritmos de Kruskal y Prim). • Ordenamiento Topológico. [3]			

Capítulo 9: PL/Máquinas Virtuales. (2 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Describir la importancia y poder de la abstracción en el contexto de máquinas virtuales. • Explicar los beneficios de los lenguajes intermedios en el proceso de compilación. • Evaluar las ventajas y desventajas entre desempeño vs. Portabilidad. • Explicar cómo los programas ejecutables pueden violar la seguridad de sistema computacional accediendo a archivos de disco y memoria.	• El concepto de máquina virtual. • Jerarquías de las máquinas virtuales. • Lenguajes intermedios. • Temas de seguridad relacionados a ejecutar código sobre una máquina externa. [2] [3]	2	5Jun	62%

Capítulo 10: PL/Declaración y Tipos (2 horas)

Objetivos Específicos:	Contenidos:	Hrs	Fecha	Avance %
• Explicar el valor de los	• La concepción de tipos como	2	11Jun	

modelos de declaración, especialmente con respecto a la programación en mayor escala. • Identificar y describir las propiedades de una variable, tales como su: dirección asociada, valor, ámbito, persistencia y tamaño. • Discutir la incompatibilidad de tipos. • Demostrar las diferentes formas de enlace, visibilidad, ámbito y manejo del tiempo de vida. • Defender la importancia de los tipos y el chequeo de tipos para brindar abstracción y seguridad. • Evaluar las ventajas y desventajas en el manejo del tiempo de vida (conteo por referencia vs. Recolección de basura).	un conjunto de valores unidos a un conjunto de operaciones. • Declaración de modelos (enlace, visibilidad, alcance y tiempo de vida). • Vista general del chequeo de tipos. • Vista general del chequeo de tipos. • Recolección de basura. [2]			74%%

Capítulo 11: PL/Programación Orientada a Objetos. (7 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Justificar la filosofía de diseño orientado a objetos y los conceptos de encapsulación, abstracción, herencia y polimorfismo. • Diseñar, implementar, probar y depurar programas simples en un lenguaje de programación orientado a objetos. • Describir como los mecanismos de clases soportan encapsulación y ocultamiento de la información. • Diseñar, implementar y probar la implementación de la relación esun *IsKindOf* entre objetos usando jerarquía de clases y herencia. • Comparar y contrastar	• Diseño orientado a objetos. • Encapsulación y ocultamiento de la información. • Separación de comportamiento e implementación. • Clases y subclases. • Herencia (sobreescritura, despacho dinámico). • Polimorfismo (polimorfismo de subtipo vs. herencia). • Jerarquías de clases. • Clases de tipo colección y protocolos de iteración. • Representaciones internas de objetos y tablas de métodos. [2] [3]	7	12-25Jun	85%

las nociones de sobrecarga y sobreescritura de métodos en un lenguaje de programación. • Explicar la relación entre la estructura estática de una clase y la estructura dinámica de las instancias de dicha clases. • Describir como los iteradores acceden a los elementos de un contenedor.				

Capitulo 12: SE/Diseño de Software.(5 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Discutir las propiedades del buen diseño de softwarem incluyendo la naturaleza y el rol de la documentación asociada. • Conducir una revisión de diseño de software con material de código abierto utilizando lineamientos apropiados.	• Conceptos y principios fundamentales de diseño. • El rol y uso de contratos. • Patrones de diseño [3]	5	26Jun-3Jul	93%

Capitulo 13: Programación con APIs (5 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Explicar el valor de las interfaces para programación de aplicaciones (APIs) en el desarrollo de software. • Usar navegadores de clases y herramientas relacionadas durante el desarrollo de aplicaciones usando APIs. • Diseñar, implementar, probar y depurar programas que usan paquetes API de larga escala.	• Programación usando API. • Diseño de API. • Navegadores de clases (*Class browsers*) y herramientas relacionadas. • Depuración en el entorno API. • Introducción a la computación basada en componentes. [3]	5	9-16Jul	95%

SE/Usando APIs. (1 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Discutir los retos de mantener software heredado.	• Técnicas de modelamiento del análisis de requerimientos. • Prototipeo.	1	17Jul	97%

• Usar un método común, no formal para modelar y especificar (en la forma de un documento de especificación de requerimientos) los requerimientos para un sistema de software de tamaño medio. • Traducir en lenguaje natural una especificación de requerimientos de software escrita en un lenguaje de especificación formal comúnmente usado.	• Conceptos básicos de técnicas de especificación formal. [3]			

SE/Validación y verificación de software. (2 horas)

Objetivos Específicos:	**Contenidos:**	**Hrs**	**Fecha**	**Avance %**
• Distinguir entre validación de programas y verificación. • Describir el rol que las herramientas pueden jugar en la validación de software. • Discutir los temas concernientes a la prueba de software orientado a objetos.	• Fundamentos del *Testing* incluyendo la creación de planes de prueba y la generación de casos de prueba. • Prueba orientado a objetos, pruebas de sistema. [3]	2	23-24Jul	100%

1.5 ACTIVIDADES

- Asignaciones
- Controles de Lectura
- Exposiciones

1.6 RECURSOS MATERIALES:

Los recursos materiales a usar son:

- Apuntes del curso.
- Libro(s) de la bibliografía

1.7 RECURSOS BIBLIOGRÁFICOS:

OBLIGATORIA:

[1] R. Johnsonbaugh. *Matemáticas Discretas*. Pearson, 1999.

[3] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley., 1997.

RECOMENDADA:

[2] Bertran Meyer. *Construcción de Software Orientado a Objetos*. Prentice Hall, 1998.

1.8 METODOLOGIA:

- Clase Magistral.
- Taller didáctico.
- Prácticas personales y en grupo

1.9 EVALUACION.-

La nota final (*NF*) se obtiene de la siguiente manera:

NE Nota de Exámenes 50 %, esta nota se divide en
- Examen Parcial 15%
- Examen Final 25%

NT Nota de Trabajos e Intervención en clase 50%

$$NF = (NE + NT)/2$$

Arequipa, Abril del 2012

Ernesto Cuadros-Vargas
Docente DAISI – FIPS – UNSA