Plan 258 Ing. Tec. en Informática de Sist.

Asignatura 16557 PROGRAMACION III

Grupo 1

Presentación

Técnicas de Construcción y Validación de Programas

Programa Básico

Introducción. Herencia. Polimorfismo. Clases diferidas y efectivas.Patrones de Diseño. Programación Bajo Contrato. Prueba de clases.

Objetivos

Proporcionar al estudiante los conocimientos teóricos y prácticos que le permitan elaborar el diseño detallado, la implementación y la validación de sistemas software Orientados a Objetos

Competencias a adquirir:
-Conocer y utilizar oportunamente los conceptos fundamentales de la Programación Orientada a Objetos
-Comprender y aplicar las técnicas de genericidad y herencia en el marco de la Programación Orientada a Objetos
-Elaborar programas de dificultad moderada mediante el lenguaje de programación Eiffel
-Juzgar de forma realista la calidad de la implementación de los programas realizados
-Utilizar de forma adecuada el «Framework» eGTK para la elaboración de interfaces de usuario de complejidad moderada
-Comprender un conjunto representativo de Patrones de Diseño
-Implementar un conjunto representativo de Patrones de Diseño en Eiffel
-Aplicar los Patrones de Diseño oportunos en el diseño detallado de sistemas informáticos de pequeña complejidad
-Utilizar adecuadamente las técnicas de programación bajo contrato en lenguaje Eiffel

Programa de Teoría

1.-Introducción: Bloque en que se repasan conceptos básicos de la programación Orientada a Objetos. En este bloque se asume que los alumnos conocen ya las definiciones introducidas de la asignatura Programación II, de modo que el énfasis se pone en la relación entre esos conceptos y el estilo adecuado de programación cuando se utilizan técnicas Orientadas a Objetos.

- Clases
- Objetos
- Clases genéricas
- El estilo OO

2.-Herencia: Bloque en que se estudian en detalle las técnicas y conceptos relacionados con la herencia, haciendo énfasis especial en cuándo utilizar, y cuando no, el mecanismo de herencia.

- Introducción
- Polimorfismo y ligadura dinámica
- Clases diferidas y efectivas
- Problemas con el chequeo de tipos

3.-Patrones: Los patrones de diseño son un buen campo de pruebas para las técnicas de herencia, que se utilizan de forma intensiva en este bloque.
Además los patrones de diseño proporcionan una guia básica para el diseño detallado de buenos sistemas software. En este sentido el objetivo es doble: Que los alumnos comprendan de forma general el concepto de patrón de diseño y que comprendan y aprendan a utilizar un conjunto pequeño pero representativo de patrones de diseño.

- Introducción
- Patrones estructurales

Patrones de comportamiento
Patrones creacionales

4.-Contratos: Los contratos software permiten garantizar la calidad de un sistema software, además de facilitar su documentación, depuración y mantenimiento. En este bloque la atención se centra no en resolver un problema mediante un sistema software, sino en asegurar y contrastar la calidad y facilidad de mantenimiento del mismo.

Introducción
Cláusulas require/ensure
Invariantes de clase
Asertos en el código
Contratos y herencia
Proceso de excepciones
Prueba de clases

Programa Práctico

En las sesiones prácticas el alumno elaborará un pequeño sistema informático utilizando las herramientas, métodos y técnicas introducidos en las sesiones teóricas de la asignatura.

Para la interfaz de usuario de este sistema informático, el alumno deberá familiarizarse previamente con el «Framework» eGTK, utilizando para ello la documentación interna que lo acompaña.

Evaluación

Consideraciones Generales

La asignatura tiene unos objetivos más centrados en el «saber hacer» que en el «conocer», por lo que buena parte de la evaluación se centra en la resolución de ejercicios, ya sea en papel en el marco de un examen, o mediante la construcción de software en el laboratorio.

La parte «teórica» de la evaluación no persigue determinar si el alumno es capaz de repetir definiciones, sino si las ha comprendido y está en condiciones de utilizar estos conceptos desde un punto de vista práctico.

La parte práctica de la evaluación no persigue determinar si los alumnos son capaces de encontrar una solución para el ejercicio - se supone que todos los alumnos de esta asignatura son ya capaces de programar pequeños sistemas de software - sino si son capaces de proporcionar una buena solución.

Criterios de evaluación

- La nota final de la asignatura se obtiene ponderando con un 80% la nota del examen y un 20% la nota de la práctica. No es necesario alcanzar nota mínima en ninguna de las dos partes.

- El sistema de recogida de la práctica se basa en un programa automático que se ajusta exactamente a las condiciones de entrega. Tratándose de una asignatura de estas características cualquier práctica que no se ajuste exactamente a las condiciones de entrega se considerará como no presentada.

Instrumentos de evaluación

- Examen final, formado por cuestiones de respuesta corta o preguntas de tipo «test» y problemas.

- Práctica evaluable de la asignatura, formada por varias fases incrementales que acaban construyendo un pequeño sistema software.

Recomendaciones para la evaluación.

- El estudio de los conceptos presentados en esta asignatura no puede ser un ejercicio memorístico, sino que debe permitir abordar con garantías la resolución de problemas concretos aplicando estas técnicas. En este sentido estudiar significa resolver problemas, y el estudio de la teoría, si bien es necesario, debe tener por objetivo saber resolver los problemas y saber cuándo un problema ha sido correctamente resuelto.

Recomendaciones para la recuperación.

- Las mismas que para el estudio, resolver problemas de programación ypreguntarse si la resolución utiliza las técnicas adecuadas en cada caso.

Bibliografía