UNIVERSIDAD CATOLICA DEL MAULE

Nota

Evaluación 2 (30%)

Nombre	
Carrera	
Actividad curricular	Programación Orientada a Objetos
Académico	Francisco Philip Vásquez Iglesias
Fecha de Aplicación	
Duración	90 minutos

Puntaje máximo	100 puntos	**Puntaje de corte**	60 puntos	**Puntaje obtenido**	

Resultados de aprendizajes evaluados	2.- Elaborar programas a través de la implementación de conceptos orientado a objeto y relaciones entre clases, comunicando resultados a través de informes en español de acuerdo a pautas establecidas.

Indicadores de evaluación	- Aplicación de los pilares de la Orientación a Objetos.
	- Diseño de un programa que soluciona un problema regional real o simulado:
	  - Clases y creación de objetos, Atributos y variables de clase, Constructor, Métodos de instancia y de clase, Visibilidad de clases, métodos y atributos. Implementación de encapsulamiento, Implementación de herencia, Implementación de polimorfismo, Manejo de errores.
	  - Implementación de relaciones entre clases: relación de uso, agregación y composición. Interfaces, Paquetes.

Instrucciones para el desarrollo de la evaluación
La evaluación 2 de Programación Orientada a Objetos versión 2025, consiste en la resolución de un par de ejercicios utilizando su conocimiento de UML y POO, con una creación de informe recopilatorio con la solución propuesta que será presentada por el equipo. El tiempo de desarrollo de la actividad es de 7 días y se debe realizar en grupos de 3 estudiantes. Resolver los ejercicios: 1. Sistema Académico UCM y 2. Pioneros de la Computación, con indicaciones formateadas en calibrí 11 color negro. Puntaje máximo de la evaluación: 100 pts. Puntaje necesario para el 4.0: 60 pts.

1. **Sistema Académico UCM**

Se requiere modelar un sistema académico de la Universidad Católica del Maule utilizando Programación Orientada a Objetos. Debes definir una clase abstracta Persona con atributos comunes como nombre, RUT y correo, y métodos apropiados. A partir de ella, crea las clases Profesor y Estudiante, que heredan de Persona, implementando sus propias versiones del método mostrar_info() y comportamientos particulares. Un profesor puede dictar múltiples asignaturas, y un estudiante puede inscribirse en varias asignaturas, donde puede registrar múltiples notas por asignatura. Cada Asignatura tiene nombre, código, cupo máximo, un profesor a cargo (asociación) y una lista de estudiantes inscritos. Un estudiante debe poder calcular su promedio por asignatura y su promedio general, estos métodos debe implementarlos mediante sobrecarga de operadores. Las inscripciones deben validar cupo disponible y evitar que un estudiante se inscriba dos veces en la misma asignatura (manejo de errores). Una Carrera está compuesta (composición) por varias asignaturas, y debe permitir buscar estudiantes por nombre.

Aplicar encapsulamiento, herencia, composición, agregación (el estudiante mantiene una referencia a las asignaturas), asociación (profesor–asignatura), métodos polimórficos, y al menos 10 métodos distribuidos en las clases. Debes instanciar al menos una carrera, tres asignaturas, tres profesores y tres estudiantes con distintas inscripciones y notas. El sistema debe poder imprimir por consola toda la información relevante.

Los estudiantes deben crear e instanciar lo siguiente:

1. Carrera
    a. Crear una carrera llamada “Ingeniería Civil Informática”.
    b. Esta carrera debe contener 3 asignaturas (ver más abajo).
    c. Utilizar composición: la carrera "posee" sus asignaturas.
2. Asignaturas (3)
    a. Crear 3 asignaturas distintas, por ejemplo:
        i. “Programación Orientada a Objetos” (ICE234)
        ii. “Bases de Datos” (ICE221)
        iii. “Estructuras de Datos” (ICE222)
    b. Asignar un cupo máximo de 2 estudiantes por asignatura.
    c. Asignar un profesor a cada asignatura mediante asociación.
    d. Deben ser agregadas a la carrera.
3. Profesores (3)
    a. Crear 3 profesores distintos.
    b. Cada profesor debe dictar al menos una asignatura.
    c. Se espera que el profesor tenga un método para listar las asignaturas que dicta.
4. Estudiantes (3)
    a. Crear 3 estudiantes distintos.
    b. Cada uno debe inscribirse en al menos 2 asignaturas distintas.
    c. Validar: no inscribir dos veces a la misma asignatura.
    d. Validar: no inscribir si el cupo ya está lleno.

5. Notas y cálculo de promedio
    a. Agregar al menos 2 notas por asignatura por estudiante.

   b. Implementar método para:
      i. Calcular el promedio por asignatura.
      ii. Calcular el promedio general del estudiante.
6. Salidas esperadas
   a. Mostrar por consola:
      i. Información detallada de cada estudiante (mostrar_info()).
      ii. Asignaturas dictadas por cada profesor.
      iii. Lista de estudiantes por asignatura.
      iv. Promedio general de al menos un estudiante.
      v. Validaciones: intento fallido de reinscripción o sobrecupo.

2. Pioneros de la Computación

Crea un sistema orientado a objetos para representar a científicos históricos influyentes en el desarrollo de la computación. Implementa una clase abstracta Cientifico con atributos como nombre, país, campo de estudio y una lista de proyectos en los que participó. Heredan de ella al menos tres clases: Matematico, Ingeniero y Criptografo, cada una con una implementación propia del método contribucion(), que describe su impacto histórico, y del método resumen(), que debe ser polimórfico. Cada Proyecto tiene nombre, año, científicos asociados (agregación), y está vinculado a una única Institucion (composición), que contiene una lista de proyectos propios. Debes evitar que un científico se agregue dos veces al mismo proyecto (manejo de errores). Los proyectos deben permitir listar científicos involucrados y buscar por nombre. Las instituciones deben permitir listar todos los científicos que han trabajado en sus proyectos y buscar proyectos por nombre. Aplica herencia, encapsulamiento, polimorfismo (sobrecarga de operadores, sobrecarga de métodos y sobrescritura de métodos), composición, agregación, asociación, y valida todas las operaciones críticas con manejo de excepciones. Debes crear al menos una institución, dos proyectos y tres científicos distintos, distribuidos adecuadamente, con salidas por consola que verifiquen el correcto funcionamiento del modelo.

Los estudiantes deben crear e instanciar lo siguiente:

1. Institución
    a. Crear una institución histórica, por ejemplo:
        i. “ENIAC Lab”, ubicada en EE.UU.
    b. Esta institución debe contener 2 proyectos (ver más abajo).
    c. Debe ser una composición: si se elimina la institución, desaparecen sus proyectos.
2. Proyectos (2)
    a. Crear 2 proyectos históricos con año de creación, por ejemplo:
        i. “ENIAC” (1945)
        ii. “Bombe” (1940)
    b. Cada proyecto debe tener al menos 2 científicos participantes.
    c. Implementar validación: no se puede agregar dos veces al mismo científico.
3. Científicos (3)
    a. Crear 3 científicos, cada uno de una subclase distinta:
        i. Alan Turing → Criptógrafo
        ii. Ada Lovelace → Matemática
        iii. John von Neumann → Ingeniero
    b. Asignar cada científico a uno o dos proyectos.
    c. Implementar método contribucion() específico para cada uno.
    d. Implementar método polimórfico resumen().
4. Validaciones y errores
    a. Mostrar error si se intenta agregar el mismo científico dos veces a un proyecto.
    b. Asegurar que cada científico solo aparezca una vez por proyecto.
5. Salidas esperadas
    a. Listar científicos involucrados en cada proyecto.
    b. Mostrar resumen de contribuciones (polimorfismo).
    c. Mostrar todos los proyectos de una institución.
    d. Mostrar todos los científicos asociados a una institución.

Pauta de revisión (65%)

Nombres:	1.- 2.- 3.-								
Caps.	Aspecto Solicitado	100%	80%	60%	50%	30%	0%	Pje	Obtenido
1.1)	Incluyen una introducción al trabajo realizado, abordan el contexto del ejercicio solicitado, POO, C++, entre otros.						Otro caso/ no cumple	5p	
2.1)	Realizan Diagrama UML que incluye las clases solicitadas, incluyendo encapsulamiento, relaciones entre objetos, herencia, instancias solicitadas, entre otros.						Otro caso/ no cumple	10p	
2.2)	Cumplimiento de las funcionalidad solicitadas por enunciado demostrando conocimientos y dominio en POO.						Otro caso/ no cumple	15p	
2.3)	Aplican correctamente sobrecarga y sobreescritura de métodos.						Otro caso/ no cumple	5p	
2.4)	Aplican correctamente la Agregación y Composición.						Otro caso/ no cumple	5p	
2.5)	Aplican correctamente la herencia.						Otro caso/ no cumple	5p	
2.6)	Aplica sobrecarga de operadores correctamente						Otro caso/ no cumple	5p	
3.1)	Realizan Diagrama UML que incluye las clases solicitadas, incluyendo encapsulamiento, relaciones entre objetos, herencia, instancias solicitadas, entre otros.						Otro caso/ no cumple	10p	
3.2)	Cumplimiento de las funcionalidad solicitadas por enunciado demostrando conocimientos y dominio en POO.						Otro caso/ no cumple	15p	
3.3)	Aplican correctamente sobrecarga y sobreescritura de métodos.						Otro caso/ no cumple	5p	
3.4)	Aplican correctamente la Agregación y Composición.						Otro caso/ no cumple	5p	
3.5)	Aplican correctamente la herencia.						Otro caso/ no cumple	5p	

3.6)	Aplica sobrecarga de operadores correctamente						Otro caso/ no cumple	5p	
4.1)	Incluyen conclusión al trabajo realizado, donde son mencionados los aprendizajes obtenidos, la utilidad del paradigma y el cierre del trabajo mismo.						Otro caso/ no cumple	5p	
Comentarios y descuentos			Total			100 puntos con 60% de exigencia			

Preguntas conceptuales (35%).

Rúbrica de preguntas							
Pregunta	**100%**	**70%**	**50%**	**30%**	**10%**	**0%**	**Obtenido**
Pregunta 1			Responde otro compañero.				
Pregunta 2			Responde otro compañero.				
Pregunta 3			Responde otro compañero.				
20 puntos x pregunta = 60 puntos en total con 60% de exigencia							