Asignatura	**Programación II**
Carrera	**Ing. en Informática**
Plan	**Ajuste 2023**
Ciclo	**2do**
Cuatrimestre	**1ero**
Tema/Título	**Informe de diferencias entre conceptos**
Profesor	**Adrian Tozzi**

Grupo de Trabajo

ID/Matrícula	APELLIDO, Nombres	Correo Electrónico
000-20-1141	Cannizzaro, Pablo	Pablom.cannizzaro@comunidad.ub.edu.ar
000-20-1394	Callizaya, Leandro	leandro.callizaya@comunidad.ub.edu.ar
000-20-0991	Conde Buades, Joaquín	Joaquin.conde@comunidad.ub.edu.ar
000-20-1147	Franco Dalla Via Oliveros	Franco.dalla@comunidad.ub.edu.ar

Grilla de calificación

Concepto	Propuesta	Marco Teórico	Desarrollo propio	Conclusiones	Fuentes y Referencias
Sobresaliente (10)					
Distinguido (9-8)					
Bueno (7-6)					
Aprobado (5-4)					
Insuficiente (3-2-1)					
Reprobado (0)					
NOTA					

Comentario adicional del Profesor:

Enunciado de la actividad:

Que diferencia hay entre:

- ☐ **Abstracción**
- ☐ **Encapsulamiento**
- ☐ **Modularidad**
- ☐ **Herencia**
- ☐ **Polimorfismo**

Respuesta:

En la Programación Orientada a Objetos, hay varios conceptos que definen cómo se estructura y organiza el código. A continuación, veremos las diferencias entre abstracción, encapsulamiento, modularidad, herencia y polimorfismo.

Abstracción:

es el proceso de simplificar el diseño y el uso de objetos ocultando la complejidad innecesaria, que está en el método

Encapsulamiento:

Se refiere a ocultar el estado interno de un objeto y controlar el acceso a él mediante métodos. El objetivo es proteger los datos de una clase y garantizar que solo se modifiquen de manera controlada.

Modularidad:

Se refiere a la idea de dividir el sistema en módulos (o clases) independientes que pueden ser desarrollados, probados y mantenidos de forma aislada

Herencia:

Es un mecanismo que permite que una clase herede propiedades y comportamientos de otra clase. Para Reutilizar código, extender funcionalidades y modelar relaciones jerárquicas entre objetos.

Polimorfismo:

Permite que una misma interfaz o método se comporte de manera diferente dependiendo del tipo de objeto que lo invoca.