IPOO

Docentes

- Profesores Adjuntos:
  - Carlos Di Cicco.
- JTPs:
  - Federico Naso (Junín) .
    Nelson Di Grazia (Pergamino) .

IPOO

- Evaluaciones
  - Para aprobar la cursada:
    - Parcial
    - Entrega de trabajo práctico o de insvestigación
  - Para aprobar la materia:
    - Examen final

IPOO – Contenidos

- Objetos, Plataforma Java
- Lenguaje Java
- Herencia. Polimorfismo
- Interfaces
- Excepciones
- Colecciones
- Swing

Antes de empezar…

- ¿Qué recuerdan de objetos?
- ¿Qué vieron hasta ahora en objetos y otros paradigmas de programación?
- ¿Hicieron algún desarrollo?
- Inglés!!!
- Repaso

Objetos

- ¿Por qué objetos?
  - Modelado de la realidad
  - Mercado volcado a OO
  - Lenguajes populares: Java, C#, C++
  - "Nuevos lenguajes": Python, Ruby, Objetive-C
  - Nuevos desarrollos: web, celulares
  - Comunidad Software Libre
  - Grandes Empresas: Microsoft, IBM, Oracle, Apple
  - Componentes: librerías y frameworks
  - Escalabilidad

Programa Orientado a Objetos

- Conjunto de objetos que colaboran enviándose mensajes
- Entonces:
  - Sólo hay objetos
  - Lo único que pueden hacer es enviar y recibir mensajes
  - Si querés que algo se haga se necesita un objeto que lo haga y otro objeto que le envíe un mensaje

Objeto

- Encapsula funcionalidad e información:
  - Retiene información
  - Sabe como realizar ciertas operaciones
- A diferencia del diseño estructurado aquí las "operaciones" y la "información" están juntas y sólo se puede acceder a esa información a través de esas operaciones
- ¿Cómo se almacena la información en un objeto?
- ¿Cómo se implementa la funcionalidad?

Encapsulamiento

- Es el proceso de agrupar dentro de un objeto “colaboradores” y “comportamiento”.
- Es una de las principales claves para conseguir software confiable.
- El encapsulamiento permite que los cambios hechos en los programas sean fiables con el menor esfuerzo.
- Una de las premisas de la programación orientada a objetos es tratar de no violar el encapsulamiento.
- Ejemplo: persona

Comportamiento

- Indica **que sabe** hacer el objeto, es decir sus responsabilidades.
- Se especifica a través del conjunto de mensajes que puede recibir el objeto.
- Se implementan con métodos.
- ¿Cuál es la relación entre mensajes y métodos?

Implementación

- Indica **cómo** hace el objeto para responder a sus mensajes.
- Es especificado mediante:
  - Un conjunto de colaboradores.
  - Un conjunto de métodos.
- Es privado del objeto. Ningún otro objeto debe acceder, lo invoca.
- ¿Dónde se escribe el código de los objetos?
- ¿Cómo implemento un objeto?

Estado interno

- La representación interna de un objeto es su lado "privado", sólo tenemos acceso a aquellas partes de su estado que el objeto revela mediante su interfaz pública
- Estado interno de un objeto:
  - La información almacenada dentro de un objeto conforman su estado interno.
  - El estado interno de un objeto puede ser cambiado sólo a través de las operaciones provistas por el objeto para dicho fin.

Ocultamiento

- El lado privado maneja la información que es necesaria para el funcionamiento interno de un objeto, pero innecesaria para los demás objetos
- En él se especifica:
  - Como lleva a cabo los requerimientos que le hacen otros objetos
  - Como representa la información que mantiene
- ¿Cómo se implementa el ocultamiento? ¿En UML?

Ocultamiento

- La manera en que el objeto lleva a cabo estos requerimientos o trata su información “no es asunto” de los demás objetos.
- De esta manera los objetos pueden cambiar el modo de realizar ciertas operaciones o representar cierta información sin afectar al resto del sistema.
- Ejemplo: cálculo de saldo, edad

Responsabilidad

- Un objeto sabe cumplir sólo su rol y el de ningún otro objeto dentro del sistema.
- No existe información fuera de objetos.
- No existen operaciones fuera de objetos.
- El diseño orientado a objetos, estructura responsabilidades mediante las preguntas: ¿qué puede hacer este objeto? y ¿qué conoce este objeto?
- ¿Quién conoce la edad de una persona?

Método

- Lo único que puede hacer un objeto cuando recibe un mensaje es enviar mensajes (colaborar) a otros objetos (colaboradores).
- Entonces, un método es simplemente
  - El conjunto de "colaboraciones" que lleva a cabo un objeto para responder un mensaje.
- Al recibir un mensaje, un objeto lleva a cabo la operación mediante la ejecución de un método.

Método

- El método es el algoritmo particular con el que el objeto realiza dicha operación.
- Un método está asociado a un mensaje. Generalmente con el mismo nombre.

Clases

- Los objetos dentro de un sistema no son completamente distintos uno del otro.
- Distintos objetos pueden comportarse de una manera muy similar.
- Los objetos que comparten el mismo comportamiento pertenecen a la misma clase.
- La clase es el molde para generar objetos.
- Todos los objetos de la misma clase se comportan de la misma forma, lo que varía es el estado interno de cada uno.
- ¿Qué incluye el código de una clase?

Instancias

- Los objetos que se comportan de la manera descripta en una clase son llamados instancias de esa clase.
- Todo objeto es instancia de alguna clase.
- Una instancia de una clase se comporta de la misma manera que las demás instancias de esa clase.
- Almacena su información en variables de instancia. ¿Donde se definen las variables?

Clases e Instancias

- Todo objeto es instancia de alguna clase
- La clase es el molde, la instancia el objeto
- Toda instancia de la misma clase se comporta de la misma manera. Lo que varía es el estado interno de la instancia
- Hay variables y métodos de clase y de instancia
- A nivel instancia puedo acceder a la clase, pero no viceversa
- ¿Para que sirven las variables y métodos de clase? ¿Que pasa si modifico el valor de una variable de clase?

Polimorfismo

- Capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.
- Capacidad de OO de permitir el envío del mismo mensaje a objetos de diferentes clases.
- ¿Qué características tienen los métodos polimórficos?

Herencia

- Toda clase es subclase de otra (excepto la clase Object) y hereda comportamiento de su superclase.
- Jerarquía de especialización o generalización.
- Subclases son especializaciones de la superclase y superclase es generalización de las subclases.
- Ejemplo: persona, estudiante y profesor

Herencia

- Una subclase a través de herencia puede extender o reducir el comportamiento de su superclase.
- Relación “es un”:
  - X es un Y si toda instancia de X es también instancia de Y
  - Un objeto de una subclase puede usarse en cualquier lugar donde se admita un objeto de la superclase pero no viceversa.

Herencia

- Clasificación:
  - De Especialización: + usada
  - De Especificación o comportamiento (abstractas)
  - De Estructura: para reutilizar comportamiento (uso incorrecto)
- Tipos:
  - Simple: Solo una superclase
  - Múltiple: varias superclases
- ¿Que tipo de herencia han visto implementada?

Nomenclatura

- Seleccionar nombres que clarifiquen el propósito del objeto y/o del método.
- Nombres descriptivos requieren de pocos comentarios explicativos.
- Mayúscula: Clases y variables de clases
- Minúscula: métodos de clase e instancia, los parámetros de los métodos, variables temporales y variables de instancias.

Objetos

- ¿Dudas? ¿Preguntas?

Patrones de diseño

- Describe un problema que debemos resolver constantemente, luego describe la idea principal de la solución de manera que pueda ser utilizada en diferentes contextos una y otra vez.

- Pensar: “idea de solución”, “abstracción de solución”, “aplicable en diferentes contextos”

Objetivo de los Patrones

- Pretenden:
  - Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
  - Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
  - Formalizar un vocabulario común entre diseñadores.
  - Estandarizar el modo en que se realiza el diseño.
  - Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Objetivo de los Patrones

- No pretenden:
  - Imponer ciertas alternativas de diseño frente a otras.
  - Eliminar la creatividad inherente al proceso de diseño.
  - Forzar la implementación de soluciones OO utilizándolos
- Existe un catálogo de patrones
- Tienen un formato específico
- No aseguran el éxito del desarrollo
- Existen los antipatrones…

Patrones de diseño

- Nombre del patrón: nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés).
- Clasificación del patrón: creacional, estructural o de comportamiento.
- Intención: ¿Qué problema resuelve el patrón?
- También conocido como: Otros nombres de uso común para el patrón.

Patrones de diseño

- Motivación: Escenario de ejemplo para la aplicación del patrón.
- Aplicabilidad: Criterios de aplicabilidad del patrón.
- Estructura: Diagramas de clases oportunos para describir las clases que intervienen en el patrón.
- Participantes: Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón.

Patrones de diseño

- Colaboraciones: Explicación de las interrelaciones que se dan entre los participantes.
- Consecuencias: Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.
- Implementación: Técnicas o comentarios oportunos de cara a la implementación del patrón.
- Código de ejemplo: Código fuente ejemplo de implementación del patrón.
- Usos conocidos: Ejemplos de sistemas reales que usan el patrón.
- Patrones relacionados: Referencias cruzadas con otros patrones.

Patrones de diseño

- ¿Dudas? ¿Preguntas?