Curso de Programación en Java

Programación Orientada a Objetos

Características de la POO

- Abstracción
  - Aislar del contexto
  - Características esenciales de un objeto
  - Provee límites conceptuales
- Herencia
  - Clases basadas en clases preexistentes
    - Comportamiento
    - Atributos

Características de la POO (cont.)

- Herencia
  - Extiende la funcionalidad de la clase padre
    - Clase base
    - Superclase
    - Clase ancestro
  - Herencia con Java
    - extends
    - super
    - this

```
public class Triangulo extends Figura2D {
    ...
}
```

Características de la POO (cont.)

- Polimorfismo
  - Programar en forma general
  - Características comunes
  - Por sobrecarga
    - Más de un método con el mismo nombre, pero diferentes acciones
    - Válida cuando:
      - Diferentes firmas
        - Numero de argumentos
        - Tipo de argumentos
      - El valor de retorno no distingue a un método

Características de la POO (cont.)

- Polimorfismo
  - Por sobreescritura
    - Método heredado
    - Diferente contenido
    - Válida cuando:
      - Tienen la misma firma
      - Tienen el mismo tipo de retorno
      - Cambio de visibilidad por uno más abierto

Características de la POO (cont.)

- Encapsulación
  - Proteger los datos del exterior
  - Ocultar el estado de un objeto
  - Acceso a través de métodos definidos
  - Exponer las operaciones necesarias
  - Beneficios
    - Componentes discretos, autocontenidos
    - Reutilización de objetos
    - Ubicar el origen de los bugs
    - Modificaciones modulares

Ventajas

- Código reutilizable
- Aplicaciones extensibles
- Modelado de software en términos que el cliente entiende.

Práctica

Programación Orientada a Objetos

Modificadores de acceso

- public
  - Visible desde cualquier lugar de la aplicación
- protected
  - Clase
  - Package
  - Subclase

Modificadores de acceso

- default, package, friendly
  - Clase
  - Package
- private
  - Clase

Modificador	Clase	Package	Subclase	Todos
public	Si	Si	Si	Si
protected	Si	Si	Si	No
default	Si	Si	No	No
private	Si	No	No	No

Práctica

Modificadores de acceso

Interface

- Colección de métodos y propiedades
- Define el qué, no el cómo
- Siempre públicos y abstractos, no es necesario especificarlo
- Plantillas o contratos
- Definiciones no código
- Clase implementa una interfaz
- Clase puede implementar más de una interfaz

Interface (cont.)

- Ventajas
  - Organiza programación
  - Desacoplar código
- Regas
  - Clases con clases -> Se heredan
  - Interfaces con Interfaces -> Se heredan
  - Clase con interfaces -> Se implementan
- Una interfaz puede heredar más de una interfaz

Práctica

Uso de interfaces

La clase Object

- Jerarquia mas alta del entorno de desarrollo
- Todas las clases son descendientes directos o indirectos
- Define estados y comportamientos básicos

La clase Object (cont.)

- Métodos importantes
  - toString
    - Devuelve un String que representa al objeto
    - Sumamente útil
  - equals
    - Compara dos objetos para determinar si son el mismo
    - Object compara referencias
    - Sobreescribir para determinar cuando un objeto es el mismo

La clase Object (cont.)

- Métodos importantes
  - hashCode
    - Identificador de 32 bits almacenado en un hash
    - Debe sobreescribirse al sobreescribir equals
    - Entero sin signo
    - Si 2 objetos son iguales, deberían devolver el mismo hashCode
    - Si 2 objetos son son diferente, no necesariamente deben tener diferente hashCode

Práctica

toString(), equals(), hashCode()