Desarrollo Avanzado de Aplicaciones I

Programación Orientada a Objetos 1

idat

i d a t

inicio desarrollo aplicación término

idat

Inicio

Logro de aprendizaje - Introducción

idat

Logro de Aprendizaje

“Al finalizar la sesión, el participante podrá usar clases instanciadas y no instanciadas, clases selladas, abstractas, heredables y no heredables, sus modificadores y la referencia this.”

idat

Introducción

idat

- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Clases y Objetos
  - Atributos y Operaciones
  - Referencia this
  - Modificadores de Clases y Variables
  - Paquetes

Desarrollo

Desarrollo del Contenido de la Sesión

idat

Clases y Objetos

idat

- ✓ La programación orientada a objetos intenta representar la realidad donde se encuentran muchos objetos del mismo tipo (clases).
- ✓ Una clase es una estructura (plantilla, modelo, etc.) que define características generales de un tipo de objetos.
- ✓ Las características de una clase son sus atributos (variables) y operaciones (métodos).
- ✓ Un objeto es un ejemplar concreto de una clase, que se estructura y comporta según se definió en la clase, pero sus características son particulares e independiente del resto de ejemplares.
- ✓ Al proceso de crear un objeto se le conoce como instanciar una clase.

Ejemplos de Clases

Clase
atributo 1 atributo 2 … atributo N
método 1 método 2 … método N

Automóvil
placa marca modelo color
frenar() acelerar() pintar() equipar()

Celular
marca modelo memoria pantalla
llamar() prender() apagar() reiniciar()

Persona
nombre apellido edad peso
correr() crecer() adelgazar() cantar()

Clases – Estructura y Declaración

idat

- ✓ Estructura de una clase:

```
class nombreClase {
        [atributos o variables de la clase]
        [operaciones o métodos de la clase]
}
```

- ✓ Declaración de una clase:

```
[modificadoresdeclase] class NombreClase [extends ClasePadre] [permits lista subclases] [implements Interfaces] {
        [modificadoresdevariable] tipodedato nombreVariable [= valor];
        [modificadoresdemetodo] tipodevuelto nombreMetodo(lista parametros) [throws listaExcepciones] {
                // sentencias
                return valor;
        }
}
```

Atributos (Variables)

idat

- ✓ Un atributo es una información que se almacena en memoria y es parte representativa del estado de la clase.
- ✓ Los atributos son variables de clase o variables globales.
- ✓ Los atributos se pueden declarar mediante tipos de datos primitivos u otras clases.
- ✓ Los atributos pueden ser usados en cualquier parte de la clase.
- ✓ Los objetos o instancias de la clase heredan todos los atributos de la clase incluso los estáticos. Una vez heredados se denominan variables de instancia.
- ✓ Si no se inicializan los atributos, se les asigna el valor por defecto que le corresponde (cero, false o null).

Operaciones (Métodos)

idat

- ✓ Los métodos se pueden organizar en cuatro tipos:
  - ➢ Constructores. Métodos que inicializan la instancia.
  - ➢ Método main. Método que permite ejecutar la clase.
  - ➢ Métodos genéricos. Realizan funcionalidades específicas.
  - ➢ Métodos para acceso directo a los atributos (getters y setters).
- ✓ Todos los tipos de métodos son opcionales, se pueden usar de acuerdo a las necesidades. Si no se específica un método constructor, toda clase tiene un método constructor por defecto.
- ✓ Los objetos o instancias de la clase heredan todos los métodos de la clase incluso los marcados como estáticos.

Instanciando Clases

idat

- ✓ Un objeto es la referencia e instancia de una clase. Al crear una referencia se asigna un espacio de memoria dinámica al objeto, pero no es utilizable. Al crear la instancia, el objeto es utilizable.
- ✓ Por tanto, se debe realizar estas dos acciones:
  - ➢ Declarar una variable que pueda referenciar al objeto en cuestión.
  - ➢ Crear el objeto y asociarlo a la variable. Para crear el objeto, se realiza con la sentencia new. Recordar que lleva implícito llamar al método constructor para que inicialice el objeto.
- ✓ Podemos crear todos los objetos que queramos, todos ellos tiene existencia propia, pudiendo evolucionar por caminos diferentes e independientes.

Instanciando Clases (cont.)

- ✓ La sintaxis de la referencia es:
  - ➢ MiClase m;
  - ➢ Donde m es la referencia del objeto.
- ✓ La sintaxis de la instancia es:
  - ➢ m = new MiClase();
- ✓ Al crear la instancia se puede acceder a los atributos y métodos públicos, protegidos y predeterminados, a través del objeto m.
- ✓ Otra sintaxis para realizar la referencia e instancia en la misma línea de código es:
  - ➢ MiClase m = new MiClase();

Referencia this

- ✓ La palabra reservada “this” permite hacer referencia a las variables y métodos de la instancia actual de la clase.

```
public class ReferenciaThis {
    int var1 = 10;
    public static void main(String[] args) {
        ReferenciaThis rt = new ReferenciaThis();
        rt.calcularDoble( var1: 8);
        rt.calcularDoble();
    }
    void calcularDoble(int var1) {
        System.out.println(this.var1 * var1);
        this.calcularDoble();
    }
    void calcularDoble() {
        System.out.println(var1 * 2);
    }
}
```

Ejemplo Práctico de Clase y Objeto

idat

```
class Automovil {
    String placa;
    String marca;
    String modelo;
    String color;
    int velocidad;
    void registrarSunarp (String placa, String marca, String modelo, String color)
        this.placa = placa;
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
    }
    void pintar(String color) {
        this.color = color;
    }
```

Atributos, variables de clase, variables globales

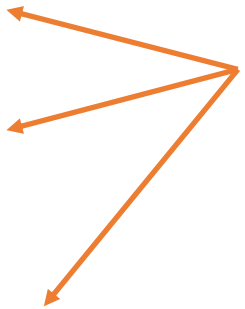
Operaciones, métodos

Java

Ejemplos de Clases (cont.)

idat

```
    void frenar() {
        this.velocidad = 0;
    }
    void acelerar(int aceleracion) {
        this.velocidad += aceleracion;
    }
    void mostrarDatos() {
        System.out.println( x: "Los datos del automovil son: ");
        System.out.println("La placa es: " + this.placa);
        System.out.println("La marca es: " + this.marca);
        System.out.println("El modelo es: " + this.modelo);
        System.out.println("El color es: " + this.color);
        System.out.println("La velocidad actual es: " + this.velocidad + " Km/h \n");
    }
}
```



Operaciones, métodos

Ejemplos de Clases (cont.)

```java
public class MiClase {
    public static void main(String[] args) {
        Automovil auto = new Automovil();
        auto.registrarSunarp( placa: "CCA004",  marca: "Chevrolet",  modelo: "Traverse",  color: "Rojo");
        auto.mostrarDatos();
        auto.pintar( color: "Negro");
        auto.acelerar( aceleracion: 20);
        auto.acelerar( aceleracion: 40);
        auto.acelerar( aceleracion: 30);
        auto.mostrarDatos();
        auto.frenar();
        auto.mostrarDatos();
    }
}
```

Clase principal del archivo

Instancia de la clase Automovil

Invocación a métodos

1

2

3

Ejemplos de Clases (cont.)

```
Los datos del automovil son:
La placa es: CCA004
La marca es: Chevrolet
El modelo es: Traverse
El color es: Rojo
La velocidad actual es: 0 Km/h
```

1

```
Los datos del automovil son:
La placa es: CCA004
La marca es: Chevrolet
El modelo es: Traverse
El color es: Negro
La velocidad actual es: 90 Km/h
```

2

```
Los datos del automovil son:
La placa es: CCA004
La marca es: Chevrolet
El modelo es: Traverse
El color es: Negro
La velocidad actual es: 0 Km/h
```

3

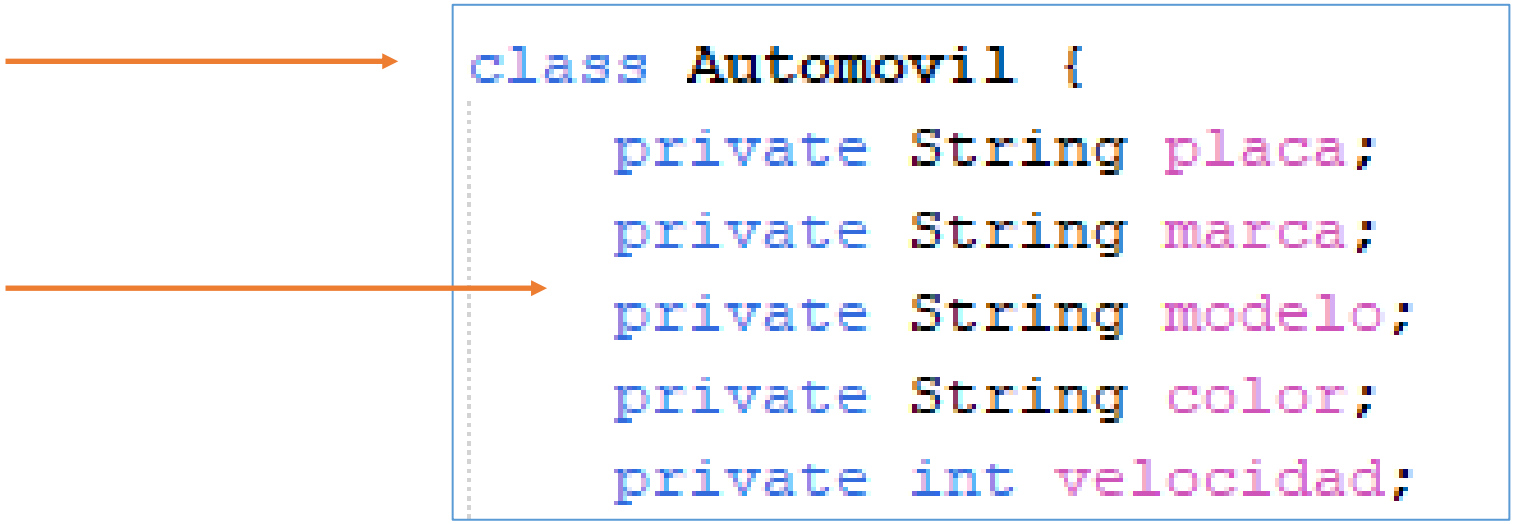
Modificadores de Acceso de Clases, Variables y Métodos

- ✓ Los modificadores de acceso nos permiten establecer el alcance y la accesibilidad o visibilidad para clases, variables y métodos.
- ✓ Los modificadores protegido y privado no pueden ser usados con clases. Si una clase es pública debe ser declarada en su correspondiente archivo .java.

Modificador de acceso	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
Público - public	X	X	X	X
Protegido - protected	X	X	X	
Sin modificador de acceso (predeterminado)	X	X		
Privado - private	X			

Ejemplo de Modificadores de Acceso en Clases y Variables

- ✓ La clase "Automovil" se encuentra dentro de un archivo de otra clase, por lo que no puede tener modificador de acceso.
- ✓ Los atributos de la clase "Automovil" al ser privados solo se pueden modificar a través de los propios métodos de la clase.



```
class Automovil {
    private String placa;
    private String marca;
    private String modelo;
    private String color;
    private int velocidad;
```

Modificadores de Comportamiento de Clases, Variables y Métodos

idat

Modificador	Aplica para	Comportamiento
Estático static	Variables y métodos	Pertenece a la clase y no a las instancias de la clase. Se puede utilizar de forma directa sin instanciar un objeto de la clase. Los métodos estáticos solo pueden invocar directamente a otros métodos y variables estáticos de la clase.
Final final	Clases, variables y métodos	Indica que una variable, método o clase no se va a modificar. Si una variable se marca como final, no se podrá asignar un nuevo valor a la variable. Si una clase se marca como final, no se podrá extender la clase. Si un método se declara como final, no se podrá sobrescribir.
Abstracto abstract	Clases y métodos	Indica que no se provee una implementación para un cierto método, sino que la implementación vendrá dada por las clases que extiendan la clase actual. Una clase que tenga uno o más métodos “abstract” debe declararse como “abstract” a su vez.
Sellada sealed	Clases	Las clases selladas nos permiten restringir que otras clases pueden extenderse de ellas. Se debe agregar la cláusula “permits” seguida de las sub clases que permitimos sean extendidas. Las sub clases extendidas deben usar los modificadores “final”, “sealed” o “non-sealed”.
No sellada non-sealed	Sub clases	Una sub clase permitida puede aplicar el modificar “non-sealed” de manera que revierte en su propia jerarquía el “sellado” de la clase padre, y abriendo la extensión a otros clases desconocidas por la clase padre.

Modificadores de Comportamiento de Clases, Variables y Métodos (cont.)

idat

Modificador	Palabra reservada	Aplica para	Comportamiento
Sincronizado	synchronized	Métodos	El método no se puede ejecutar de forma simultánea en diferentes hilos (threads).
Punto flotante estricto	strictfp	Clases y métodos	Los cálculos con números flotantes se restringen a los tamaños definidos por el estándar de punto flotante de la IEEE.
Nativo	native	Métodos	El método está escrito en un lenguaje distinto a Java. Puede ser C, C++ o assembler.
Transitorio	transient	Variables	Los atributos de un objeto no son parte persistente del objeto. No serialización estándar.
Volátil	volatile	Variables	Los atributos de los objetos van a ser modificado por varios hilos (threads) de forma simultanea y asíncrona.

Constantes

idat

- ✓ Las constantes son variables que tienen el modificador de comportamiento “final” y no pueden ser modificados después de ser inicializados.
  - ➢ final double pi = 3.1416;
- ✓ Las constantes pueden ser de cualquier tipo de dato y la convención indica que deben nombrarse en mayúscula. El compilador lo evidencia solo si se incluye también el modificador de comportamiento “static”.
  - ➢ final static String PREFIJO = “archivo”;
  - ➢ final static double PI = 3.1416
- ✓ Una constante se puede declarar en una sentencia y asignar valor por única vez en una segunda sentencia. No aplica para variables de clase.
  - ➢ final double pi;
  - ➢ pi = 3.1416;

Ejemplo de Constantes

idat

```
public class Constantes {
    int var1 = 10;
    final int var2 = 20;
    final static int VAR3 = 30;
    static int var4 = 40;
    final static String PREFIJO = "archivo";
    public static void main(String[] args) {
        final int var5;
        var5 = 50;
        Constantes con = new Constantes();
        con.var1 = 100;
        con.var1 += con.var2;
        con.var1 += VAR3;
        System.out.println( x: con.var1);
        var4 = var4 + VAR3 + var5;
        System.out.println( x: var4);
        System.out.println( x: PREFIJO);
    }
}
```

Variables de Clase

Constantes

Constante

Java

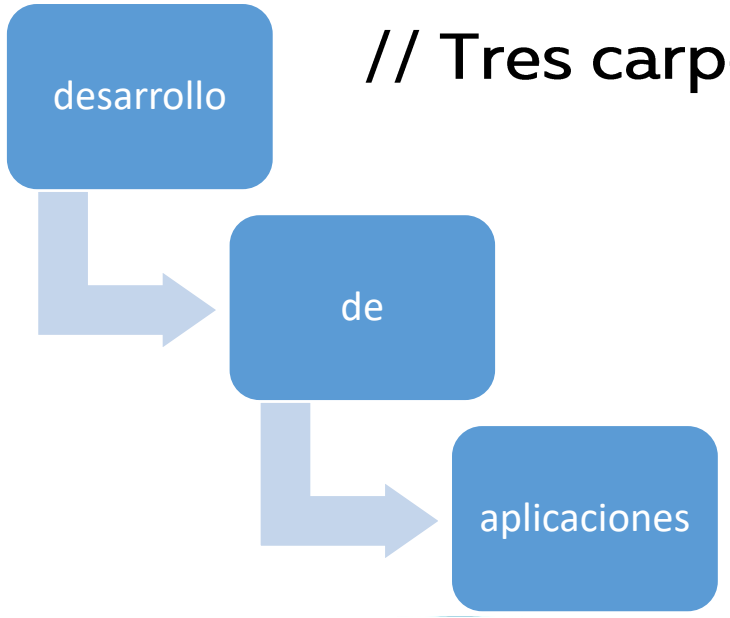
Paquetes

- ✓ Un paquete es un contenedor de clases. Se utiliza para ordenar el código de forma consistente de acuerdo a las funcionalidades implementadas.
- ✓ Los paquetes se almacenan de modo jerárquico, a través de carpetas y sub carpetas.
- ✓ Los paquetes se declaran utilizando la palabra reservada “package” seguida del nombre del paquete y debe colocarse al inicio del archivo .java.
  - ➢ package nombredepaquete;
  - ➢ package nombre.de.paquete;
- ✓ A los paquetes y sus clases se accede utilizando la palabra reservada “import”.
  - ➢ import nombredepaquete;
  - ➢ import nombredepaquete.nombredeclase;
  - ➢ import nombredepaquete.*;

Paquetes

idat

- ✓ Ejemplos:
  - ➢ package mipaquete; // Una carpeta
  - ➢ package desarrollo.de.aplicaciones; // Tres carpetas
  - ➢ import mipaquete;
  - ➢ Import mipaquete.miclase;
  - ➢ Import mipaquete.*;



Aplicación

Revisar ejemplos y realizar ejercicios prácticos

idat

Término

Indicaciones generales y/o Resumen de Sesión

idat

Resumen de Sesión

idat

- Clases y Objetos
- Atributos y Operaciones
- Referencia this
- Modificadores de Clases y Variables
- Paquetes
- Ejemplos y ejercicios

GRACIAS