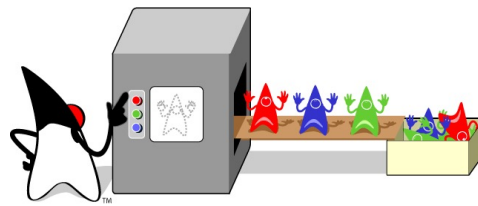


Tipos abstractos de datos

Creando nuevas clases



Facultad de Estudios Superiores
Acatlán
Programa de Matemáticas Aplicadas y Computación

2

Actividad

Coloca los datos de tu perro, o de uno que conozcas en tu cuaderno.

Nombre:

Raza:

Edad:

Tamaño:

Nota: Para tamaño escribir pequeño, mediano o grande

Carmen Villar / René Martínez

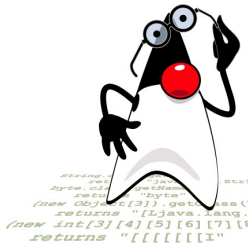
Facultad de Estudios Superiores
Acatlán
Programa de Matemáticas Aplicadas y Computación

4

4

Tipos de datos

- El tipo de dato de un objeto, determina el espacio que éste ocupa en memoria y cómo usarlo.



- Primitivos
 - Enteros
 - Flotantes
 - Caracter
 - Lógicos
- Referencias a objetos
 - Arreglos o vectores
 - Cadenas de caracteres
 - Entrada/Salida
 - Excepciones
 - Archivos

✓ Con objetos podemos crear nuestros propios Tipos Abstractos de Datos (TAD/ADT)

... *veamos algo de objetos*

**Facultad de Estudios Superiores
Acatlán**
Programa de Matemáticas Aplicadas y Computación

Carmen Villar / René Martínez

6

6

Diseño orientado a objetos (DOO)

- Los humanos pensamos en términos de objetos
 - Aprendemos acerca de ellos:
 - Estudiando sus atributos
 - tamaño, forma, color, peso, etc.
 - Observando comportamiento
 - Perro: ladra, camina, corre, salta, muerde
 - Foco: Prender, apagar, fundir
- El DOO modela los componentes de software, de igual forma que describimos los objetos del mundo real:
 - Por sus atributos
 - Su comportamiento
 - Las relaciones entre ellos

**Facultad de Estudios Superiores
Acatlán**
Programa de Matemáticas Aplicadas a la Ingeniería

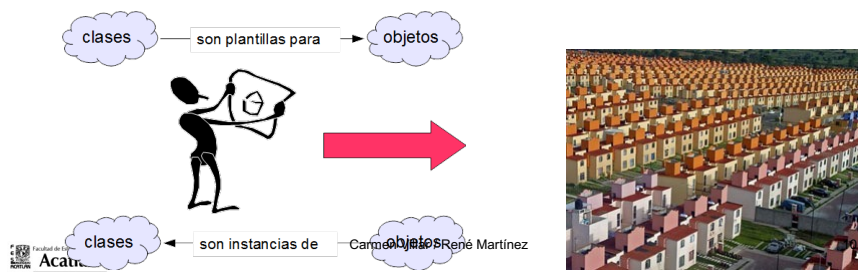
Carmen Villar / René Martínez

8

8

Relación entre clases y objetos

- Son los dos conceptos más importantes de la Programación Orientada a Objetos(POO)
 - Las clases son para los objetos lo que los planos de construcción son para las casas
 - Podemos construir muchas casas a partir de un plano
 - Podemos instanciar muchos objetos a partir de una clase



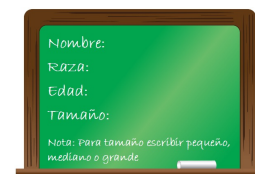
10

Clase

- Elemento de software que describe un **tipo abstracto de dato** y su **implementación**
 - Define nuevos tipos de datos
- Plantilla(template) para un objeto
 - Describe los datos que cada objeto debe incluir
 - Describe el comportamiento de cada objeto debe exhibir

```

class Perro{
    tipo atributo1
    :
    tipo atributoN
    constructor (lista parámetros)
    tipo método1 (lista parámetros)
    :
    tipo métodoN (lista parámetros)
}
  
```



12

Tipo abstracto de dato

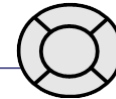
□ Definición.

- Clase de objetos cuyo comportamiento lógico es definido por un conjunto de valores y un conjunto de operaciones.
 - Análogo a una estructura algebraica en matemáticas.

□ Son conceptos teóricos en computación.

- Diseño y análisis de algoritmos, estructuras de datos, sistemas de software.

Objeto



- Modelan objetos de la realidad
- Es una instancia (un ejemplar, un caso concreto) de una sola clase
 - Objeto e instancia se usan como sinónimos

□ Estructura

- Estado
 - atributos/variables de instancia
 - Valores que lo hacen único

□ Comportamiento

- operaciones/métodos/servicios
- Invocar métodos



objeto = estado + comportamiento

AppInventor y orientación a objetos

Paquetes (User Interface, Layout, Media, Drawing and Animation, Sensors, Social, Storage, Connectivity, LEGO MINDSTORMS)

Clases (Programa de usuario (java main class))

Atributos (Objetos o Instancias)

Valores iniciales (constructor)

Carmen Villar / René Martínez

18

Los métodos (comportamiento)

Canvas1

- call Canvas1 - Clear
- call Canvas1 - DrawCircle
 - x
 - y
 - r
- call Canvas1 - DrawLine
 - x1
 - y1
 - x2
 - y2
- call Canvas1 - DrawPoint
 - x
 - y
- call Canvas1 - DrawText
 - text
 - x
 - y
- call Canvas1 - DrawTextAtAngle
 - text
 - x
 - y
 - angle
- call Canvas1 - GetBackgroundPixelColor
 - x
 - y
- call Canvas1 - GetPixelColor
 - x
 - y
- call Canvas1 - Save
- call Canvas1 - SaveAs
 - fileName
- call Canvas1 - SetBackgroundPixelColor
 - x
 - y

ImageSprite

- call Mazo - Bounce
 - edge
- call Mazo - CollidingWith
 - other
- call Mazo - MoveIntoBounds
- call Mazo - MoveTo
 - x
 - y
- call Mazo - PointInDirection
 - x
 - y
- call Mazo - PointTowards
 - target

Carmen Villar / René Martínez

20

Sintaxis clase y atributos

- La clase es la estructura lógica y la base de la POO sobre la cual Java está construido
 - Es una unidad modular reusable
 - Los subprogramas forman parte de una clase, no existen de manera independiente
- Sintaxis básica en Java de una clase

```
[modificador] class <identificador>
{
    [declaración_atributos]
    [declaración_constructores]
    [declaración_métodos]
}
```

- Sintaxis básica de un atributo

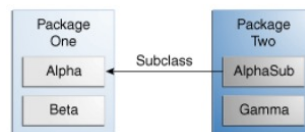
```
<modificador> <tipo> <identificador>[=<valor_inicial>];
```

22

Modificadores a nivel miembro

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N



Classes and Packages of the Example Used to Illustrate Access Levels

Visibility

Modifier	Alpha	Beta	Alphasub	Gamma
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

24

Cómo sería la clase perro

```
public class Perro {
    private String nombre;
    private String raza;
    private int edad;
    private char tamano;

    public Perro(String n, String r, int e, char t){
        nombre = n;
        raza = r;
        edad = e;
        tamano = t;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getRaza() {
        return raza;
    }

    public void setRaza(String raza) {
        this.raza = raza;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public char getTamano() {
        return tamano;
    }

    public void setTamano(char tamano) {
        this.tamano = tamano;
    }

    public void imprimir(){
        System.out.println("Nombre: "+nombre+"\nRaza: "+raza);
        System.out.println("Edad: "+edad+"\nTamaño: "+tamano);
    }
}
```

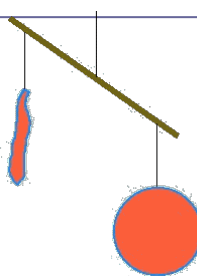
Carmen Villar / René Martínez

26

26

Creando un nuevo tipo de dato

```
public class Globo {
    private int radio=0;
    boolean inflable = true;
    public void inflar(int r){
        if(inflable) radio=r;
    }
    public void ponchar() {
        radio = 0;
        inflable = false;
    }
    public int medir(){
        return radio;
    }
}
```



```
class NombreClase {
    tipo atributo1
    :
    tipo atributoN
    constructor (lista parámetros)
    tipo metodo1 (lista parámetros)
    :
    tipo metodoN (lista parámetros)
}
```

Carmen Villar / René Martínez

Facultad de Estudios Superiores
Acatlán
Programa de Matemáticas Aplicadas y Computación

28

Constructor

- Toda clase en Java tiene *al menos* un constructor
 - Su nombre es igual al nombre de la clase
 - Sirve para dar valores iniciales a un objeto

```
public class Globo {
    private int radio;
    boolean inflable;
    public Globo() {
        radio = 0;
        inflable = true;
    }
    public void inflar(int r){
        if (inflable) radio=r;
    }
    public void ponchar() {
        radio = 0;
        inflable = false;
    }
    public int medir(){ return radio; }
}
```

Escuela Superior de Ingeniería y Tecnología
Acatlán
Programa de Ingeniería en Software y Computación

Carmen Villar / René Martínez

30

30

Constructor por defecto

- Si no se define, Java crea un constructor sin argumentos, ni inicialización
 - Lo correcto es tener siempre al menos uno
 - Se puede tener más de un constructor

```
public class Globo {
    private int radio;
    boolean inflable;
    public Globo() {
        radio = 0;
        inflable = true;
    }
    ...
}
```

El constructor siempre se llama como la clase y nunca regresa un tipo, ni siquiera void, por lo tanto no contiene un return

```
Globo rojo, azul;
rojo= new Globo();
```

...

Carmen Villar / René Martínez

32

32

Usando la clase

```
public class CrearGlobo {
    public static void main(String[] args){
        Globo a;
        a = new Globo();
        a.inflar(10);
        a.ponchar();
        System.out.println("El radio del globo
                           es "+a.medir());
    }
}
```

Java Class

Java Main Class



Carmen Villar / René Martínez



34

Ejercicio constructores

1. Crea una clase llamada **Reloj**, que almacene la hora.

Atributos: hh, mm y ss

Métodos constructores, reciben datos enteros:

1. No recibe valores: Marcaría la hora 00:00:00
2. Recibe un valor, el correspondiente a la hora hh:00:00
3. Recibe dos valores : hora y minutos hh:mm:00
4. Recibe tres valores : hora, minutos y segundos hh:mm:ss



Carmen Villar / René Martínez

36

36

Ejercicio constructores (cont)

2. Crea una clase llamada `Punto`, formado por coordenadas `double (x, y)` de un punto.

Métodos constructores a usar:

- a) No recibe valores: $\Rightarrow (0, 0)$
- b) Recibe dos valores `double`
- c) Recibe un arreglo de dos valores `double`

38

UML

- Lenguaje Unificado de Modelado
- Herramienta gráfica, para el análisis y diseño de aplicaciones orientadas a objetos
- Permite dar una notación básica a objetos, clases y relaciones de herencia



40

Diagrama UML de una Clase

■ Atributos

[visibilidad] nombre[:tipo [= default]]

■ Métodos

[visibilidad] nombre([parametro:tipo
[, lista parametros]])[:tipo_regresado]

■ Visibilidad

- + público Para todos
- - privado Clase
- ~ paquete Paquete (default)
- # protegido Clase y clases derivadas

Ejemplo: diagrama UML de una clase

```
public class Globo {
    private int radio=0;
    boolean inflable = true;
    public Globo() {
        radio = 0;
        inflable = true;
    }
    public void inflar(int r){
        if (inflable) radio=r;
    }
    public void ponchar() {
        radio = 0;
        inflable = false;
    }
    public int tamaño(){
        return radio;
    }
}
```

Globo
- radio : int=0 ~ inflable: boolean=true
+ Globo() + inflar (r:int) + ponchar() + tamaño():int

```

public class Prueba_abecedario {
    public static void main(String[] args)
    {
        Abecedario abc = new Abecedario();
        abc.imprime();
        System.out.println();
        abc.reversa();
        System.out.println();
    }
}

```

run:

```

ABCDEF GHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
BUILD SUCCESSFUL (total time: 0 seconds)

```

```

public class Abecedario {
    public final int LETRAS = 26;
    char[] s = new char[LETRAS];
    public Abecedario() {
        for( int i=0; i<LETRAS; i++ )
            s[i] = (char) ('A' + i);
    }
    public void imprime(){
        for( char c : s )
            System.out.print(c);
    }
    public void reversa(){
        for(int i=s.length-1; i>=0; i--)
            System.out.print(s[i]);
    }
}

```

46

Ejercicio: Identifica: clases, atributos, constructores y métodos.

```

public class Prueba_abecedario {
    public static void main(String[] args)
    {
        Abecedario abc = new Abecedario();
        abc.imprime();
        System.out.println();
        abc.reversa();
        System.out.println();
    }
}

```

🔴 Hacer diagrama UML

run:

```

ABCDEF GHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
BUILD SUCCESSFUL (total time: 0 seconds)

```

```

public class Abecedario {
    public final int LETRAS = 26;
    char[] s = new char[LETRAS];
    public Abecedario() {
        for( int i=0; i<LETRAS; i++ )
            s[i] = (char) ('A' + i);
    }
    public void imprime(){
        for( char c : s )
            System.out.print(c);
    }
    public void reversa(){
        for(int i=s.length-1; i>=0; i--)
            System.out.print(s[i]);
    }
}

```

48

Ejercicio UML: ¿Cuál sería el diagrama UML de este código?

```
public class PilaInt {
    private final int MAXPILA = 25;
    private int cima;
    private int [] listaPila;
    public PilaInt(){
        cima = -1;
        listaPila = new int[MAXPILA];
    }
    public boolean pilaVacía(){
        return cima == -1;
    }
    public boolean pilaLlena(){
        return cima == MAXPILA-1;
    }
    public void push(int elemento) {
        cima ++;
        listaPila[cima]=elemento;
    }
    public int pop() {
        int aux;
        aux=listaPila[cima];
        cima --;
        return aux;
    }
}
```

50

Crea en NetBeans la clase del diagrama UML

Foco

```
+ vatios : int
- prendido : boolean
~ fundido : boolean

+ Foco(w:int,estado:boolean )
+ Foco()
+ fundir()
+ apagar()
+ prender
+ calcularConsumo(horas:double):double
+ estaPrendido():boolean
```

52