Programación Java

Tutorial Java. Aprende a programar con Java desde cero.

Objetos en Java

Un objeto es una instancia de una clase. En un programa el objeto se representa mediante una variable. Esta variable contiene la dirección de memoria del objeto.

Cuando se dice que Java no tiene punteros estamos diciendo que Java no tiene punteros que podamos ver y manejar como tales en otros lenguajes como C/C++, pero debemos saber que todas las referencias a un objeto son de hecho punteros, es decir, son variables que contienen la dirección de memoria del objeto que representan.

Para crear un objeto se deben realizar dos operaciones:

- Declaración
- Instanciación

Declaración de un objeto

En la declaración se crea la referencia al objeto, de forma similar a cómo se declara una variable de un tipo primitivo.

La referencia se utiliza para manejar el objeto.

La sintaxis general para declarar un obejto en Java es:

```
NombreClase referenciaObjeto;
```

Por ejemplo, para crear un objeto de la clase Persona creamos su referencia así:

```
Persona p;
```

La referencia tiene como misión almacenar la dirección de memoria del objeto. En este momento la referencia p almacena una dirección de memoria nula (null).

Instanciación de un objeto

Mediante la instanciación de un objeto se reserva un bloque de memoria para almacenar todos los atributos del objeto.

Al instanciar un objeto solo se reserva memoria para sus atributos. No se guardan los métodos para cada objeto. Los métodos son los mismos y los comparten todos los objetos de la clase.

La dirección de memoria donde se encuentra el objeto se asigna a la referencia.

De forma general un objeto se instrancia en Java así:

```
eferenciaObjeto = new NombreClase();
```

new es el **operador Java para crear objetos**. Mediante new se asigna la memoria necesaria para ubicar el objeto y devuelve la dirección de memoria donde empieza el bloque asignado al objeto.

Por ejemplo:

```
p = new Persona();
```

Las dos operaciones pueden realizarse en la misma línea de código:

```
NombreClase referenciaObjeto = new NombreClase();
```

Por ejemplo:

PUBLICACIONES DEL BLOG

JAVA - Ejercicios básicos resueltos

ENTRADAS POPULARES

```
Persona p = new Persona();
```

Se ha creado un objeto persona y se ha asignado su dirección a la variable p.

Utilización

Una vez creado manejaremos el objeto a través de su referencia.

En general, el acceso a los atributos se realiza a través del operador punto, que separa al identificador de la referencia del identificador del atributo:

referenciaObjeto.Atributo;

Las llamadas a los métodos para realizar las distintas acciones se llevan a cabo separando los identificadores de la referencia y del método correspondiente con el operador punto:

```
referenciaObjeto.metodo([parámetros]);
```

Por ejemplo:

Para asignar valores a los atributos del objeto p lo debemos hacer a través de sus métodos ya que nombre y edad son private:

```
p.setNombre("Alonso");
p.setEdad(20);
```

Si intentamos hacer algo así:

```
p.edad = 20;
```

el compilador nos avisará del error ya que intentamos acceder de forma directa a un atributo privado.

La utilización del modificador **private** sirve para implementar una de las características de la programación orientada a objetos: el ocultamiento de la información o **encapsulación**.

La declaración como público de un atributo de una clase no respeta este principio de ocultación de información. Declarándolos como privados, no se tiene acceso directo a los atributos del objeto fuera del código de la clase correspondiente y sólo puede accederse a ellos de forma indirecta a través de los métodos proporcionados por la propia clase.

Una de las ventajas prácticas de obligar al empleo de un método para modificar el valor de un atributo es asegurar la consistencia de la operación.

Por ejemplo, el método setEdad de la clase Persona lo podemos escribir para evitar que se asigne que asignen valores no permitidos para el atributo edad:

```
public void setEdad(int ed) {
        if(ed > 0){
          edad = ed;
        }else{
          edad = 0;
        }
}
```

Con este método, una instrucción p.setEdad(-20); asegura que no se asignará a edad un valor no válido.

Si el atributo edad fuese público podemos escribir p.edad = -20; provocando que edad contenga un valor no válido.

Podemos crear tantos objetos como sean necesarios:

```
Persona q = new Persona();  //Crea otro objeto persona
```

En una asignación del tipo:

q = p;
no se copian los valores de los atributos, sino que se tiene como resultado una única instancia apuntada por dos referencias distintas

El objeto referenciado previamente por q se queda sin referencia (inaccesible).

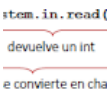
El **recolector de basura** de Java elimina automáticamente las instancias cuando detecta que no se van a usar más (cuando dejan de estar referenciadas).

Java Ejercicios Básicos Resueltos 1
Relación Nº 1: Ejercicios 1, 2 y 3 Empezaremos por unos ejercicios básicos de programas Java con estructura secuencial, es decir, en es...

Java printf para dar formato a los datos de salida
Vamos a ver como utilizar printf para dar formato a los datos que se imprimen por pantalla en Java. Este problema se nos plantea por ejempl...

Calcular el factorial de un número en Java
Factorial en java Programa que calcule el factorial de un número entero que se introduce por teclado. El factorial de un número se expresa m...

Ejercicio Básico POO Java. Clase Cuenta
Ejercicio básico de programación orientada a objetos en Java Escribe una clase Cuenta para representar una cuenta bancaria. Los datos de l...



Leer un char por teclado en Java
Cómo leer un carácter desde teclado en Java La clase Scanner NO CONTIENE un método nextChar() para leer un dato de tipo char desde tecl...

Estructuras de control en Java
Las estructuras de control determinan la secuencia de ejecución de las sentencias de un programa. Los programas contienen instrucciones...

Otro ejemplo: Vamos a diseñar una clase que almacene una fecha, comprobando que sea correcta. La definición de la clase con sus métodos podría ser:

```
public class Fecha {
    private int dia;
    private int mes;
    private int año;

    private boolean esBisiesto() {
        return ((año % 4 == 0) && (año % 100 != 0) || (año % 400 == 0));
    }

    public void setDia(int d) {
        dia = d;
    }

    public void setMes(int m) {
        mes = m;
    }

    public void setAño(int a) {
        año = a;
    }

    public void asignarFecha(int d, int m, int a) {
        setDia(d);
        setMes(m);
        setAño(a);
    }

    public int getDia() {
        return dia;
    }

    public int getMes() {
        return mes;
    }

    public int getAño() {
        return año;
    }

    public boolean fechaCorrecta() {
        boolean diaCorrecto, mesCorrecto, anyoCorrecto;
        anyoCorrecto = (año > 0);
        mesCorrecto = (mes >= 1) && (mes <= 12);
        switch (mes) {
            case 2:
                if (esBisiesto()) {
                    diaCorrecto = (dia >= 1 && dia <= 29);
                } else {
                    diaCorrecto = (dia >= 1 && dia <= 28);
                }
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                diaCorrecto = (dia >= 1 && dia <= 30);
                break;
            default:
                diaCorrecto = (dia >= 1 && dia <= 31);
        }
        return diaCorrecto && mesCorrecto && anyoCorrecto;
    }
}
```

Un método main de un proyecto donde usar la clase podría ser:

```
public class Principal {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Fecha fecha = new Fecha();
        int d, m, a;
        System.out.println("Introduce fecha: ");
        System.out.println("dia: ");
        d = sc.nextInt();
        System.out.println("mes: ");
        m = sc.nextInt();
        System.out.println("año: ");
        a = sc.nextInt();
        fecha.asignarFecha(d, m, a);
        if (fecha.fechaCorrecta()) {
            System.out.println(fecha.getDia() + "-" + fecha.getMes()+ "-" + fecha.getAño());
        } else {
            System.out.println("Fecha no valida");
        }
    }
}
```

Si te ha sido útil compártelo

Post

3 comentarios:

Carlos 5 de noviembre de 2013, 5:16

Que crees ya lo pude resolver lo que paso fue que tenia declaradas las dos clases dentro del mismo programa
public class Fecha {
private int dia;
private int mes;
.
.
.
public class Principal {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
Fecha fecha = new Fecha();
.
.
.
Elimine la declaracion de la clase Principal y me funciono; había olvidado que solo debe existir una clase pública ... Gracias por responder y aquí sigo aprendiendo

Responder

Carlos 29 de octubre de 2013, 5:34

Que tal, estoy aprendiendo Java de tu curso pero me atore en este tema ... lo que pasa es que me marca el siguiente error non-static variable this cannot be referenced from a static context y no he podido resolverlo. Me lo marca cuando creo el objeto de la clase (Fecha fecha = new Fecha()).
Además en el método asignarFecha en los set me marca error
setDia(d);
setMes(m);
setAño(a);
Ojala me pudieras ayudar ya que quiero seguir aprendiendo y tu lo explicas muy bien. Saludos y gracias

Responder

Respuestas

Responder

Enrique 4 de noviembre de 2013, 18:24

Hola Carlos, podrías subir el código que has escrito para poder ver donde está el error?
Un saludo y gracias por seguir el blog

Escribe tu comentario

Suscribirse a: Enviar comentarios (Atom)

LICENCIA