Constructores y creación de objetos

Una vez que se tiene definida la clase a partir de la cual se crearán los objetos se está en la posibilidad de instanciar los objetos requeridos.

Para la clase Usuario del ejemplo anterior podemos crear un objeto de la siguiente manera:

```
Usuario usr1;     //usr1 es una variable del tipo Usuario
usr1 = new Usuario();
```

La primera línea corresponde a la declaración del objeto, es decir, se declara una variable del tipo de objeto deseado.

La segunda línea corresponde a la iniciación del objeto.

- **El operador new**

  El operador **new** crea una instancia de una clase asignando la cantidad de memoria necesaria de acuerdo al tipo de objeto. El operador **new** se utiliza en conjunto con un *constructor*. El operador **new** regresa una referencia a un nuevo objeto.

- **Constructores**

  Un constructor es un tipo específico de método que siempre tiene el mismo nombre que la clase, y que se utiliza cuando se desean crear objetos de dicha clase, es decir, se utiliza al crear e iniciar un objeto de una clase.

- **Constructores múltiples**

  Cuando se declara una clase en Java, se pueden declarar uno o más constructores (*constructores múltiples*) opcionales que realizan la iniciación cuando se instancia un objeto de dicha clase.

Para la clase Usuario del ejemplo anterior no se especificó ningún constructor, sin embargo, Java proporciona un constructor por omisión que inicia las variables del objeto a sus valores predeterminados.

Ej.

```
/* ProgUsuario.java */

class ProgUsuario
{
```

```
  public static void main(String args[])
  {
    Usuario usr1, usr2;     /* Se declaran dos objetos de la clase Usuario */
    boolean si_no;

    usr1 = new Usuario();       /* Se utiliza el constructor por omisión */
    si_no = usr1 instanceof Usuario;

    if(si_no == true)
      System.out.println("\nEl objeto usr1 SI es instancia de Usuario.");
    else
      System.out.println("\nEl objeto usr1 NO es instancia de Usuario.");

    usr2 = usr1;                /* usr1 y usr2 son el mismo objeto */
    si_no = usr2 instanceof Usuario;

    if(si_no == true)
      System.out.println("\nEl objeto usr2 SI es instancia de Usuario.");
    else
      System.out.println("\nEl objeto usr2 NO es instancia de Usuario.");
   }
}
```

Constructor en Java

Un **constructor** es un método especial de una clase que se llama automáticamente siempre que se declara un objeto de esa clase.
Su función es inicializar el objeto y sirve para asegurarnos que los objetos siempre contengan valores válidos.

Cuando se crea un objeto en Java se realizan las siguientes operaciones de forma automática:

1. Se asigna memoria para el objeto.
2. Se inicializan los atributos de ese objeto con los valores predeterminados por el sistema.
3. Se llama al constructor de la clase que puede ser uno entre varios.

El constructor de una clase tiene las siguientes características:

Tiene el mismo nombre que la clase a la que pertenece.

En una clase puede haber varios constructores con el mismo nombre y distinto número de argumentos (se puede sobrecargar)

No se hereda.

No puede devolver ningún valor (incluyendo void).

Debe declararse público (salvo casos excepcionales) para que pueda ser invocado desde cualquier parte donde se desee crear un objeto de su clase.

MÉTODO CONSTRUCTOR POR DEFECTO

Si para una clase no se define ningún método constructor se crea uno automáticamente por defecto.

El **constructor por defecto** es un constructor sin parámetros que no hace nada. Los atributos del objeto son iniciados con los valores predeterminados por el sistema.

Por ejemplo, en la clase Fecha:

```
import java.util.*;

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() {
        Calendar fechaSistema = Calendar.getInstance();
        setDia(fechaSistema.get(Calendar.DAY_OF_MONTH));
        setMes(fechaSistema.get(Calendar.MONTH));
        setAño(fechaSistema.get(Calendar.YEAR));
    }

    public void asignarFecha(int d) {
        Calendar fechaSistema = Calendar.getInstance();
        setDia(d);
        setMes(fechaSistema.get(Calendar.MONTH));
        setAño(fechaSistema.get(Calendar.YEAR));
    }
```

```
    public void asignarFecha(int d, int m) {
        Calendar fechaSistema = Calendar.getInstance();
        setDia(d);
        setMes(m);
        setAño(fechaSistema.get(Calendar.YEAR));
    }

    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;
    }
}
```

no se ha definido ningún constructor, por lo que al declarar un objeto el compilador utilizará un constructor por defecto.
En este caso el método constructor por defecto es:
Fecha() { }
Vamos a añadir un constructor a la clase Fecha para poder iniciar los atributos de los nuevos objetos con valores determinados que se envían como parámetros. Si los valores corresponden a una fecha incorrecta se asigna la fecha del sistema:

```
public Fecha(int d, int m, int a) {
        dia = d;
        mes = m;
        año = a;

        if (!fechaCorrecta()) {
            Calendar fechaSistema = Calendar.getInstance();
            setDia(fechaSistema.get(Calendar.DAY_OF_MONTH));
            setMes(fechaSistema.get(Calendar.MONTH));
            setAño(fechaSistema.get(Calendar.YEAR));
        }
}
```

Con este método constructor podremos crear objetos de la siguiente manera:

Fecha fecha1 = new Fecha(10,2,2011);

Se crea un objeto fecha1 y se le asignan esos valores a sus atributos.

Si se crea un objeto con una fecha no válida:

Fecha fecha2 = new Fecha(10,20,2011);

A dia, mes y año se les asignará los valores del sistema.

Java crea un constructor por defecto si no hemos definido ninguno en la clase, pero si en una clase definimos un constructor ya no se crea automáticamente el constructor por defecto, por lo tanto si queremos usarlo deberemos escribirlo expresamente dentro de la clase.

En este ejemplo hemos definido un método constructor por lo que también es necesario poner el método constructor por defecto.

public Fecha(){}
Este constructor se invocará cuando se declare un objeto sin parámetros.
La clase tiene ahora dos constructores por lo que podemos crear objetos Fecha de dos formas:
Fecha f1 = new Fecha(); //se ejecuta el constructor sin parámetros
Fecha f2 = new Fecha(1,1,2010); //se ejecuta el constructor con parámetros

INVOCAR A UN MÉTODO CONSTRUCTOR

Un constructor se invoca cuando se crea un objeto de la clase mediante el operador new.
Si es necesario invocarlo en otras situaciones, la llamada a un constructor solo puede realizarse desde dentro de otro constructor de su misma clase y debe ser siempre la primera instrucción.
Si tenemos varios métodos constructores en la clase, podemos llamar a un constructor desde otro utilizando **this**.
Por ejemplo, si en el constructor con parámetros de la clase Fecha fuese necesario llamar al constructor sin parámetros de la misma clase escribimos:

```
public Fecha(int d, int m, int a) {
        this();  //llamada al constructor sin parámetros
        dia = d;
        ……
}
```

CONSTRUCTOR COPIA

Se puede crear un objeto a partir de otro de su misma clase escribiendo un constructor llamado **constructor copia**.
Este constructor copia los atributos de un objeto existente al objeto que se está creando.
Los constructores copia tiene un solo argumento, el cual es una referencia a un objeto de la misma clase que será desde el que queremos copiar.
Por ejemplo, para la clase Fecha podemos escribir un constructor copia que permita crear objetos con los valores de otro ya existente:

```
Fecha fecha = new Fecha(1,1,2011);   //se invoca al constructor con parámetros

Fecha fecha1 = new Fecha(fecha);  //se invoca al constructor copia
```

El constructor copia que escribimos para la clase es:

```
//constructor copia de la clase Fecha
public Fecha(final Fecha f) {
        dia = f.dia;
        mes = f.mes;
        año = f.año;
}
```

El constructor copia recibe una referencia al objeto a copiar y asigna sus atributos uno a uno a cada atributo del objeto creado.
Declarar el parámetro como final no es necesario pero protege al objeto a copiar de cualquier modificación accidental que se pudiera realizar sobre él.
Declarar y crear un objeto en Java

EJEMPLO Supongamos que, en un programa escrito en Java, se ha definido la siguiente clase (**Persona**):

```
public class Persona
{
   // Atributos de la clase Persona
   private String nombre;
```

```
    private int edad;

    // Métodos de la clase Persona
    public String getNombre()
    {
        return nombre;
    }

    public int getEdad()
    {
        return edad;
    }

    public void setNombre(String n)
    {
        nombre = n;
    }

    public void setEdad(int e)
    {
        edad = e;
    }
}
```

Obsérvese que, en el cuerpo de la clase –entre las llaves **{}**– se han definido:

- 2 atributos –también llamados campos– privados (**private**): **nombre** y **edad**.
- 4 métodos públicos (**public**): **getNombre**, **getEdad**, **setNombre** y **setEdad**.

De tal forma que, todos los objetos que se creen de la clase **Persona** tendrán un **nombre** y una **edad** que podrán almacenar valores distintos, pudiendo ser modificados o consultados cuando se invoque a sus métodos definidos:

- **setNombre** permite asignar –*set*– un nombre (**String**) a un objeto de la clase **Persona**.
- **setEdad** permite asignar –*set*– una edad (**int**) a un objeto de la clase **Persona**.

- **getNombre** sirve para consultar –*get*– el nombre de un objeto de la clase **Persona**.
- **getEdad** sirve para consultar –*get*– la edad de un objeto de la clase **Persona**.

Para declarar y crear un objeto de la clase **Persona** se puede escribir:

```
Persona p1; // Declaración de la variable p1 de tipo Persona
p1 = new Persona();  // Creación de un objeto de la clase Persona
```

Nota: con la declaración **Persona p1**, se está reservando un espacio de memoria para almacenar una referencia (dirección de memoria) a un objeto de la clase **Persona**. Al respecto, es importante comprender que **p1** no es un objeto, sino una variable que almacenará la referencia a un objeto (de la clase **Persona**) que todavía no existe. Seguidamente, mediante la sentecia **p1 = new Persona()**, el operador **new** creará un objeto de la clase **Persona**, reservando memoria para guardar sus atributos (**nombre** y **edad** en este caso). Finalmente, con el operador de asignación (**=**), la dirección de memoria donde esté creado el objeto, es asignada a **p1**.

También, se puede indicar lo mismo en una sola línea:

```
Persona p1 = new Persona();
```