**Referencias a Objeto e Instancias**

Cada vez que se crea una clase se añade otro tipo de dato que se puede utilizar igual que uno de los tipos simples vistos hasta el momento (entero, real, cadena, entre otros). Por lo tanto cuando se declara una nueva variable, se puede utilizar un nombre de clase para definir su tipo. A estas variables se les conoce como referencias a objeto. Ejemplo:

Pseudocódigo🡪MiVariable: MiClase

Java🡪MiClase MiVariable;

Esta es una declaración de una variable denominada ‘MiVariable’ que es una referencia a un objeto de la clase MiClase, de momento con un valor por defecto de null.

**Constructores**

Las clases pueden implementar un método especial llamado constructor. Un constructor es un método que inicia un objeto inmediatamente después de su creación. De esta forma, se evita la tarea tediosa de inicializar las variables de forma individual.

El constructor tiene exactamente el mismo nombre de la clase que lo implementa; no puede existir otro método que posea el mismo nombre de la clase. Una vez definido, se llamará automáticamente al constructor al crear un objeto de esa clase (al utilizar el operador new).

El constructor no devuelve ningún tipo. Su misión es inicializar todos los atributos del objeto, haciendo que sea utilizable inmediatamente, mediante la reservación de memoria para sus atributos y la inicialización de sus valores.

Por ejemplo si se tiene una clase MisNumeros cuyos atributos sean num1 y num2, ambos de tipo entero, su constructor podrá ser el siguiente:

MisNumeros( ) {

Num1=-1;

Num2=-1;

}

El constructor mostrado en el ejemplo anterior, se denomina constructor por defecto, ya que no posee parámetros, por lo tanto se establece el valor -1 a las variables de instancia num1 y num2 de los objetos que construya.

Existe otro constructor, como el que se muestra a continuación, el cual recibe dos parámetros:

MisNumeros( int numero1, int numero2 ) {

Num1=numero1;

Num2=numero2;

}

La lista de parámetros especificada después del nombre de una clase en una sentencia new se utiliza para pasar parámetros al constructor.

Se llama al método constructor justo después de crear la instancia y antes de que new devuelva el control al punto de la llamada.

Así, cuando ejecutamos el siguiente programa:

MisNumeros numeros = new MisNumeros(10, 20);

System.out.println( "Numero 1 = " + numeros.num1 + " Numero 2 = " + numeros.num2 );

Se muestra en la pantalla:

Numero 1 = 10 y Numero 2= 20

**El operador new**

El operador new crea una instancia de una clase (objetos) y devuelve una referencia a ese objeto. Por ejemplo:

Pseudocodigo 🡪n1🡨nuevo(MisNumeros)

Java🡪MisNumeros n1 = new MisNumeros();

Este es un ejemplo de la creación de una instancia de MisNumeros, que es controlador por la referencia a objeto n1.

Cuando ya no haya ninguna variable que haga referencia a un objeto, Java reclama automáticamente la memoria utilizada por ese objeto, a lo que se denomina recogida de basura.

Cuando se realiza una instancia de una clase (mediante new) se reserva en la memoria un espacio para un conjunto de datos como el que definen los atributos de la clase que se indica en la instanciación.

Es importante el comprender que cada objeto tiene su propia copia de las variables de instancia de su clase, por lo que los cambios sobre las variables de instancia de un objeto no tienen efecto sobre las variables de instancia de otro.

En el código presentado a continuación se crean dos objetos MisNumeros y se establecen los valores de num1 y num2 de cada uno de ellos de manera independiente para mostrar que están realmente separados.

MisNumeros n2 = new MisNumeros( 2, 11 );

MisNumeros n3 = new MisNumeros( 26, 24 );

System.out.println("n2.num1 = " + n2.num1 + " n2.num2 = " + n2.num2);

System.out.println("n3.num1 = " + n2.num1 + " n3.num2 = " + n3.num2);

La salida generada de la ejecución del código es la siguiente:

n2.num1 = 2 y n2.num2= 11

n3.num1 = 26 y n3.num2 = 24

**El operador punto (.)**

El operador punto (.) se utiliza para acceder a las variables de instancia y los métodos contenidos en un objeto, mediante su referencia a objeto:

referencia\_a\_objeto.nombre\_de\_variable\_de\_instancia

referencia\_a\_objeto.nombre\_de\_método( lista\_de\_parámetros )

**La destrucción de los objetos**

Cuando un objeto no va a ser utilizado, el espacio de memoria dinámica que utiliza ha de ser liberado, así como los recursos que poseía, permitiendo al programa disponer de todos los recursos posibles. A esta acción se la da el nombre de destrucción del objeto.

En Java la destrucción se puede realizar de forma automática ya que el intérprete de Java posee un sistema de recogida de basura, el cual se encarga de liberar una zona de memoria dinámica que había sido reservada mediante el operador new, cuando el objeto ya no va a ser utilizado más durante el programa. El sistema de recogida de basura se ejecuta periódicamente, buscando objetos que ya no estén referenciados.

**Métodos**

Son las funcionalidades asociadas a los objetos, es decir, son todas las actividades que se pueden realizar sobre un objeto. Los métodos se dividen en funciones y procedimientos.

Funciones: Tienen como característica principal el hecho de que retornan un valor.

Procedimientos: No retornan un valor, solo pueden modificar el valor de los parámetros por referencia que recibe y realizar operaciones internas.

**Paso de parámetros:**

Los parámetros son los datos que reciben los métodos y son utilizados para realizar las operaciones que en ellos se incluyen. Un método puede recibir cualquier número de parámetros, incluso ninguno. Los parámetros que un método recibe son indicados al definirlo, por ejemplo:

Método calcular\_area(altura, anchura: real)

Existen dos (2) formas de pasar parámetros, ellas son:

* ­Paso de parámetros por valor: Consiste en copiar el contenido de la variable que se desea pasar en otra, dicha copia es válida únicamente dentro del método invocado, por lo que todas las modificaciones que se le realicen a la variable dentro del método, no afectan el valor de la variable original pasada por parámetro.
* Paso de parámetros por referencia: Consiste en proporcionar al método invocado la dirección de memoria donde se almacena el contenido de la variable pasada por parámetro. En este caso cualquier modificación realizada a la variable dentro del método, si afecta el valor de la variable original pasada por parámetro.

Realizar un algoritmo en pseudocódigo y en lenguaje java que permita calcular el área de un rectángulo.

Rectangulo = **clase** de

{

alto, ancho= real

}

**Método** calcular\_area()

**Inicio**

area: entero

area🡨 ancho \* alto

Escribir “El área del rectángulo es de ”, area

**Fin metodo**

**Método** llenar\_datos()

**inicio**

Escribir “Introduzca la altura del rectángulo: ”

Leer alto

Escribir “Introduzca el ancho del rectángulo: “

Leer ancho

**Fin metodo**

}

**Algoritmo** area\_rectangulo

**Variables**

Rectan: Rectangulo

resp: char

**Inicio**

Rectan 🡨 nuevo (Rectangulo)

Rectan.llenar\_datos()

Escribir “Desea calcular el area del rectángulo?(s/n)”

Leer (resp)

**Si**( resp==’s’ o resp==’S’) **entonces**

Rectan.calcular\_area()

**Fin si**

**Fin**

**JAVA**

**Import** java.io.\*;

**Class** Rectangulo {

**Public Float** alto, ancho;

**Public void** calcular\_area()

{

**Int** area;

area= ancho \* alto;

**System.out.println** (“El área del rectángulo es de ”+area);

}

**Public void** llenar\_datos()

{

**BufferedReader** teclado = **New BufferedReader (new InputStreamReader(System.in));**

**System.out.print** (“Introduzca la altura del rectángulo: ”);

alto = **Float.parseFloat(teclado.readLine());**

**System.out.print** (“Introduzca el ancho del rectángulo: “);

ancho = **Float.parseFloat(teclado.readLine());**

}

}

**Public class** area\_rectangulo {

**Public static void main(string[] args) throws Exception{**

Rectangulo Rectan;

**Char** resp;

Rectan = new Rectangulo();

Rectan.llenar\_datos();

**BufferedReader** teclado = **New BufferedReader (new InputStreamReader(System.in));**

**System.out.println** (“Desea calcular el area del rectángulo?(s/n)”);

Resp = **teclado.readLine()**;

**if**( resp==’s’ || resp==’S’){

Rectan.calcular\_area();

}

}