APRENDERAPROGRAMAR.COM

EJEMPLO DE CÓDIGO JAVA BÁSICO. CREAR CLASES CON CAMPOS, CONSTRUCTOR Y MÉTODOS. LA PALABRA CLAVE THIS (CU00652B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº52 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

UN EJEMPLO DE CÓDIGO JAVA BÁSICO. CREAR CLASES CON CAMPOS, CONSTRUCTOR Y MÉTODOS

Para familiarizarnos con el código Java escribe, ejecuta y estudia el código que mostramos a continuación, correspondiente a dos clases. Todos los elementos que forman parte de él ya los hemos estudiado excepto la llamada *this (0, 0, “”)*. La palabra clave *this* tiene distintos usos en Java y en general podríamos interpretarla como “este objeto”.

La invocación *this*, o *this (parámetros)* supone una invocación al constructor que coincida con los parámetros que se pasan para que se ejecute. Al igual que existen formas de invocar a métodos, existen formas de invocar a constructores, y ésta es una de ellas.

El código de la primera clase sería el siguiente:

```
/* Esta clase representa un  depósito cilíndrico donde se almacena aceite  */

public class Deposito {

   //Campos de la clase

   private float diametro;

   private float altura;

   private String idDeposito;


   //Constructor sin parámetros auxiliar
   public Deposito () { //Lo que hace es llamar al constructor con parámetros pasándole valores vacíos
      this(0,0,"");          } //Cierre del constructor


   //Constructor de la clase que pide los parámetros necesarios
   public Deposito (float valor_diametro, float valor_altura, String valor_idDeposito) {
      if (valor_diametro > 0 && valor_altura > 0) {
         diametro = valor_diametro;
         altura = valor_altura;
         idDeposito = valor_idDeposito;
      } else {
         diametro = 10;
         altura = 5;
         idDeposito = "000";
         System.out.println ("Creado depósito con valores por defecto diametro 10 metros altura 5 metros id 000" );
      }  } //Cierre del constructor
```

```
    public void setValoresDeposito (String valor_idDeposito, float valor_diametro, float valor_altura) {
       idDeposito = valor_idDeposito;
       diametro = valor_diametro;
       altura = valor_altura;
       if (idDeposito !="" && valor_diametro > 0 && valor_altura > 0) {
       } else {
          System.out.println ("Valores no admisibles. No se han establecido valores para el depósito");
          //Deposito (0.0f, 0.0f, ""); Esto no es posible. Un constructor no es un método y por tanto no podemos llamarlo
          idDeposito = "";
          diametro = 0;
          altura = 0;
       }    } //Cierre del método

    public float getDiametro () { return diametro; } //Método de acceso
    public float getAltura () { return altura; } //Método de acceso
    public String getIdDeposito () { return idDeposito; } //Método de acceso
    public float valorCapacidad () { //Método tipo función
       float capacidad;
       float pi = 3.1416f; //Si no incluimos la f el compilador considera que 3.1416 es double
       capacidad = pi * (diametro/2) * (diametro/2) * altura;
       return capacidad;
}    } //Cierre de la clase
```

En el método setValoresDeposito nos encontramos un código un tanto extraño: un if donde las instrucciones a ejecutar se encuentran vacías. Esto es admitido en Java, tanto en un if como en un else o en otras instrucciones. En este caso, el código equivale a: “Si el idDeposito es distinto de una cadena vacía y el valor_diametro es mayor que cero y el valor_altura es mayor que cero no se hace nada, y en caso contrario se han de ejecutar las instrucciones indicadas en el else”. Este tipo de construcciones no consideramos conveniente utilizarlas frecuentemente. Tan solo pueden ser indicadas cuando queremos remarcar que en determinadas circunstancias no se debe ejecutar ninguna instrucción.

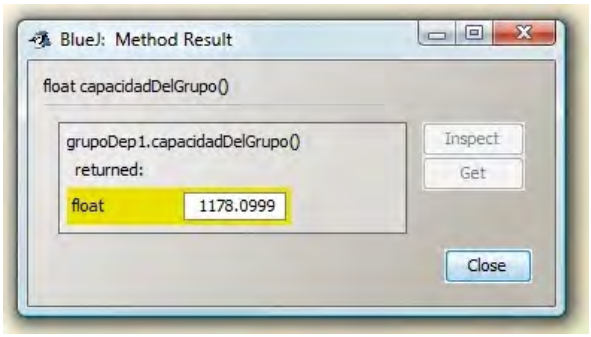
Otra cuestión a tener en cuenta es que de momento estamos desarrollando una programación informal: el sistema de comentarios no se atiene a lo establecido por el sistema de documentación de Java, y hemos incluido algunas sentencias de impresión por consola que normalmente no forman parte del código de los programas. Usaremos estas y otras técnicas informales con el fin de facilitar el aprendizaje, no porque puedan ser recomendadas como técnicas de programación.

La segunda clase sería la siguiente:

```
/*Esta clase  representa un conjunto de depósitos formado por entre 2 y 3 depósitos  */
public class GrupoDepositos {
   //Campos de la clase, algunos de ellos son tipo objetos de otra clase
   private Deposito deposito1;
   private Deposito deposito2;
   private Deposito deposito3;
   private String idGrupo;
   private int numeroDepositosGrupo;
```

```
        //Constructor para la clase. En ella se crean objetos de otra clase.
        public GrupoDepositos (int numeroDeDepositosGrupo, String valor_idGrupo) {
           idGrupo = valor_idGrupo;
           switch (numeroDeDepositosGrupo) {
              case 1: System.out.println ("Un grupo ha de tener más de un depósito"); break;

              case 2:
              deposito1 = new Deposito(); /*Al crear el objeto automáticamente se llama al constructor del mismo, en este
caso sin parámetros. ESTO ES EJEMPLO DE SINTAXIS DE CREACIÓN DE UN OBJETO, EN ESTE CASO DENTRO DE OTRO */
              deposito2 = new Deposito();
              numeroDepositosGrupo = 2;
              break;

              case 3: deposito1 = new Deposito(); deposito2 = new Deposito(); deposito3 = new Deposito();
              numeroDepositosGrupo = 3;
              break;

              default: System.out.println ("No se admiten más de tres depósitos");
              //Esto no evita que se cree el objeto.
              break;
           } //Cierre del switch
        } //Cierre del constructor

        public int getNumeroDepositosGrupo () { return numeroDepositosGrupo; }

        public String getIdGrupo () { return idGrupo; }
        public float capacidadDelGrupo () {      //Este método usa objetos de otra clase e invoca métodos de otra clase
           if (numeroDepositosGrupo == 2) { return (deposito1.valorCapacidad() + deposito2.valorCapacidad() );
           } else { return (deposito1.valorCapacidad() + deposito2.valorCapacidad()+ deposito3.valorCapacidad() ); }
           //Si el grupo se ha creado con un número de depósitos distinto de 2 o 3 saltará un error en tiempo de ejecución
        } //Cierre del método
} //Cierre de la clase
```

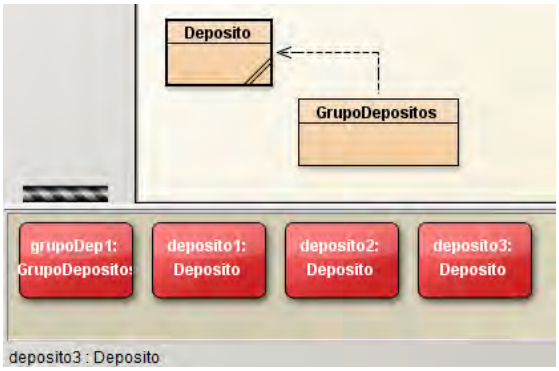
Con botón derecho sobre el icono de la clase GrupoDepositos, crea un grupo de depósitos que conste de 3 depósitos y cuyo idGrupo sea “Grupo KHP”. Invoca los métodos que devuelven el número de depósitos del grupo, el identificador del grupo y la capacidad de los depósitos del grupo. Como capacidad deberás obtener un valor de aproximadamente 1178.1 unidades cúbicas.



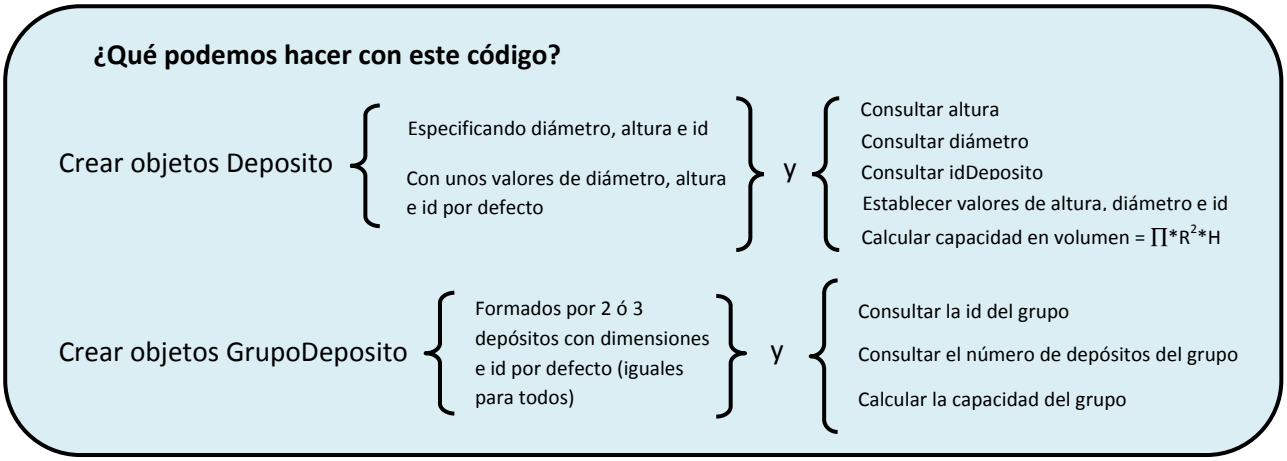
Verifica con la calculadora si este valor es correcto y trata de razonar sobre por qué se obtiene este valor y no otro. Crea también distintos objetos de tipo Deposito y utiliza sus métodos. En el caso de

resultados numéricos, comprueba si los resultados que te ofrece el ordenador son correctos comparándolos con los resultados que te ofrece una calculadora.

Este ejemplo de código, todavía muy elemental y rudimentario, tiene un diagrama de clases donde nos indica que la clase GrupoDepositos usa a la clase Deposito.



A modo de resumen, el siguiente esquema nos indica lo que podemos hacer con este código. De este ejemplo lo único que nos interesa es practicar cómo una clase puede usar objetos y métodos de otra clase y la sintaxis a emplear. Aunque no hemos creado ningún programa, estamos viendo cómo crear clases y objetos que intervendrán en los programas.



Próxima entrega: CU00653B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188