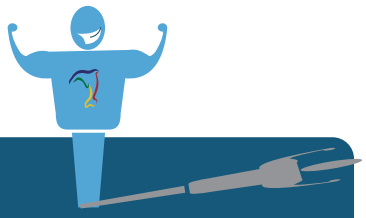
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

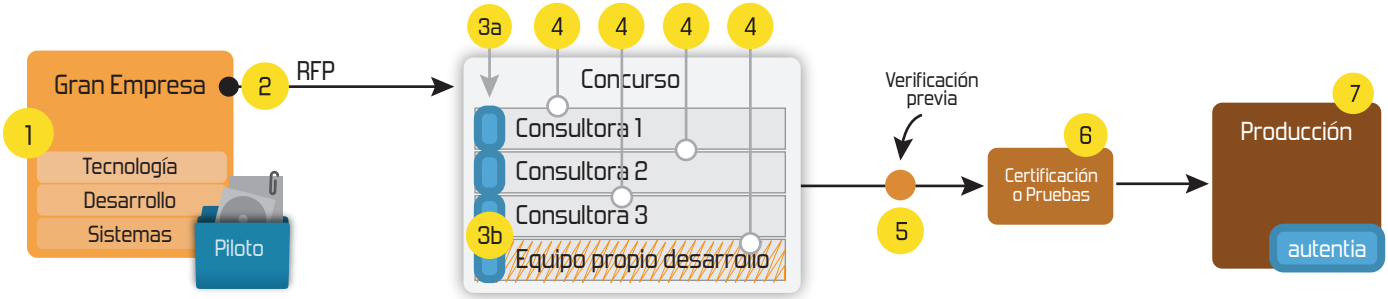
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por:
Roberto Canales Mora 2003-2005
Creador de AdictosAlTrabajo.com y

Director General de Autentia S.L.

Labores Básicas en Java

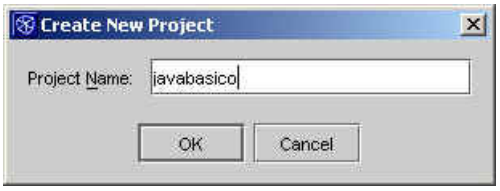
Cuando empezamos a desarrollar aplicaciones en Java podemos encontrar un montón de libros que nos explican los principios sobre el uso de variables, creación de clases, etc....

Normalmente en las aplicaciones necesitamos hacer ciertas cosas que no encontramos en muchos manuales y son bastante necesarias.

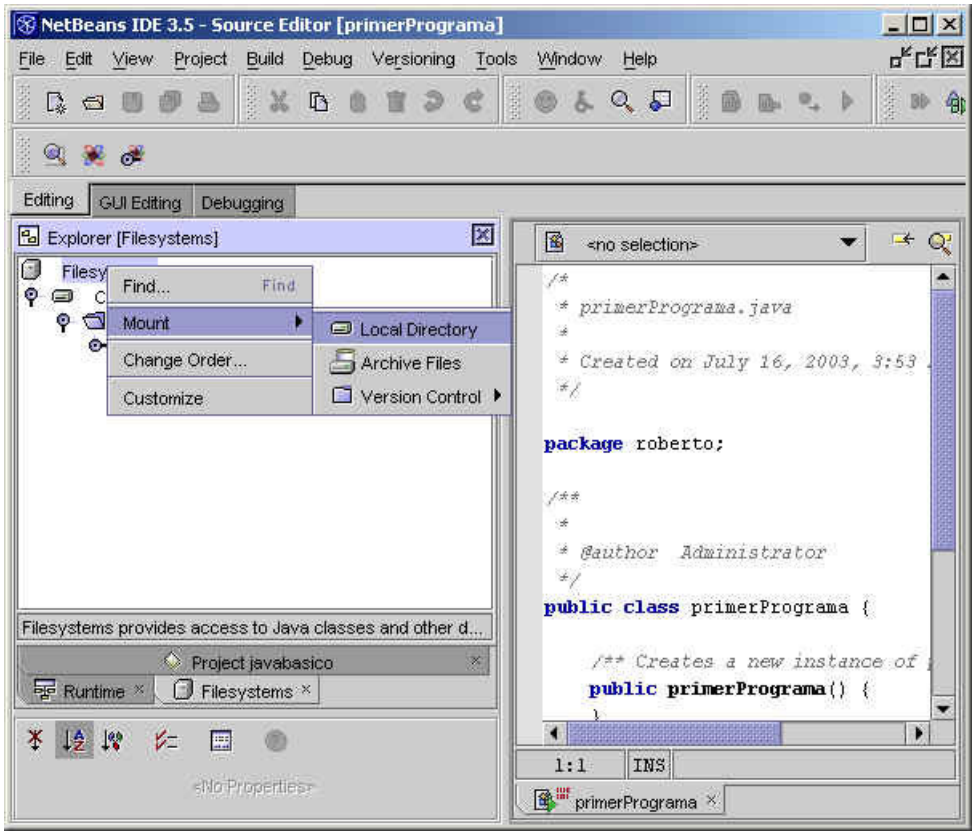
- Formatear decimales
- Mostrar enteros en formato Hexadecimal
- Escribir y leer las preferencias de nuestro usuario
- Capacidad de comparar objetos de nuestras clases

Vamos a mostraros como empezar a desarrollar aplicaciones Java con NetBeans y los ejemplos mencionados anteriormente.

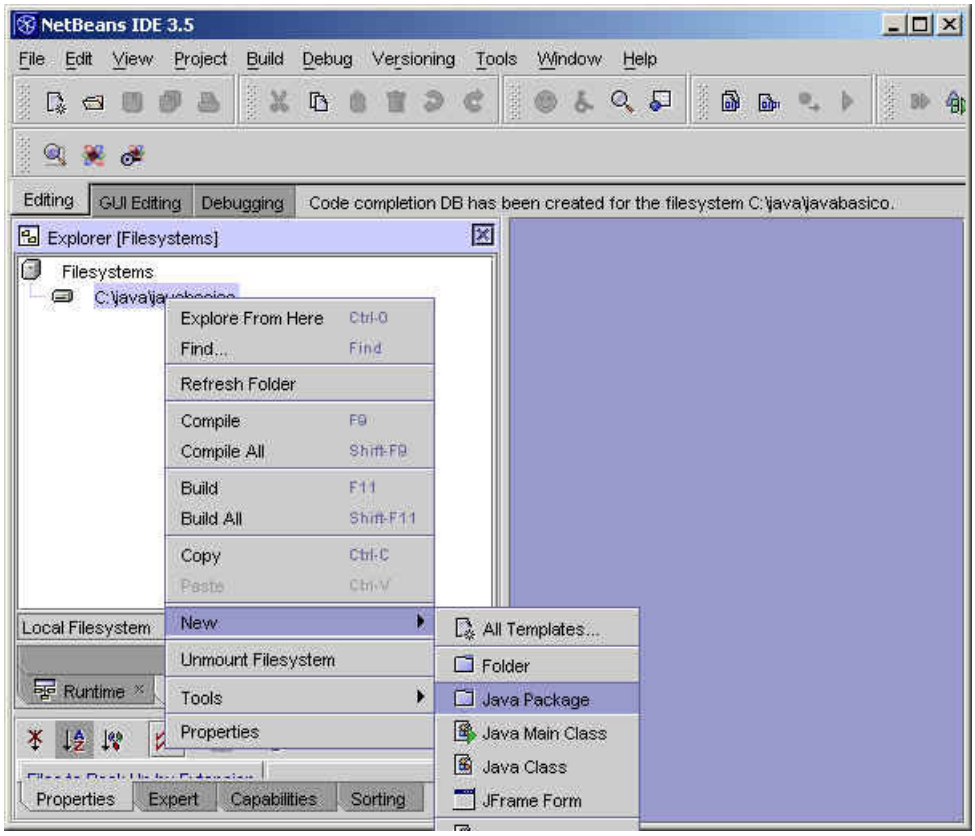
Creamos un proyecto



Seleccionamos el directorio de trabajo

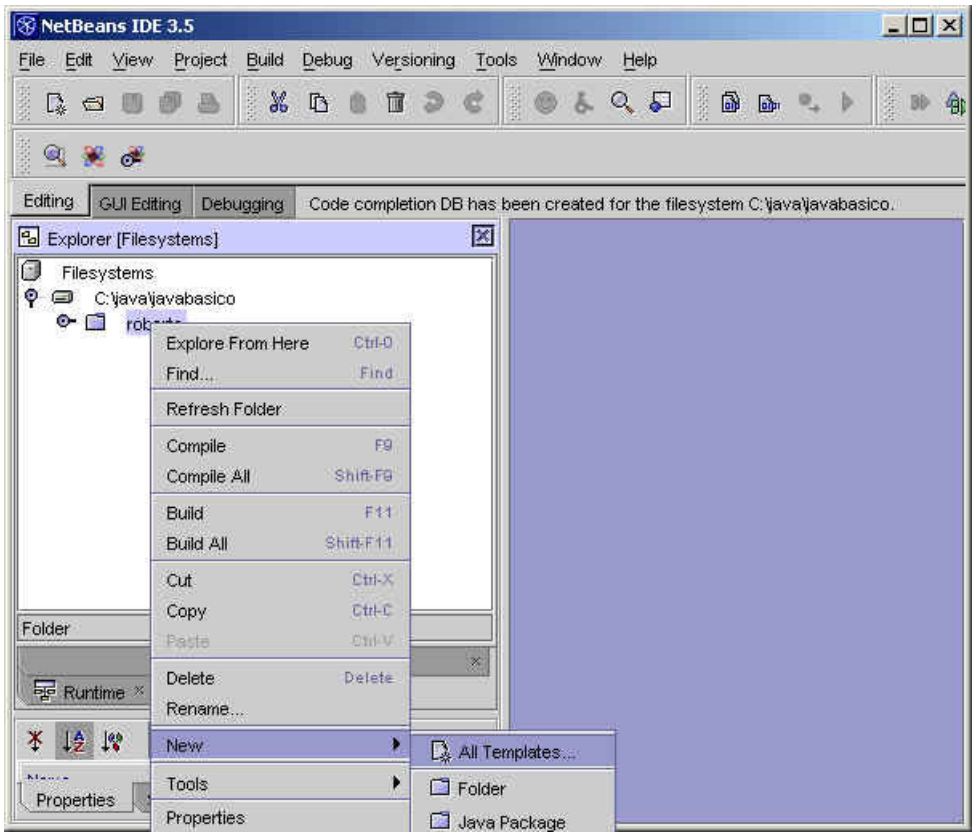


Creamos un nuevo paquete (un paquete nos vale para agrupar de un modo lógico nuestro código y que no entre en conflicto con el código desarrollado por otros equipos)

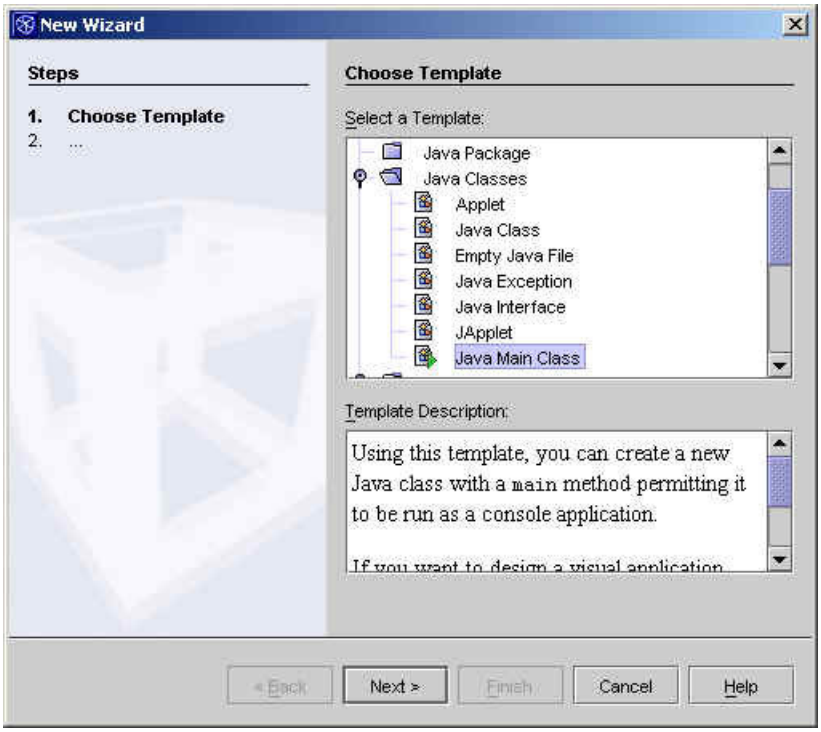


Damos nombre al paquete

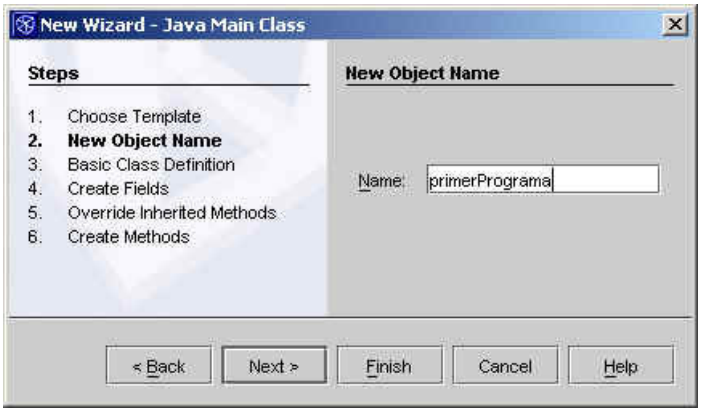
Añadimos un nuevo elemento, listando todos los disponibles



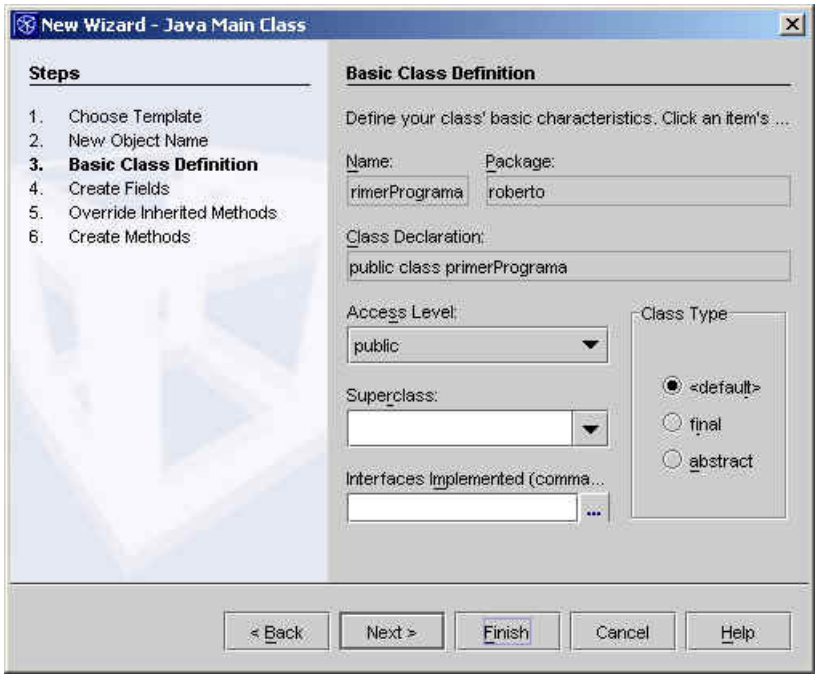
Seleccionamos una nueva clase principal



Le asignamos un nombre



Vemos la definición básica

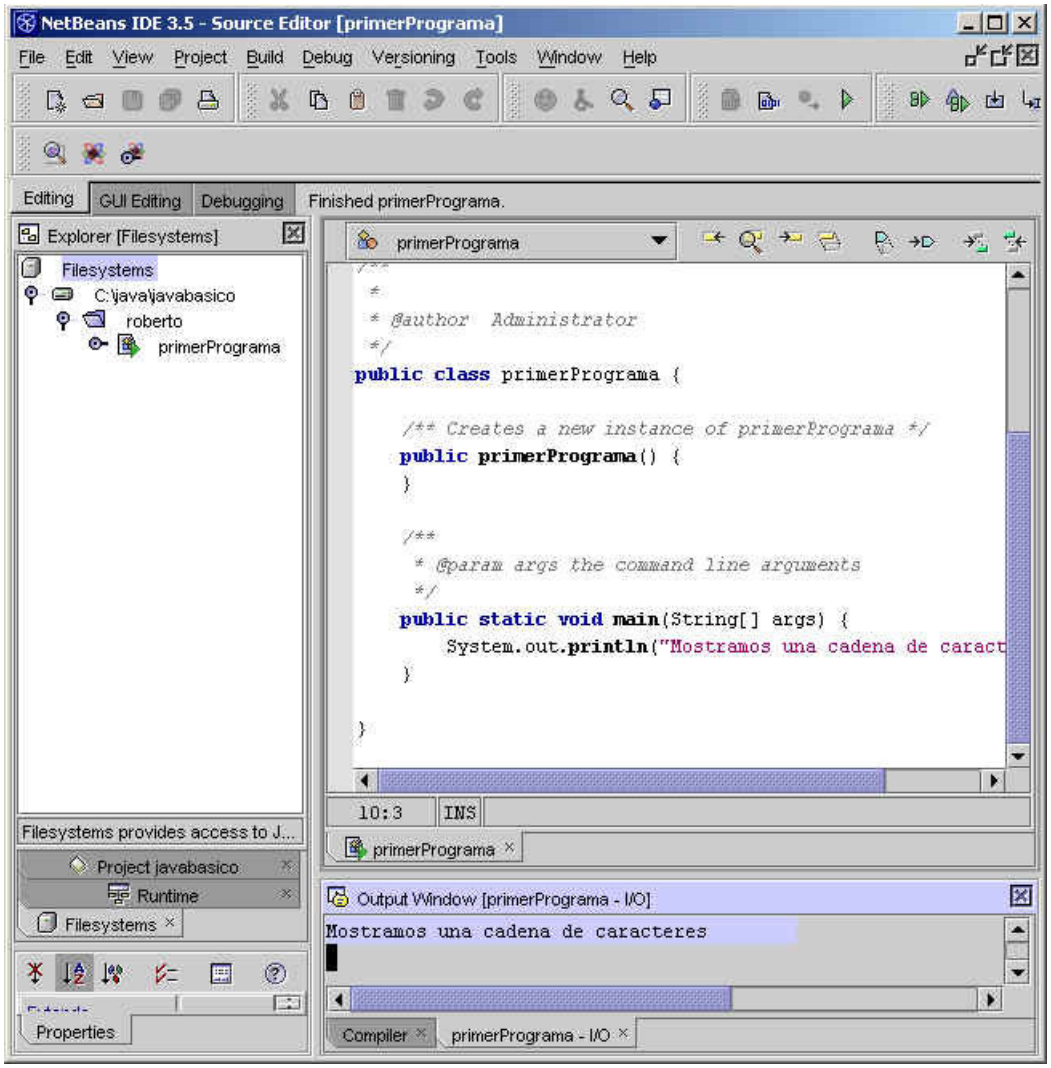


Y pulsamos Finalizar ... el resto de las opciones de momento no nos interesan.

Vemos el código generado y en azul, una línea para mostrar un mensaje por pantalla.

```
/*
 * primerPrograma.java
 *
 * Created on July 16, 2003, 3:53 AM
 */

package roberto;

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

    /** Creates a new instance of primerPrograma */
    public primerPrograma() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Mostramos una cadena de caracteres");


    }
}
```

Si pulsamos el botón ejecutar, vemos abajo y a la derecha .... el mensaje que hemos escrito.



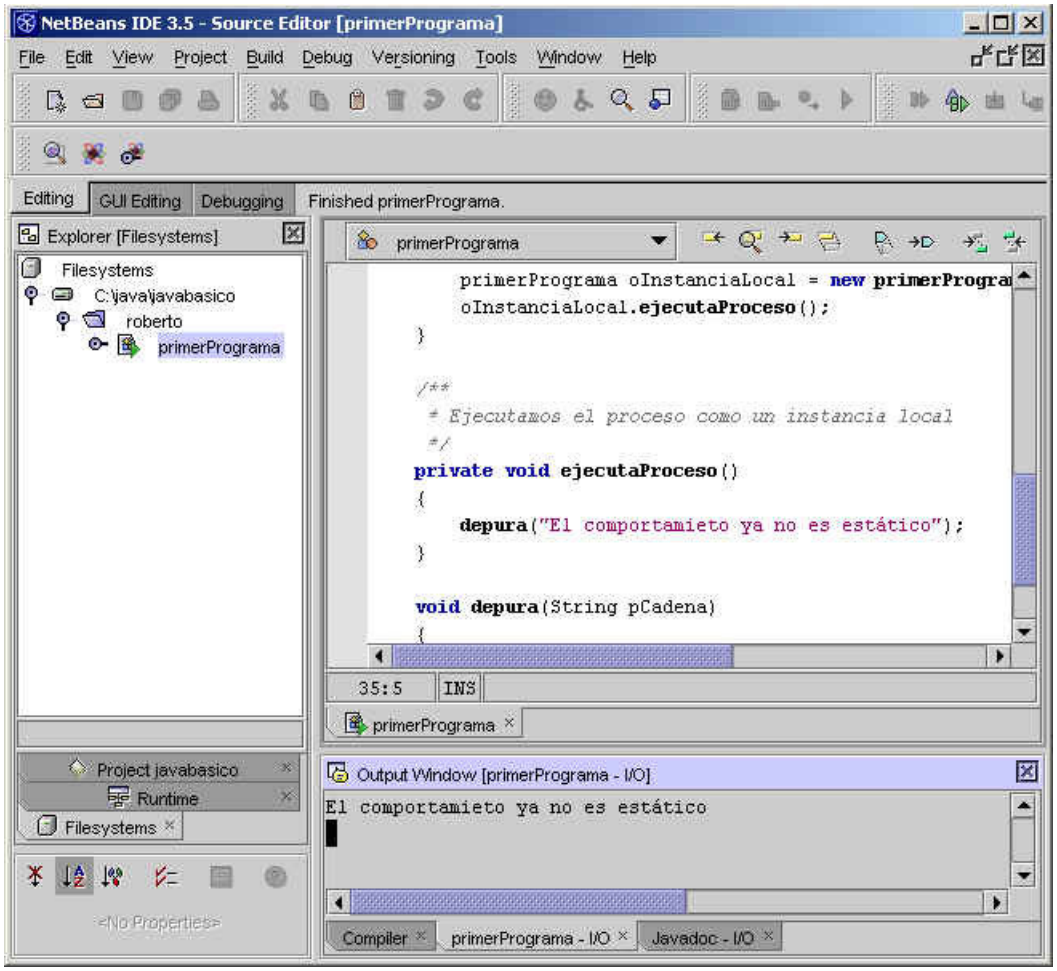
Referencia no Estáticas

La función de entrada, esta declarada como **static,** lo que significa que para que se comporte como una clase normal, debemos crear un objeto (del mismo tipo que la clase) .... que sea quien ejecute realmente el trabajo.

Podemos ver que también hemos creado una función llamada **depura**, donde podemos centralizar todas las salidas por pantalla.

```
package roberto;
/*
```

```
* primerPrograma.java
*
* Created on July 16, 2003, 3:53 AM
*/
import java.text.*;

/**
*
* @author  Administrator
*/
public class primerPrograma {

    /** Constructor por defecto de la clase */
    public primerPrograma() {
    }

    /**
     * Punto de entrada estático de la aplicación
     * @param args recibe los parámetros de línea de comando
     */
    public static void main(String[] args) {
        primerPrograma oInstanciaLocal = new primerPrograma();
        oInstanciaLocal.ejecutaProceso();
    }

    /**
     * Ejecutamos el proceso como un instancia local
     */
    private void ejecutaProceso()
    {
        depura("El comportamieto ya no es estático");
    }

    void depura(String pCadena)
    {
        System.out.println(pCadena);
    }
}
```



Formatear números decimales

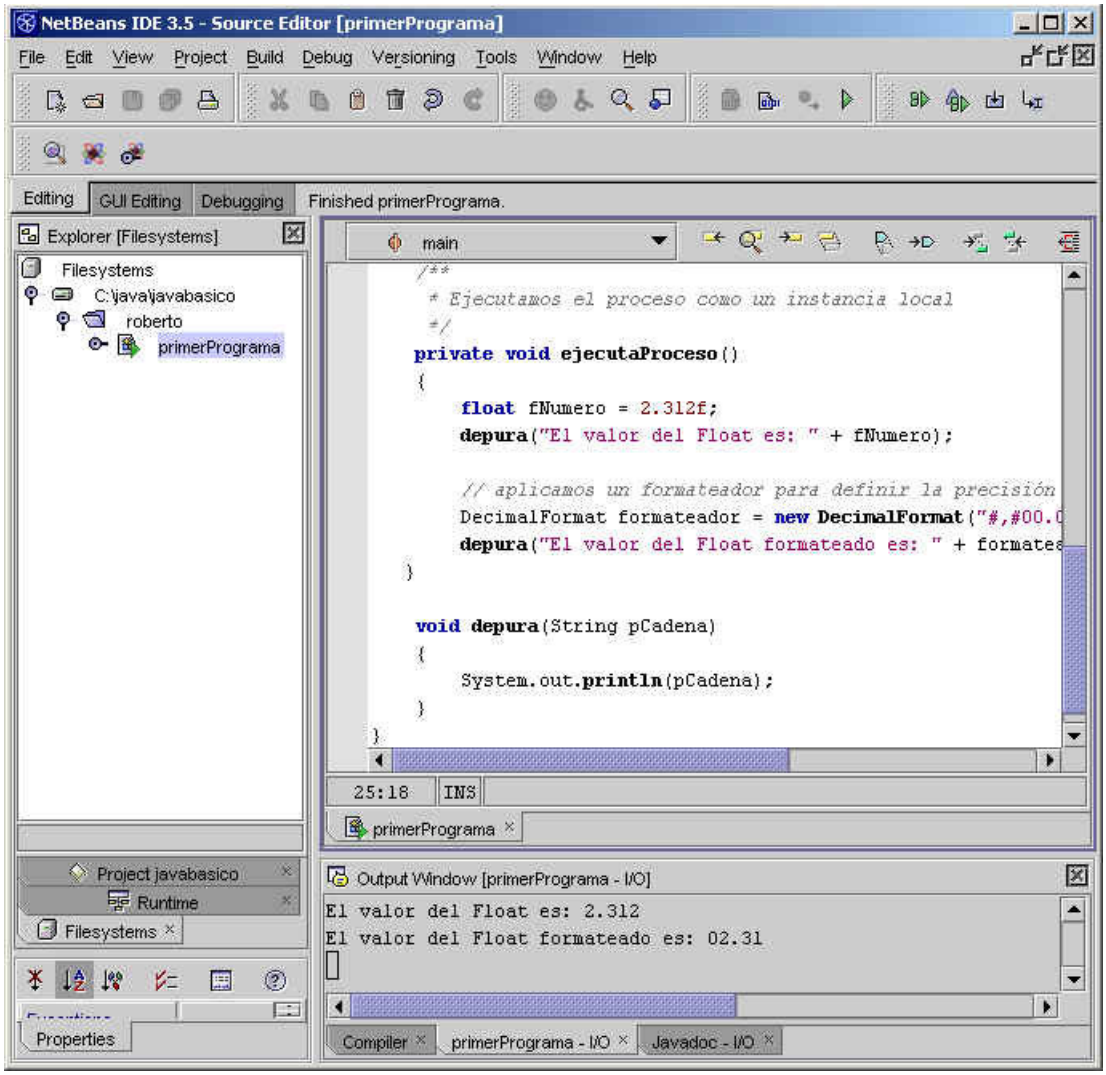
Cuando trabajamos con valores numéricos, muchas veces debemos presentar la salida en formatos muy concretos.

En el ejemplo siguiente podemos ver como se formatea por defecto un número y como podemos definir el formato de los números.

Con esta cadena, **"#,#00.0#;(-#,#00.0#)"** definimos el formato para los número positivos y negativos

- Los **#** indican valores no obligatorios
- Los **0** indican que si no hay valor se pondrá un cero

```
package roberto;
/*
 * primerPrograma.java
 *
 * Created on July 16, 2003, 3:53 AM
 */
import java.text.*;

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

    /** Constructor por defecto de la clase */
    public primerPrograma() {
    }

    /**
     * Punto de entrada estático de la aplicación
     * @param args recibe los parámetros de línea de comando
     */
    public static void main(String[] args) {
        primerPrograma oInstanciaLocal = new primerPrograma();
        oInstanciaLocal.ejecutaProceso();
    }

    /**
     * Ejecutamos el proceso como un instancia local
     */
    private void ejecutaProceso()
    {
        float fNumero = 2.312f;
        depura("El valor del Float es: " + fNumero);

        // aplicamos un formateador para definir la precisión y formato de los números
        DecimalFormat formateador = new DecimalFormat("#,#00.0#;(-#,#00.0#)");
        depura("El valor del Float formateado es: " + formateador.format(fNumero));
    }

    void depura(String pCadena)
    {
        System.out.println(pCadena);
    }

}
```



Mostrar enteros en formato Hexadecimal

En algoritmos complejos se suelen utilizar operaciones a nivel de bits, utilizando máscaras en formato hexadecimal.

Os mostramos un ejemplo de este tipo de operaciones:

- Definimos dos short
- Los desplazamos 16 bits a la izquierda (es como multiplicar por 2 a la 16)
- Mostramos la salida en Hexadecimal

```
package roberto;
/*
 * primerPrograma.java
 *
 * Created on July 16, 2003, 3:53 AM
 */
import java.text.*;

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

    /** Constructor por defecto de la clase */
    public primerPrograma() {
    }

    /**
     * Punto de entrada estático de la aplicación
     * @param args recibe los parámetros de línea de comando
     */
    public static void main(String[] args) {
        primerPrograma oInstanciaLocal = new primerPrograma();
        oInstanciaLocal.ejecutaProceso();
```

```
  }

  /**
   * Ejecutamos el proceso como un instancia local
   */
  private void ejecutaProceso()
  {
     short a = 0xFF;
     short b = 0xEE;

     int c = a;
     c <<= 16;
     c |= b;

     depura("El valor de a es: " + c);
     depura("En base hexadecimal " +  Integer.toHexString(c));
     depura("En base hexadecimal mayúsculas " +  Integer.toHexString(c).toUpperCase());
  }

  void depura(String pCadena)
  {
     System.out.println(pCadena);
  }
}
```

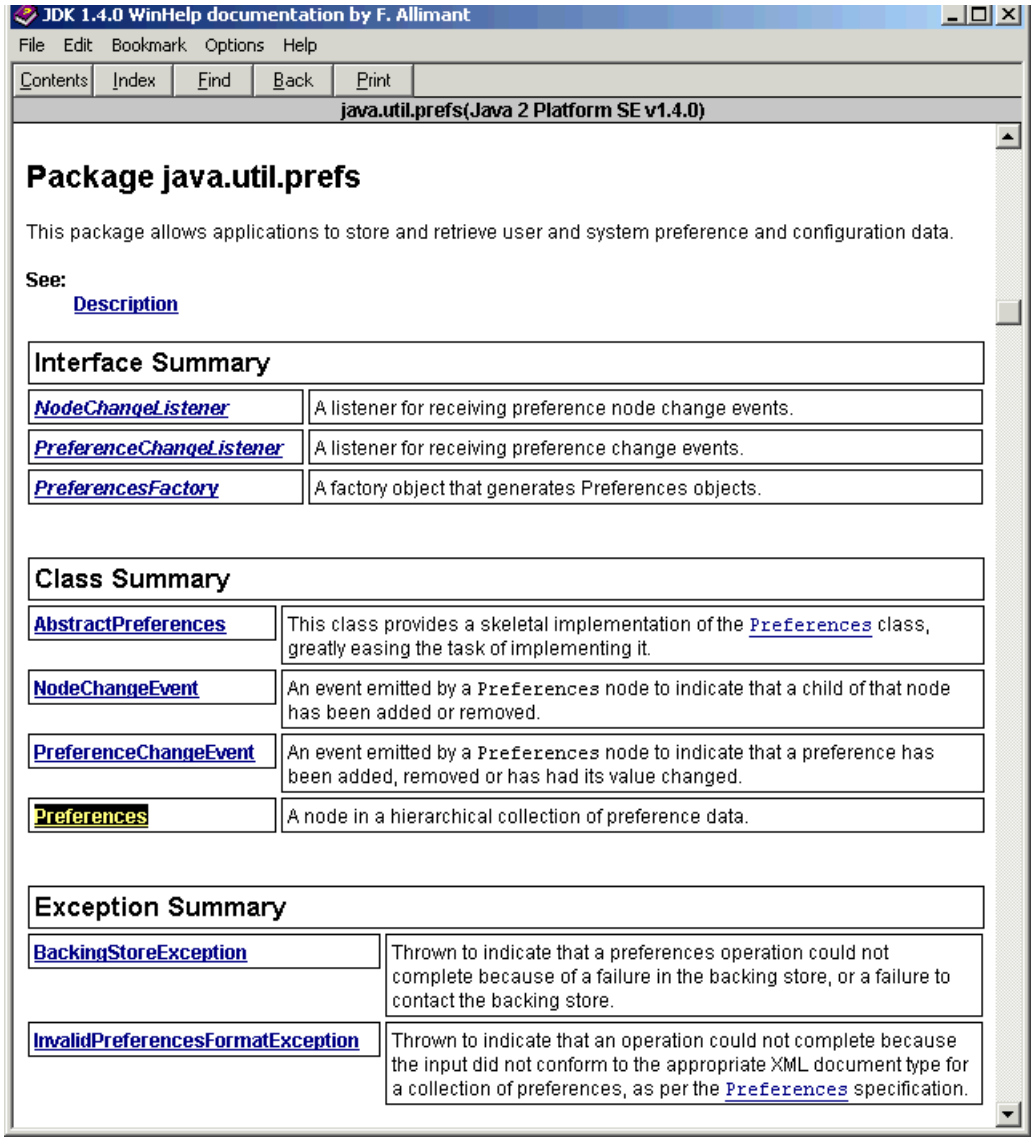
La salida es:

```
El valor de a es: 16711918
En base hexadecimal ff00ee
En base hexadecimal mayúsculas FF00EE
```

Guardar las preferencias de nuestro programa

Cuando construimos una aplicación, normalmente guardamos las preferencias del usuario ... para cuando vuelva a arrancar la aplicación.

Esta labor se ha normalizado en Java... a través de la clase Preferences



El procedimiento es bastante sencillo y os lo mostramos en el siguiente código

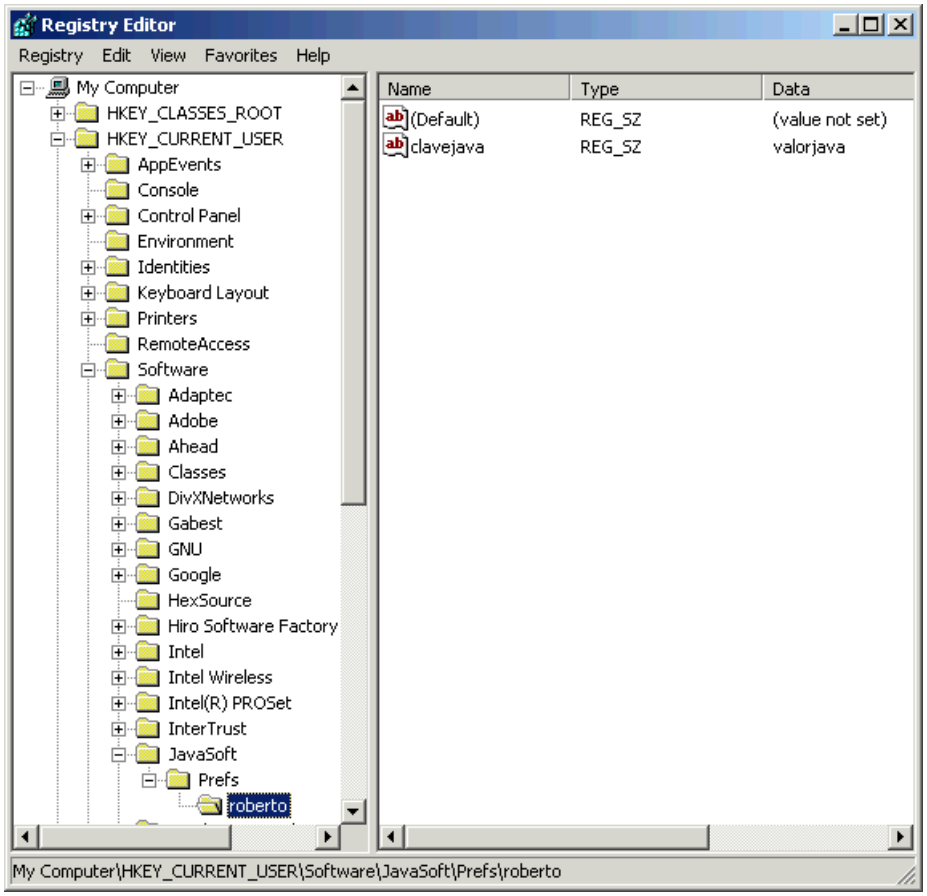
```
package roberto;
/*
 * primerPrograma.java
 *
 * Created on July 16, 2003, 3:53 AM
 */
import java.text.*;
import java.util.prefs.*;

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

    /** Constructor por defecto de la clase */
    public primerPrograma() {
    }

    /**
     * Punto de entrada estático de la aplicación
     * @param args recibe los parámetros de línea de comando
     */
    public static void main(String[] args) {
        primerPrograma oInstanciaLocal = new primerPrograma();
        oInstanciaLocal.ejecutaProceso();
    }

    /**
     * Ejecutamos el proceso como un instancia local
     */
    private void ejecutaProceso()
    {
        this.escribirPreferencias();
        this.leerPreferencias();
    }

    void escribirPreferencias()
```

```
{
   try
   {
      Preferences prefs = Preferences.userNodeForPackage(this.getClass());
      prefs.put("clavejava","valorjava");
      prefs.flush();
      depura("Proceso de escritura ejecutado correctamente");
   }
   catch(Exception e)
   {
      depura("Error al escribir preferencias");
   }

}

void leerPreferencias()
{
   try
   {
      Preferences prefs = Preferences.userNodeForPackage(this.getClass());
      String sValorRecuperado = prefs.get("clavejava","valorpordefecto");
      depura("El valor recuperado es: " + sValorRecuperado);
      depura("Proceso de lectura ejecutado correctamente");
   }
   catch(Exception e)
   {
      depura("Error al leer");
   }

}

void depura(String pCadena)
{
   System.out.println(pCadena);
}
}
```

La salida de nuestro programa es

```
Proceso de escritura ejecutado correctamente
El valor recuperado es: valorjava
Proceso de lectura ejecutado correctamente
```

En entorno Windows, esta clave se guarda en el Registro



Comparación entre objetos de nuestras clases

Cuando creamos una nueva clase y queremos comparar si son iguales dos objetos de la misma clase, podemos usar el método equals

Si vemos el siguiente trozo de código:

```
package roberto;
/*
 * primerPrograma.java
 *
 * Created on July 16, 2003, 3:53 AM
 */
import java.text.*;
import java.util.prefs.*;

class CEmpleado
{
   String dni;
   String nombre;

   CEmpleado(String dni, String nombre)
   {
      this.dni = dni;
      this.nombre = nombre;
   }
}

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

   /** Constructor por defecto de la clase */
   public primerPrograma() {
   }

   /**
    * Punto de entrada estático de la aplicación
    * @param args recibe los parámetros de línea de comando
    */
   public static void main(String[] args) {
      primerPrograma oInstanciaLocal = new primerPrograma();
      oInstanciaLocal.ejecutaProceso();
   }

   /**
    * Ejecutamos el proceso como un instancia local
    */
   private void ejecutaProceso()
   {
      CEmpleado sRoberto = new CEmpleado("1234", "");
      CEmpleado sTambienRoberto = new CEmpleado("1234", "");

      depura("El código interno de sRoberto es " + sRoberto.hashCode());
      depura("El código interno de sTambienRoberto es " + sTambienRoberto.hashCode());

      if (sRoberto.equals(sTambienRoberto) == true)
      {
         depura("Lo dos objetos son iguales");
      }
      else
      {
         depura("Los dos objetos son distintos");
      }
   }

   void depura(String pCadena)
   {
      System.out.println(pCadena);
   }
}
```

El resultado obtenido es el siguiente

```
El código interno de sRoberto es 3541984
El código interno de sTambienRoberto es 4565111
Los dos objetos son distintos
```

¿Podría ser?

Es decir, los objetos se consideran distintos porque la representación interna en Java de ellos es distinta y Java no tiene que realizar ningún trabajo adicional para hacer la comparación.

Para que el código anterior diese el mismo resultado (dos objetos iguales), podríamos redefinir el método equals ....

En este caso .... vamos a dar por iguales dos objetos con el mismo DNI

```
package roberto;
/*
 * primerPrograma.java
 *
```

```
 * Created on July 16, 2003, 3:53 AM
 */
import java.text.*;
import java.util.prefs.*;

class CEmpleado
{
   String dni;
   String nombre;

   CEmpleado(String dni, String nombre)
   {
      this.dni = dni;
      this.nombre = nombre;
   }

   public boolean equals(Object obj)
   {
      if( obj instanceof CEmpleado)
      {
         CEmpleado oObjetoLocal = (CEmpleado)obj;

         if(this.dni.compareTo(oObjetoLocal.dni) == 0)
         {
              return true;
         }
      }
      return false;
   }
}

/**
 *
 * @author  Administrator
 */
public class primerPrograma {

   /** Constructor por defecto de la clase */
   public primerPrograma() {
   }

   /**
    * Punto de entrada estático de la aplicación
    * @param args recibe los parámetros de línea de comando
    */
   public static void main(String[] args) {
      primerPrograma oInstanciaLocal = new primerPrograma();
      oInstanciaLocal.ejecutaProceso();
   }

   /**
    * Ejecutamos el proceso como un instancia local
    */
   private void ejecutaProceso()
   {
      CEmpleado sRoberto = new CEmpleado("44", "");
      CEmpleado sTambienRoberto = new CEmpleado("44", "");

      depura("El código interno de sRoberto es " + sRoberto.hashCode());
      depura("El código interno de sTambienRoberto es " + sTambienRoberto.hashCode());

      if (sRoberto.equals(sTambienRoberto) == true)
      {
         depura("Lo dos objetos son iguales");
      }
      else
      {
         depura("Los dos objetos son distintos");
      }
   }

   void depura(String pCadena)
   {
      System.out.println(pCadena);
   }
}
```

La salida en este caso será:

```
El código interno de sRoberto es 3541984
El código interno de sTambienRoberto es 4565111
Lo dos objetos son iguales
```

Por mantener principios de Java, cuando dos objetos son considerados como iguales, el metodo hashCode de ambos debe retornar el mismo valor.

Creamos un método (aunque un poco chapuza)

```
public int hashCode()
{
```

```
    return Integer.parseInt(dni);
}
```

Sobre el Autor ..

Puedes opinar sobre este tutorial aquí

Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....
Autentia = Soporte a Desarrollo & Formación

Autentia S.L. Somos expertos en:
J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Programa de dibujo en Java con NetBeans	En este tutorial os enseñamos a manejar el entorno de desarrollo NetBeans a través de la creación de una aplicación gráfica que sea capaz de pintar lineas de un modo persistente (a repintados). Es un buen ejemplo de gestión de eventos gráficos .
Generación automática de código JDBC	En este tutorial os enseñamos como, sin conocimiento de JDBC, crear vuestro programas en Java, gracias a JDBCTest.
Uso de JNDI, includes y cookies en Servlets	En este tutorial veremos como usar variables de entorno desde JNDI, incluir un servlet en otro (include) y como usar cookies en Servlets

www.AdictosAlTrabajo.com Opimizado 800X600