Ejercicio 1 (2.5 puntos)

La empresa BK, tras el desarrollo exitoso de la aplicación de venta de productos cosméticos nos ha encargado el desarrollo de una aplicación para móviles que permita a sus clientes consultar y realizar compras de sus productos. Los objetivos que persiguen con esta aplicación son los siguientes:

- Consultar los productos por categorías.
- Posibilidad de recibir notificaciones en el móvil con las principales novedades.
- Visualización de ofertas en función de las preferencias y compras previas de los clientes.
- Realización de pedidos y consulta del estado de un pedido ya realizado.

Tu tarea consiste en diseñar la planificación del proyecto software encargado, teniendo en cuenta todos los aspectos estudiados. Para ello lo primero que tendrás que determinar es el modelo de Ciclo de vida y a continuación planificar las distintas etapas de desarrollo del software.

Ciclo de Vida

El modelo de ciclo de vida idóneo es un modelo en cascada con retroalimentación ya que nos encontramos ante un modelo de software clásico, pero donde es necesario introducir realimentación entre etapas, de forma que podamos volver atrás en cualquier momento y corregir, modificar o depurar aspectos necesarios. Dado que no se prevén muchos cambios durante el desarrollo es el modelo más idóneo. Se trata de un proyecto con pocos cambios, poco evolutivo y los requisitos están claros. El inconveniente de este modelo es que puede ser muy rígido para este proyecto.

No obstante, también podría ser válido un modelo iterativo incremental, que está basado en el modelo en cascada con retroalimentación, ya que se trata de una tecnología novedosa donde se van ha producir muchos cambios y es posible que haya que introducir mejoras en diferentes versiones para adaptarnos a los avances tecnológicos. En este caso sería interesante desarrollar varias fases que se repitan y refinan donde se vayan propagando las mejoras de una fase a las siguientes. Con este modelo también evitamos la rigidez que puede introducir el modelo en cascada con retroalimientación.

Basándonos en lo anterior, también sería adecuado un modelo en espiral que nos permita una construcción repetida en forma de versiones mejoradas del proyecto donde se vaya incrementando la funcionalidad de cada una de las versiones desarrolladas. No obstante, este modelo presenta el inconveniente de que es un modelo muy complejo para este proyecto y de los tres presentados es el menos adecuado.

Aálisis de Requisistos del Sistema:

Requisitos funcionales.

- Consultar los productos por categoría.
- Posibilitar la recepción de notificaciones en el móvil con las principales novedades.
- Visualizar las ofertas en función de las preferencias y compras previas de los clientes.
- Realizar pedidos.
- Consultar el estado de un pedido realizado.

Requisitos no funcionales.

- Tratamiento simultáneo de peticiones.
- Tiempo de respuesta rápido del programa.
- Seguridad.

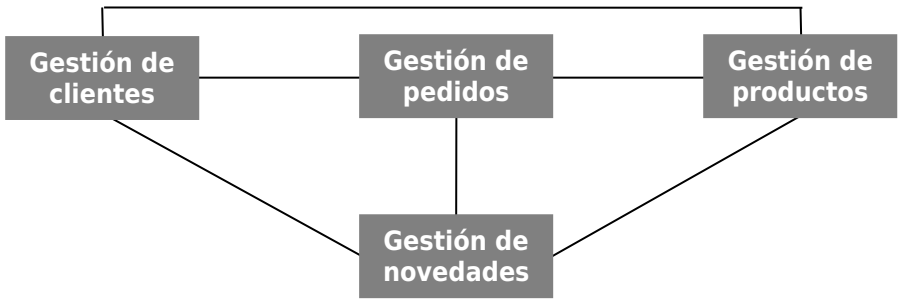
Diseño:

En esta etapa vamos a dividir el sistema en partes y a determinar la función de cada una de ellas. Debemos de determina que hará exactamente cada parte, para ello debemos crear un modelo funcional-estructural de los requerimiento del sistema global y poder así dividirlo y afrontar las distintas pares por separado.

De este modo, podríamos distinguir las siguientes partes:
Gestión de productos. Debe de realizar todas las acciones relacionadas con la gestión de los productos (alta, baja, modificación, consulta). La información de los productos se guardará en una tabla de la base de datos con los siguientes campos: improducto, nombreProducto, descripcionProducto, marcaProducto, modeoProducto, precioProducto.

- Gestión de clientes. Debe de realizar todas las acciones relacionadas con la gestión de los clientes (alta, baja, modificación, consulta). La información de los clientes se guardará en una tabla de la base de datos con los siguientes campos: idCliente, dniCliente, nombreCliente, apellidosCliente, direccionCliente, telefonoCliente, sexoCliente, fechaNacimientoCliente.
- Gestión de pedidos. Debe de realizar todas las acciones relacionadas con la gestión de los pedidos (alta, baja, modificación, consulta). La información de los pedidos se guardará en una tabla de la base de datos con los siguientes campos: idPedido, fechaPedido, idCliente, productosPedido, precioPedido.
- Gestión de novedades. Debe de realizar todas las acciones relacionadas con la gestión de las ofertas (alta, baja, modificación, consulta, configuración). La información de las ofertas se guardará en una tabla de la base de datos con los siguientes campos: idOferta, productosOferta, precioOferta, fechaInicioOferta, fechaFinOferta.

Estos módulos estarían relacionados de la siguiente forma en el sistema.



Tratándose de una aplicación para telefonía móvil, lo ideal es usar un lenguaje de programación multiplataforma que proporcione portabilidad a los distintos sistemas operativos para dispositivos móviles. Un leguaje tan difundido como Java sería una buena elección.

Lo mismo ocurre a la hora de elegir un sistema gestor de bases de datos. Lo ideal es buscar una solución tipo Oracle o MySQL.

Codificación:

En la etapa de codificación de debe de elegir un lenguajes de programación y codificar el programa. La elección del lenguaje de programación para codificar un programa dependerá de las características del problema a resolver, pero teniendo en cuenta que cualquier código fuente, debe de presentar características como modularidad, corrección, fácil de leer, eficiencia y portabilidad. Por ello una buena elección es usar como lenguaje de programación Java, que además de facilitar la consecución de las características ante-

riormente expuestas se trata de un software libre y permite programación orientada a objetos. Al tratare se un lenguaje ampliamente difundido, también dispone de una amplia gama de librerías que puede facilitar mucho nuestro trabajo.

Como entorno de desarrollo se puede usar NetBeans ya que es una herramienta consolidada en el mercado.

Pruebas:

En esta etapa se prueban los programas para detectar errores y se depuran. Generaremos un conjunto de datos de prueba para poder realizarlas. Usaremos un conjunto seleccionado y predefinido de valores límite a los que someteros la aplicación.
En la realización de pruebas es imprescindible para asegurar la validación y la verificación del software desarrollado.

Se efectuarán se siguientes tipos de pruebas sobre el software:

- Pruebas unitarias. Consisten en probar, una a una, las diferentes partes de software y comprobar su funcionamiento (por separado, de manera independiente). Una buena elección para este tipo de pruebas es la herramienta JUnit de Java.
- Pruebas de integración. Se lleva a cabo una vez que se han realizado con éxito las pruebas unitarias y consisten en comprobar el funcionamiento del sistema completo.
- Levaremos también a cabo una prueba Beta Test que se realizará sobre el entorno de producción donde el software va a ser utilizado por el cliente. En este caso, un teléfono móvil.

El período de prueba será pactado con el cliente.

Documentación:

Todas las etapas en el desarrollo deben quedar perfectamente documentadas. Es necesario dar toda la información a los usuarios del software que hemos desarrollado y poder acometer futuras revisiones del proyecto.

Se debe de ir documentando el proyecto en todas las fases del mismo, para pasar de una a otra de una forma clara y definida. De esta forma crearemos los siguientes documentos:

- Guía técnica. El objetivo de este documento es facilitar un correcto desarrollo, realizar correcciones en los programas y permitir un mantenimiento futuro. En él especificaremos el diseño de la aplicación, la codificación de los programas y las pruebas realizadas. Irá dirigido al personal técnico en informática (analistas y programadores).
- Guía de uso. Con este documento buscamos dar a los usuarios finales toda la información necesaria para utilizar la aplicación. Contendrá la descripción de la funcionalidad de la aplicación, la forma de comenzar a ejecutar la aplicación, ejemplos de uso del programa, los requerimientos software de la aplicación y la solución de los posibles problemas que se pueden presentar. Este documento irá dirigido a los usuarios que van a usar la aplicación (clientes).
- Guía de instalación. A través de este documento se pretende dar toda la información necesaria para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa. Contendrá toda la información necesaria para poner en marcha la aplicación, para su explotación y para mantener la seguridad del sistema. Este documento estará dirigido al personal informático responsable de la instalación, en colaboración con los usuarios que van a usar la aplicación (clientes).

Explotación:

La fase de explotación es la fase en la que los usuarios finales conocen la aplicación y comienzan a utilizarla. Durante esta fase, se instalará la aplicación, se realizará la puesta a punto y se comprobará el funcionamiento de la aplicación en el equipo final del cliente (teléfono móvil). Los programas serán transferidos al computador del usuario cliente y se configurarán y verificarán. Al tratarse de una aplicación para telefonía móvil, la instalación la llevará a cabo el cliente descargándola desde la página web correspondiente, pero antes de esto, habremos realizado una fase de explotación previa que garantice que no se produce ningún error mientras lo hace el cliente.

En este momento, llevaremos a cabo las Beta Test, que son las últimas pruebas que se realizan en los propios equipos del cliente y bajo cargas normales de trabajo.

Una vez instalada la aplicación, pasaremos a la fase de configuración. En ella, asignaremos los parámetros de funcionamiento normal y probaremos que la aplicación es operativa.

Mantenimiento:

La fase de mantenimiento es la fase en la que el software deberá actualizarse y evolucionar. Deberá ir adaptándose de forma paralela a las mejoras del hardware en el mercado y afrontar situaciones nuevas que no existían cuando el software se construyó.

Por otro lado, siempre surgen errores que habrá que ir corrigiendo y nuevas versiones mejoradas del producto.

Por ello, se pactará con el cliente un servicio de mantenimiento de la aplicación. Este servicio, tendrá un coste temporal y económico.

Durante este proceso se realizará el control, mejora y optimización del software.

También tendremos en cuenta aspectos como nuevas necesidades del cliente, adaptaciones a nuevas tendencias del mercado o nuevos componentes software y la corrección de posible errores futuros.

Ejercicio 2 (2.5 puntos)

Se está desarrollando un software que permita realizar operaciones con números complejos. El siguiente código muestra la codificación de la clase Complejo que ha realizado un programador y que deberás testear adecuadamente.

```
/**
* Clase básica que modela las operaciones de un número complejo
* @author Jose Luis Berenguel
*/
public class Complejo {
private double real;
private double imaginaria;

/**
* Constructor por defecto de la clase. Construye el objeto inicializa-
do
* a los valores (0,0).
*/
public Complejo(){
      this(0,0);
}
```

```
/**
* Constructor con parámetros para inicializar el objeto
* @param real Valor de la parte real del número complejo
* @param imaginaria Valor de la parte imaginaria del número complejo
*/
public Complejo(double real, double imaginaria){
      this.real = real;
      this.imaginaria = imaginaria;
}

/**
* Obtiene el valor de la parte real del número complejo
* @return El valor de la parte real del número complejo
*/
public double getImaginaria() {
      return imaginaria;
}

/**
* Modifica el valor de la parte imaginaria del número complejo
* @param imaginaria Nuevo valor de la parte imaginaria
*/
public void setImaginaria(double imaginaria) {
      this.imaginaria = imaginaria;
}

/**
* Obtiene el valor de la parte real del número complejo
* @return El valor de la parte real del número complejo
*/
public double getReal() {
      return real;
}

/**
* Modifica el valor de la parte real del número complejo
* @param real Nuevo valor de la parte real
*/
public void setReal(double real) {
      this.real = real;
}

/**
* Suma el número complejo <code>c</code> al objeto.
* @param c Objeto a sumar.
*/
public void suma(Complejo c){
      this.real = this.real + c.real;
      this.imaginaria = this.imaginaria + c.imaginaria;
}

/**
* Calcula la multiplicación del complejo <code>c</code> con el objeto.
* @param c Objeto a multiplicar
*/
public void multiplicacion(Complejo c){
    this.real = this.real*c.real - this.imaginaria*c.imaginaria;
    this.imaginaria = this.real*c.imaginaria + this.imaginaria*c.real;
}
```

```
/**
* Compara si dos números complejos son iguales. Dos números complejos
son
* iguales si la parte real y la parte imaginaria de ambos es igual.
* @param c El número complejo a comparar.
* @return <code>true</code> si los números son iguales y
<code>false</code>
* en caso contrario.
*/
public boolean equals(Complejo c){
boolean sonIguales=false;
    if(this.real == c.real && this.imaginaria == c.imaginaria)
        sonIguales = true;
        return sonIguales;
    }
}
```

Las operaciones de números complejos se realizan del siguiente modo. Sean los dos números complejos c1(3,5) y c2(4,8):

- c1 + c2 = (3+4, 5+8) = (7,13);
- c1 * c2 = (3*4 – 5*8, 3*8 + 5*4) = (-28, 44)

Tu tarea será realizar la implementación de los test unitarios para los siguientes métodos:

- Método suma.
- Método multiplicación.
- Método de comparación de igualdad.

Test unitario para el Método Suma

```
/**
 * . Prueba Exitosa.
 */
public void testSumaOk() {
    System.out.println("Test para el método Suma");

    double parteReal1 = 3;
    double parteReal2 = 4;
    double  parteImagniaria1 = 5;
    double  parteImagniaria2 = 8;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    try{
        numComplejo1.suma(numComplejo2);

        assertTrue( (numComplejo1.getReal()==7) &&
(numComplejo1.getImaginaria()==13));
    }catch (Exception e){
        fail("El método suma ha fallado.");
    }
}
```

Test unitario para el Método Multiplicación

```
/**
 * Prueba exitosa.
 */
public void testMultiplicacionOk() {
    System.out.println("Test para el método Multiplicacion");

    double parteReal1 = 3;
    double parteReal2 = 4;
    double  parteImagniaria1 = 5;
    double  parteImagniaria2 = 8;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    try{
        numComplejo1.multiplicacion(numComplejo2);

        assertTrue((numComplejo1.getReal()==-
28)&&(numComplejo1.getImaginaria()==44));
    }catch (Exception e){ {

        fail("El método multiplicacion ha fallado.");
    }
}
```

Test unitario para el Método de Comparación de Igualdad

```
/**
 * Prueba exitosa.
 */
public void testEqualsOk() {
    System.out.println("Test para el método Equals");

    double parteReal1 = 3;
    double parteReal2 = 3;
    double  parteImagniaria1 = 7;
    double  parteImagniaria2 = 7;

    Complejo numComlejo1 = new Complejo(parteReal1, parteImaginaria1);
    Complejo numComlejo2 = new Complejo(parteReal2, parteImaginaria2);

    boolean expResult = true;

    try{
        boolean result = numComplejo1.equals(numComplejo2);
        assertEquals(expResult, result);

    }catch (Exception e){
        fail("El método equals ha fallado.");
    }
}
```

Módulo: ENTORNOS DE DESARROLLO
I.E.S.: AGUADULCE C.F.G.S. : DESARROLLO DE APLICACIONES WEB
Fecha: 2 DE FEBRERO DE 2012 Hora de Comienzo: 9:00 H
Examen: Parte 1 (TEÓRICA) Duración: 2 HORAS
Nombre: D.N.I.:
Centro en el que se realiza el examen:

INSTRUCCIONES:

Marca el profesor que tengas asignado: **Fran [ ]** **José Luis [ ]**

La puntuación total del examen PARTE 1 + PARTE 2 será de 10 puntos. Parte teórica: 5 puntos**. Parte Práctica:** 5 puntos.

La nota del examen se calculará como la media ponderada de la parte teórica (50% del total) y la parte práctica (50% del total), siempre y cuando la nota de cada parte sea mayor o igual que 3.

Para el examen práctico se podrá hacer uso del material bibliográfico y digital que se estime oportuno, así como de apuntes. No obstante, se advierte del peligro de pérdida de tiempo que conlleva ponerse a consultarlo durante el examen, pudiendo consumirse el tiempo disponible en la consulta, y quedándose sin tiempo para las respuestas.

PARTE 1: CUESTIONES TEÓRICAS: (5 puntos) (Cada pregunta correcta puntuará con 0,1 puntos, cada incorrecta restará 0,05 puntos. Si de deja sin contestar ni suma ni resta)

1. ¿Qué lenguaje es directamente ejecutable por la computadora?
a. Fuente.
b. **Máquina.**
c. Ensamblador.
d. Objeto.

2. Tener deficiencias en la fase de _____ es la principal causa del gran porcentaje de fracasos de los proyectos software.
a. Diseño
b. Codificación
c. Documentación
d. **Análisis**

3. ¿Qué componente es vital para unir archivos en la generación del ejecutable?
a. Compilador.
b. **Linker.**
c. Ensamblador.
d. Intérprete.

4 La etapa consistente en dividir el problema general en partes con funciones definidas es:
a. El análisis.
b. **El diseño.**
c. La programación.
d. La ejecución.

5. El entorno de ejecución de aplicaciones está compuesto por _____ y la máquina virtual del lenguaje de programación.
a. El enlazador
b. El sistema operativo
c. El lenguaje de programación
d. **El API**

6. El documento de la guía técnica contiene información:
a. Para que los usuarios de la aplicación sepan utilizarla.
b. Para que los desarrolladores y usuarios finales instalen la aplicación.
c. Para que los analistas y desarrolladores pueden mantener la aplicación.
d. Para analistas y usuarios finales.

7. ¿Cómo se llama el cambio consistente en mejorar la funcionalidad de todo software?
a. Correctivo.
b. **Perfectivo.**
c. Adaptativo.
d. Evolutivo.

8. En la etapa de compilación el tipo de código que se obtiene es:
a. **Código Objeto.**
b. Código Fuente.
c. Código ejecutable.
d. Código ejecutable directo.

9. De todos los tipos de software,_____ es el encargado de gestionar los recursos hardware de un computador.
a. La aplicación.
b. El compilador.
c. El código objeto.
d. **El sistema operativo**.

10. En un supuesto práctico, ¿qué tipo de requisito es el deseo del cliente de incluir en la aplicación el control de stock de productos en el almacén?
a. No funcional.
b. **Funcional.**
c. No es un requisito.
d. Depende de la fase de codificación.

11. ¿Cómo se llama el proceso de traducción de código fuente a código objeto?
a. Interpretación.
b. **Compilación.**
c. Ejecución.
d. Runtime Environment.

12. ¿En qué momento se suelen realizar las pruebas Beta Test?
a. Antes de las pruebas de integración.
b. **En el mismo entorno productivo de la aplicación, una vez instalada**.
c. Después de realizar las pruebas unitarias.
d. lo largo de la etapa de mantenimiento.

13. La realización de pruebas _____ nos permite detectar errores de cada parte del programa por separado.
a. semánticas
b. de integración
c. funcionales
d. **unitarias**

14. El modelo en cascada con realimentación es
a. El modelo perfecto si los requisitos están claros y el proyecto es rígido
b. El modelo idóneo si se prevén muchos cambios durante el desarrollo.
c. Es el modelo de vida clásico del software.
d. Es un tipo de modelo evolutivo.

15. El modelo en espiral
a. Se trata de varios ciclos en cascada que se repiten y se refinan en cada incremento.
b. Se divide en 6 zonas, llamadas regiones de tareas.
c. Es un modelo bastante sencillo.
d. Es un modelo rígido que permite pocos cambios.

16. Una máquina virtual
a. Es un tipo especial de Hardware que se instala el el ordenador para poder ejecutar los programas.
b. Es un tipo especial de software cuya misión es separar el funcionamiento del ordenador de los componentes hardware instalados.
c. Funciona como una capa de Hardware de bajo nivel y actúa como puente entre el bytecode de la aplicación y los dispositivos físicos del sistema.
d. Es un tipo de software que se utiliza para compilar los programas de código fuente a código objeto.

17. Entre las ventajas de utilizar Frameworks tenemos:
a. Poca dependencia del código.
b. Consume pocos recursos.
c. **Permite la reutilización de partes de código para otras aplicaciones.**
d. Poco portable.

18. ¿Qué componentes de los IDE permite realizar la escritura del código?
a. Depurador.
b. **Editor de textos.**
c. Compilador.
d. Intérprete.

19. Tener previamente instalado _____ es imprescindible para poder instalar y ejecutar NetBeans.
a. Linux
b. JVM
c. JRE
d. **JDK**

20. ¿Qué componente es responsable del seguimiento de las variables en tiempo de ejecución?
a. Compilador.
b. **Depurador.**
c. Editor de textos.
d. Intérprete.

21. De las funcionalidades siguientes, cual no se pueden conseguir añadiendo plugins a nuestro entorno.
a. **Reducción de tiempo de ejecución.**
b. Posibilidad de importar proyectos de otros lenguajes de programación.
c. Refactorización de programas.
d. Utilidades para la realización de pruebas al software.

22. La extensión propia de los módulos y plugins en NetBeans es:
a. .nmb.
b. **.nbm.**
c. ~.jar.
d. ~tar.gaz.

23. ¿Cómo se llama el primer software que se consideró el precursor de los actuales IDE, en la década de los 70?
a. MSDOS.
b. **Maestro.**
c. Cliente.
d. Evolutivo.

24. El lenguaje _____ se considera que fue el primero en usar un IDE.
a. C++
b. JavaScript
c. PHP
d. **BASIC**

25. En Linux no podemos instalar el entorno:
a. **Visual Studio.**
b. NetBeans.
c. Eclipse.
d. Gambas.

26. ¿Cómo se llama el cambio consistente en mejorar la legibilidad de programas, sin alterar la funcionalidad del mismo?
a. Correctivo.
b. **Refactorización.**
c. Adaptativo.
d. Evolutivo.

27. La extensión _____ es característica de proyectos empaquetados en Java.
a. .java
b. .class
c. .gaz
d. **.jar**

28. De los siguientes lenguajes, cual no es orientado a objetos?
a. **Lenguaje C.**
b. Lenguaje C++.
c. Lenguaje Java.
d. Lenguaje PowerBuilder.

30. ¿Cómo se llama la adición de plugins sin salir del IDE?
a. Off-line.
b. **On-line.**
c. Instantánea.
d. Demorada.

31. ¿Cómo se llama el proceso de alteración de código para mejorar su legibilidad, sin cambiar su funcionalidad asociada?
a. Depuración.
b. **Refactorización.**
c. Ejecución.
d. Compilación.

32. ¿Qué es el llamado manifest file ?
a. Un archivo de manifiesto.
b. **Archivo especial que identifica a un módulo**.
c. Archivo resultado de la compilación de un programa.
d. Un archivo ejecutable.

33. De las siguientes funciones, cual de ellas no se puede realizar con el editor de textos en el IDE:
a. **Conseguir la compilación de las aplicaciones.**
b. Resaltar y colorear la sintaxis del código.
c. Inserción automática de paréntesis y corchetes.
d. Proponer sugerencias de resolución de problemas.

34. ¿En qué momento se realiza la ejecución de un programa?
a. Antes de depurar el programa.
b. **Una vez corregido, compilado y depurado el programa.**
c. Antes de compilar el programa.
d. Es indiferente el momento de la ejecución.

35. El archivo llamado _____ es el archivo principal de una aplicación Java.
a. java.main
b. java.first
c. include
d. **Main.java**

36. ¿Cual de los siguientes entornos de desarrollo es propietario?
a. Netbeans.
b. Eclipse.
c. JCreator
d. Gambas.

37. La prueba de software.
a. Solo sirve para verificar el sistema.
b. **Sirve para verificar y validar el sistema.**
c. Solo sirve para validar el sistema.
d. Su realización es opcional.

38. En la planificación de pruebas.
a. Se depura el programa.
b. Se realiza la documentación de las pruebas.
c. Se establecen puntos de ruptura en el código.
d. **Se diseñan los tipos de prueba y los casos de prueba**.

39. ¿Qué componente del IDE es básico en la realización de pruebas?
a. Compilador.
b. **Depurador.**
c. Ensamblador.
d. Intérprete.

40. La herramienta de prueba unitaria más extendida en Java es
a. Es SimpleTest.
b. **El JUnit.**
c. El NUnit.
d. MOQ.

41. La regresión es
a. un tipo de prueba de validación.
b. **un proceso que se realiza cuando se produce un cambio en el código.**
c. un paso necesario en la depuración del programa.
d. un proceso asociado a la fase de diseño del proyecto.

42. Un caso de prueba.
a. Es cada fase de la planificación de pruebas.
b. Es cada elemento que interviene en la depuración.
c. Es diseñado con la colaboración del cliente.
d. **Se diseña intentando que la probabilidad de detección de errores sea máxima.**

43. ¿Cómo se llama la prueba que comprueba el cumplimiento de los requisitos funcionales?
a. Regresión.
b. **Validación.**
c. Integración.
d. Sistema.

44. El objetivo del cubrimiento
a. es obtener casos de prueba representativos.
b. **comprobar que todos los caminos se pueden ejecutar.**
c. establecer casos de prueba con valores en el límite del rango.
d. establecer clases de equivalencia que disminuyan el número prueba.

45. Las clases de equivalencia.
a. Nos ayudan a diseñar casos de prueba con valores límite.
b. Nos permiten validar el sistema.
c. Son herramientas de depuración.
d. **Nos permite crear casos de prueba representativos de un conjunto de valores posibles.**

46. Con las clases de equivalencia diseñamos casos de pruebas
a. con valores fuera del rango admitido.
b. **con valores representativos del rango admitido.**
c. con valores en el límite del rango admitido.
d. todas las respuestas anteriores no son válidas.

47. ¿Qué herramienta de automatización de pruebas no es para Java?
a. Unit.
b. **FoxUnit.**
c. TestNG.
d. JTiger.

48. ¿En qué momento se suelen realizar las pruebas de la unidad?
a. **Antes de las pruebas de integración.**
b. En el mismo entorno productivo de la aplicación, una vez instalada.
c. Después de realizar las pruebas unitarias.
d. A lo largo de la etapa de mantenimiento.

49. La realización de pruebas _____ nos permite detectar errores de cada parte del programa por separado.
a. semánticas.
b. de integración.
c. validación.
d. **unitarias.**

50. Las pruebas funcionales.
a. **son pruebas de caja negra.**
b. son pruebas de caja blanca
c. analizan y prueban directamente el código de la aplicación.
d. Todas las respuestas anteriores son falsas.
.

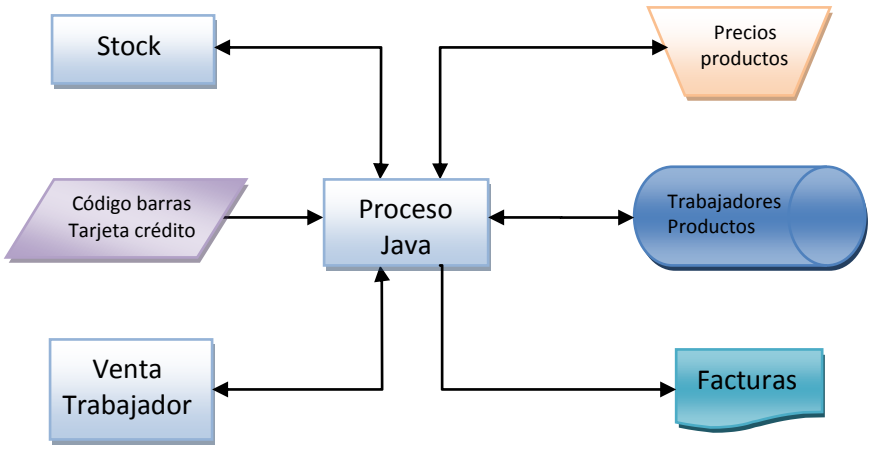
La empresa BK desea una aplicación que no requiere de demasiados cambios y que posea una estructura lo suficientemente rígida, y con unos requisitos lo suficientemente claros como para optar por un modelo de ciclo de vida de los denominados ***en cascada con retroalimentación***.

Como una de las premisas es la utilización de software libre, decidimos desarrollar el proyecto con lenguaje de programación Java, el cual, al ser un lenguaje orientado a objetos, nos dará la suficiente libertad para poder crear una aplicación lo bastante amplia para abarcar cualquier requerimiento solicitado por el cliente.

Así mismo, para poder realizar dicho proyecto, nos apoyaremos en una herramienta muy eficaz y que nos permite trabajar con lenguaje Java, y que, aunque es software propietario, igualmente es gratuito y libre, como JDeveloper de Oracle.

Tanto el lenguaje Java como el paquete JDeveloper lo podemos descargar desde la web principal de Oracle, y tras su instalación, ponernos a trabajar desarrollando la aplicación solicitada.

Para comenzar con el proyecto nos creamos un pequeño esquema indicativo de los distintos requerimientos solicitados:



Como **requisitos funcionales** tenemos los siguientes:

- Almacenar los datos del trabajador:
  - DNI
  - Nombre
  - Apellidos
  - Nº Seguridad Social
  - Fecha de nacimiento
  - Teléfono
  - Localidad
- Almacenar información de los productos:
  - Código
  - Marca
  - Nombre comercial
  - Precio
  - Cantidad
- Proporcionar facturas de las ventas.
- Llevar la cuenta de lo que vende cada trabajador.

- Controlar el stock de productos en almacén.
- Operar con lector de código de barras y tarjetas de crédito.
- Controlar los precios de los productos y ofrecer la posibilidad de operar con ellos.

Y como **requisitos no funcionales**:

- Reducir el tiempo de respuesta de la aplicación
- Imposibilitar el proceso simultáneo de peticiones aunque haya varios equipos funcionando al mismo tiempo

Tras estas premisas iniciales, planificaremos una serie de reuniones con el cliente para poder perfilar con más detenimiento cada uno de los requerimientos de la aplicación a diseñar, como por ejemplo:

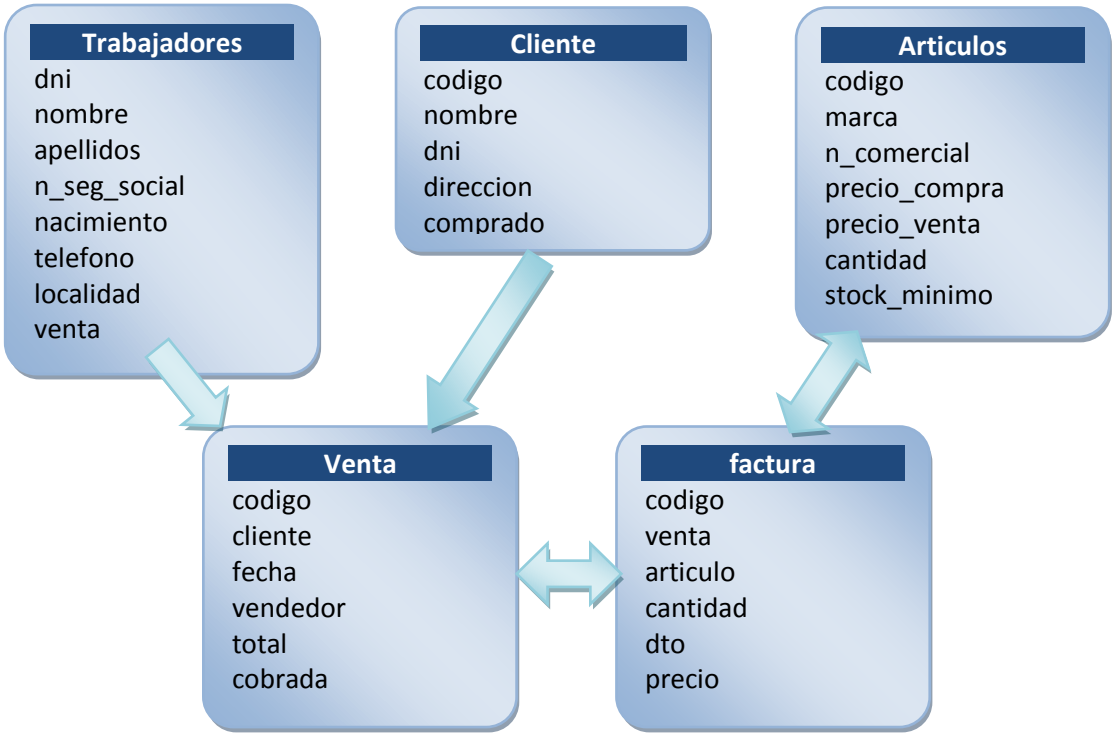
- Si necesitamos controlar el stock de almacén, deberemos añadir un campo más para incluir el stock mínimo y conseguir que el programa nos avise cuando se vaya a rebasar (***cantidad*** menor o igual que ***stock mínimo***).
- Al necesitar emitir facturas, necesitamos igualmente incluir los datos de los clientes a los que irán dirigidas, y el código del vendedor que realizará la venta para poder controlar cuántas transacciones ha realizado dicho empleado.
- Para poder admitir tarjetas de crédito como pago, tendremos que poner un campo nuevo en la factura que permita indicar la forma de entrega económica y opte por simplemente indicar qué cantidad se abona en metálico o que se va a hacer uso del mencionado dispositivo magnético para realizar el pago.
- Los artículos que se vayan a vender se obtendrán de la tabla de artículos pero se controlarán en una nueva tabla que recogerá en cada registro la factura a la que pertenece, la cantidad comprada, el código del producto, el precio e incluso un campo que denominemos descuento por si en cualquier momento tuviéramos que realizar una rebaja en un producto determinado dentro de una operación estipulada.

Cuando finalicemos con el refinamiento de todas estas premisas gracias a las reuniones con los encargados de la empresa y empleados que vayan a manejarla, sólo nos quedará comprobar cuáles son los equipos que se van a utilizar y si será necesario ampliar o modificar el parque informático de dicha empresa.

Con todo ello terminamos con la fase del **Análisis de Requisitos** y tendremos que enfocar el problema en base a un diseño eficaz, para lo cual podemos basarnos en el esquema realizado anteriormente con el objetivo de dividir dicho sistema en partes, determinando la función que realizará cada una de ellas de forma inequívoca.

Crearemos una base de datos con SQL para poderla utilizar con Java mediante JDBC.

Tenemos dos entidades ya definidas con claridad como son ***Trabajadores*** y ***Artículos*** aunque, como vimos anteriormente, tendríamos que crear otras como son ***Ventas*** con la que podríamos obtener los artículos vendidos, las facturas a emitir o ya emitidas, los trabajadores que han realizado la venta, etc. ***factura*** en la que se grabará toda la información individual de cada artículo vendido, y ***Cliente*** para poder identificar a quién se le realiza la venta e ir controlando las adquisiciones realizadas.



Con ello terminamos la fase del **Diseño** y comenzamos con la **Codificación**, y dado que el cliente ha decidido utilizar software libre, procederemos a la creación de la aplicación utilizando el lenguaje Java y apoyándonos en el entorno de desarrollo JDeveloper, el cual usaremos para el diseño de la base de datos, sus tablas, relaciones, y toda la aplicación en general, obteniendo los ficheros objeto en esta fase de codificación, el cual podremos utilizar para realizar las ***pruebas unitarias*** gracias a JUnit que es el entorno de pruebas para Java.
Una vez realizada con éxito este conjunto de pruebas unitarias, comprobaremos el funcionamiento de todo el sistema con todas sus partes interrelacionadas, para lo cual contactaremos con el cliente con el fin de poder implementar en sus equipos una versión *Beta* que nos permita probar su desarrollo en el entorno donde se implantará definitivamente.

Cuando el conjunto de todas estas pruebas haya finalizado con éxito, podremos comenzar a crear la ***documentación*** donde especificaremos todas y cada una de las etapas realizadas en el desarrollo del software, es decir, tendremos que entregar tres guías al cliente:

- ***Guía técnica:*** Dirigida a analistas y programadores, y que nos permitirá realizar correcciones y mantener la aplicación en un futuro. Incluiremos el diseño de la aplicación, la codificación de los programas y las pruebas que se le han realizado.
- ***Guía de uso:*** Dirigida a los usuarios finales o clientes. Describiremos la funcionalidad de la aplicación, cómo comenzar a ejecutarla, ejemplos de uso, requerimientos software y solución a los posibles problemas que se puedan presentar, es decir, intentaremos dar a los usuarios finales toda la información necesaria para utilizar la aplicación.
- ***Guía de instalación:*** Incluiremos toda la información necesaria para la puesta en marcha y la explotación, así como la seguridad del sistema. Esta información la dirigiremos al personal informático responsable de la instalación, en colaboración con los usuarios finales (clientes). Con ella intentaremos dar toda la información para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa.

Comprobado que el software es fiable, carece de errores y terminado el proceso de documentación, procederemos a dar a conocer la aplicación a los usuarios finales para que comiencen a utilizarla, o lo que se denomina ***fase de explotación***, para lo cual haremos que los futuros clientes estén presentes en el proceso de instalación y le iremos comentando el proceso, realizando las Beta Test en los equipos

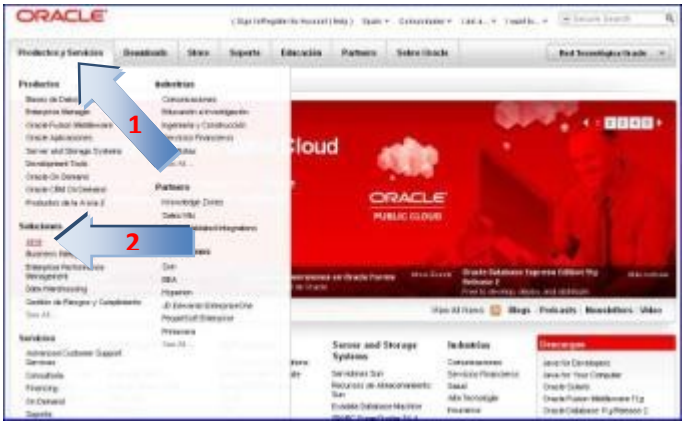
del cliente y con carga de trabajo normales (como el software es “a medida”, nosotros realizaremos la instalación).

Para finalizar con el ciclo de vida del software que hemos creado procederemos a pactar con el cliente un servicio de ***mantenimiento*** de la aplicación que pueda conllevar una futura mejora de la funcionalidad del software, nuevas necesidades, expansiones o incluso modificaciones y actualizaciones para adaptarse a las nuevas tendencias del mercado, o al nuevo hardware que pueda adquirir el cliente.

Se le ofrecerá al cliente un listado con los distintos precios de dicho mantenimiento, ya que no cuesta lo mismo añadir una tabla más para controlar los proveedores, que modificar la aplicación para que admita entrada de datos a través de una red externa que incluiría un extra de seguridad.

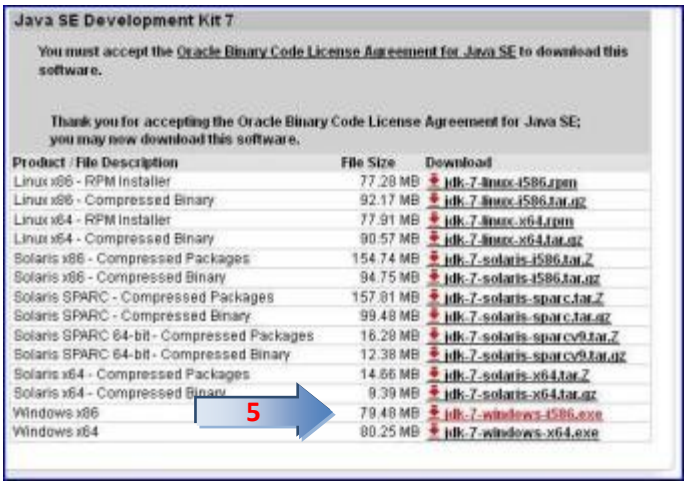
Le ofreceremos un precio por un mantenimiento ***perfectivo*** (que mejorará su funcionalidad), ***evolutivo*** (modificaciones o expansiones para futuras nuevas necesidades), ***adaptativo*** (modificaciones o actualizaciones que se adapten a las nuevas tendencias del mercado o nuevos componentes hardware) y ***correctivos*** (para errores que pueda tener la aplicación en el futuro).

Para la descarga de Java he optado por dirigirme a su propietario ORACLE para iniciar la grabación desde su página oficial: http://www.oracle.com/es/index.html
donde seleccionamos *Java*[2] en el apartado *Soluciones* del menú *Productos y Servicios*[1].



Pulsamos sobre *Java Platform, Standard Edition*[3], en el apartado de *Java para desarrolladores*

Hacemos clic sobre el botón marcado como *Java EE*[4], ya que sólo necesitamos el compilador y el entorno de ejecución Java.



Ahora, y en mi caso que tengo instalado Windows Vista 32 bits, descargaré el jdk (*Java Development Kit*) marcado para Windows x86[5].

Comenzará la descarga, y dependiendo de la velocidad de bajada que poseamos, tardará bastante (son casi 80 MB)

Una vez que haya concluido la descarga procederemos a su instalación, para lo cual tendremos que darle permisos de administrador

Nos aparece la primera pantalla de instalación avisándonos que comenzará el asistente de instalación. Pulsamos sobre *Next*

Dejamos las opciones que vienen por defecto (aplicaciones a instalar, directorios donde grabarlo, etc) y pulsamos sobre *Next*.

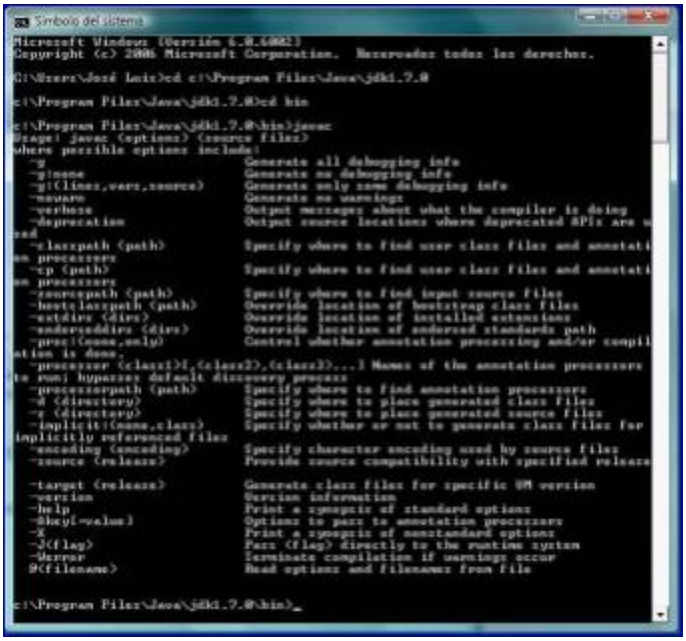
Comienza la instalación

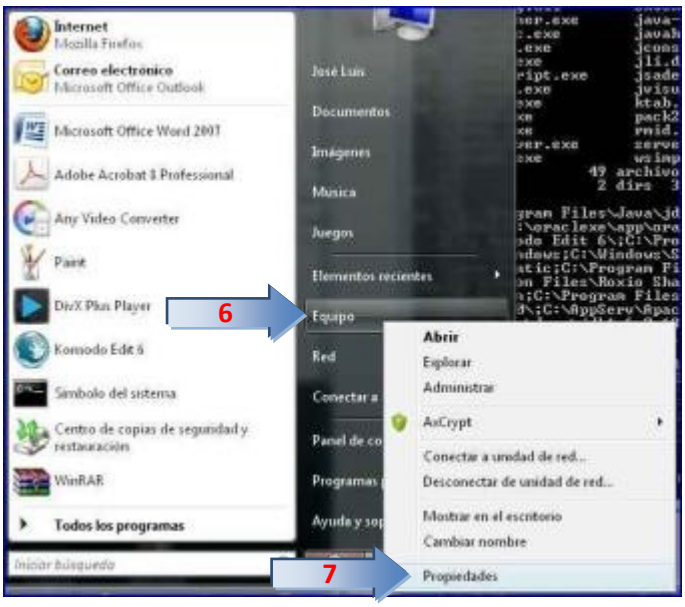
Dejamos el directorio que viene por defecto

Instalará Java en el directorio proporcionado.

Si queremos registrarnos nos enviará por email información sobre futuras actualizaciones y ofertas sobre otros productos de Oracle.
Pulsamos sobre *Finish* y finalizamos la instalación.

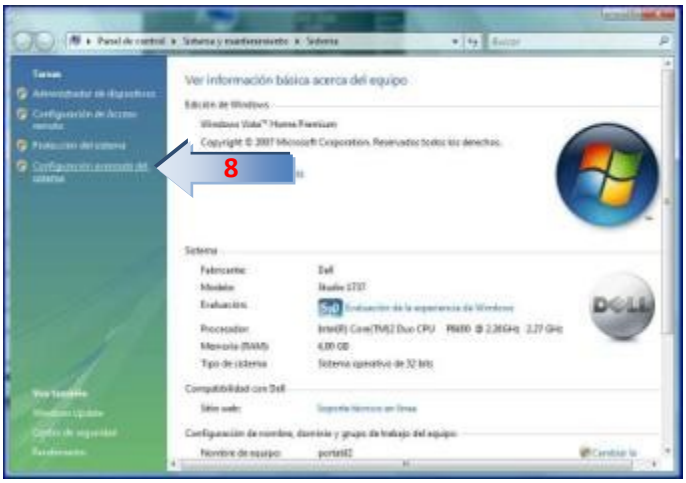
Si una vez instalado Java nos desplazamos a través del terminal de comandos hasta el directorio donde hemos realizado la instalación:
cd c:\Program Files\Java\jdk1.7.0
podremos probar su instalación tecleando *javac* en el directorio *bin*:
cd bin
javac

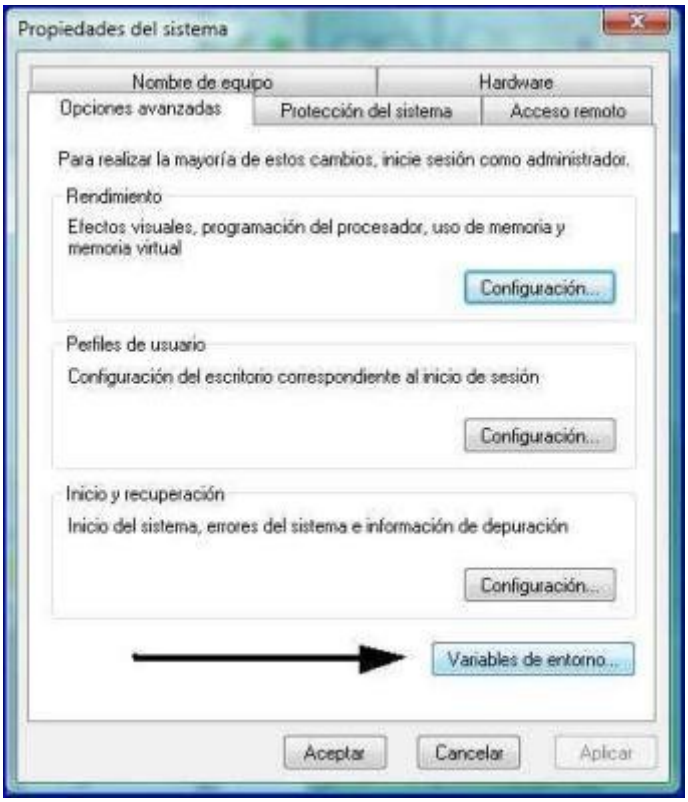




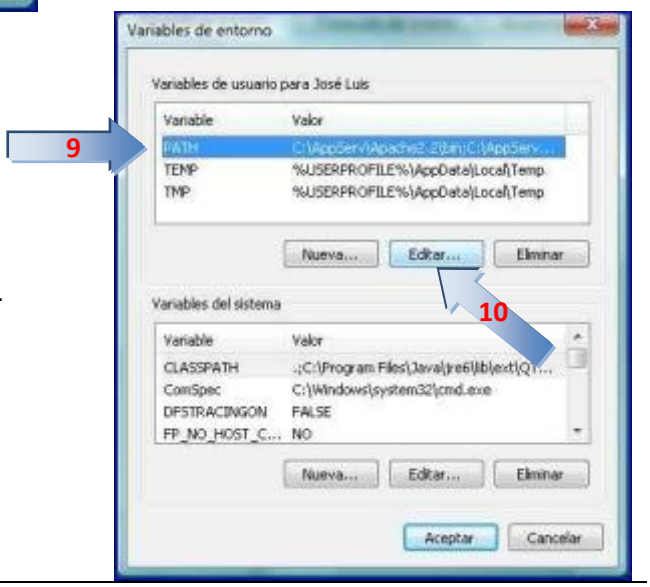
Ahora definiremos las variables correspondientes para poder utilizar Java desde cualquier lugar de nuestro equipo, para lo cual pulsamos el botón derecho del ratón sobre *Equipo*[6] y pulsamos en Propiedades[7].

Pulsaremos sobre *Configuración avanzada del sistema*[8].





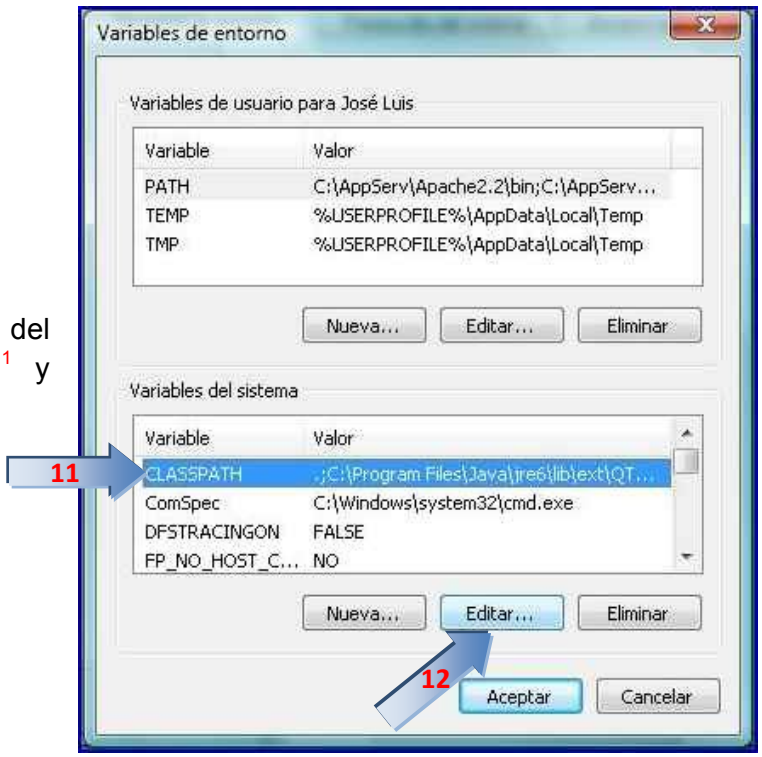
Pulsaremos sobre el botón de *Variables de entorno*.

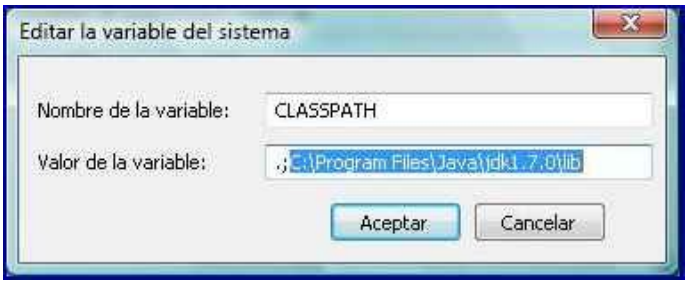


Seleccionamos *PATH*[9] y pulsamos sobre *Editar*[10].

Añadiremos el camino completo hasta el directorio *bin* en el valor de la variable, asegurándonos que lo insertamos a continuación del actual contenido, y separado de él por un punto y coma (**;**)

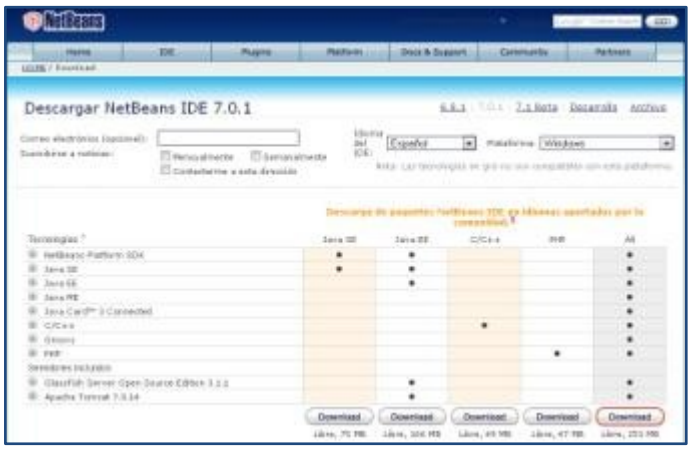
Ahora, en el apartado de variables del sistema, seleccionamos *CLASSPATH*[11] y pulsamos sobre *Editar*[12].





Configuramos la variable *CLASSPATH* con la ruta del directorio *lib* del apartado donde hemos instalado Java. La dirección hemos de ponerla comenzando por un punto y punto y coma (**.;**) con el fin de que pueda buscar nuevas librerías incluso en el directorio donde ejecutemos nuestra aplicación.

Para el NetBeans tendremos que ir a página web oficial en http://netbeans.org/index.html y descargar su última versión, la 7.0.1

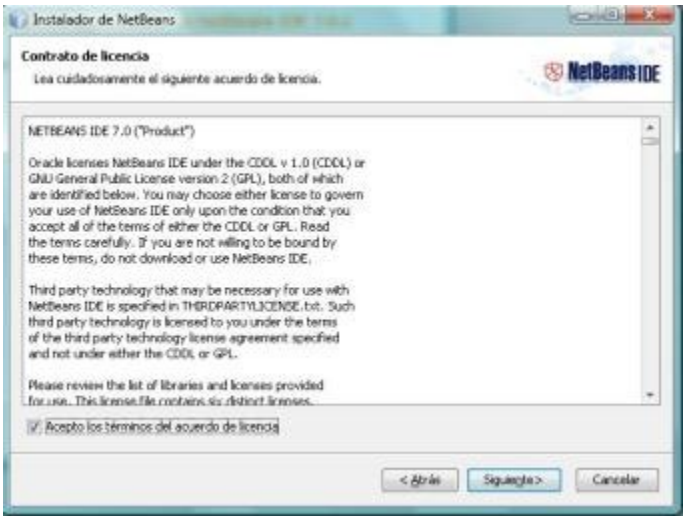


En mi caso selecciono la versión en español para Windows que admite todas las tecnologías e incluye dos servidores (*para un futuro*)

Guardamos el fichero de instalación en una carpeta donde iremos para poder instalar la aplicación

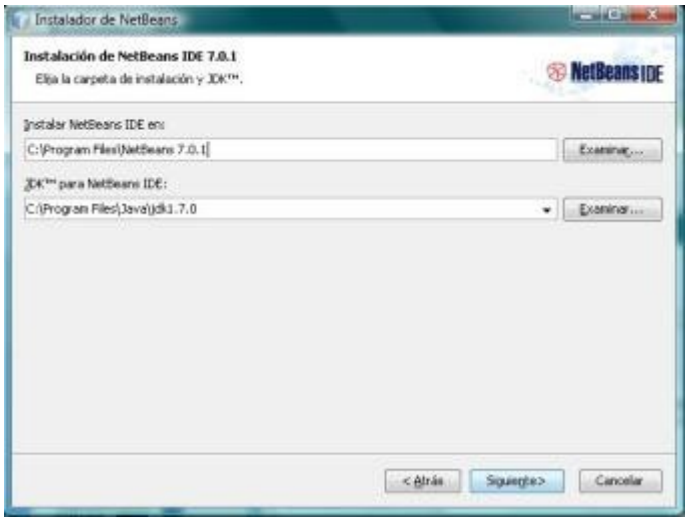
Comenzamos a instalar nuestro nuevo IDE

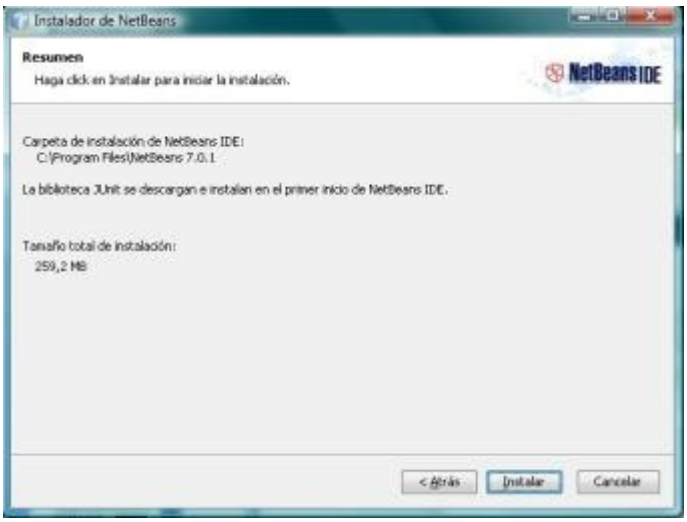
Aceptamos los términos de la licencia y pulsamos sobre *Siguiente*.



Aceptamos los términos de licencia para instalar JUnit(conjunto de *frameworks* que permite realizar la ejecución de clases Java de manera controlada) y pulsamos sobre *Siguiente*.

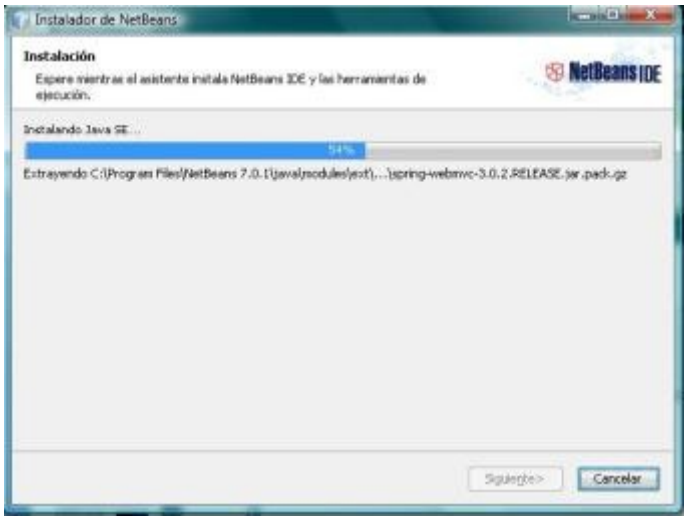
Dejamos los directorios de instalación por defecto y pulsamos sobre *Siguiente*.



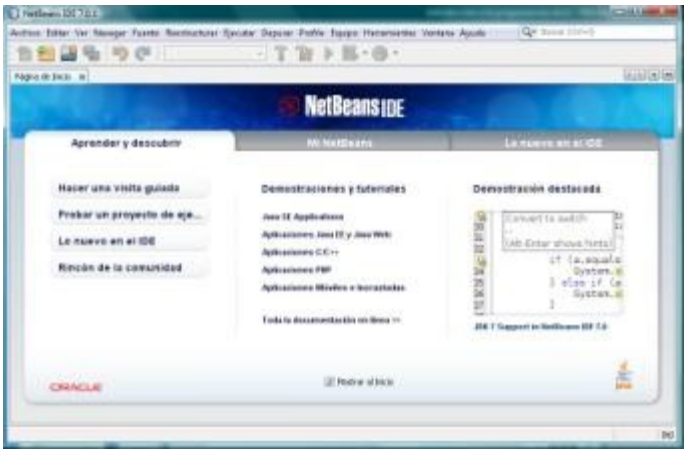


Tras ver un resumen de cuánto ocupará y dónde se instalará, pulsamos sobre Instalar.

Comienza el proceso de instalación

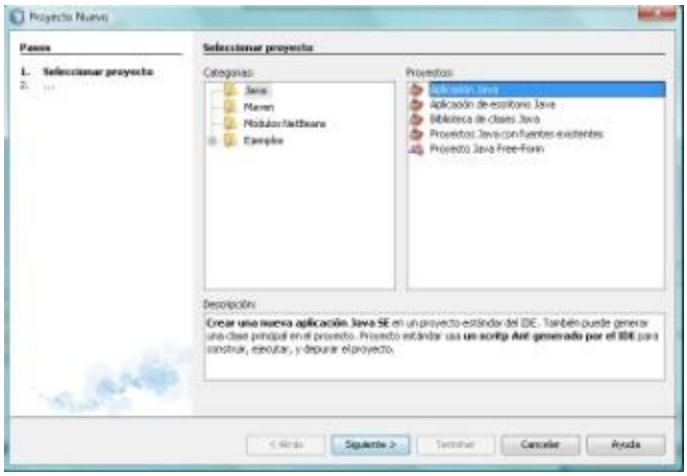


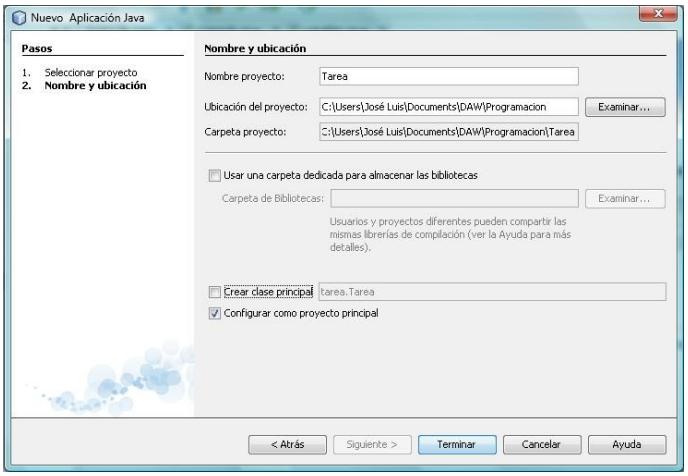
Tras informarnos que se ha completado la instalación, pulsamos sobre *Terminar*.



Cuando ejecutamos NetBeans por primera vez nos aparece ésta página de inicio.

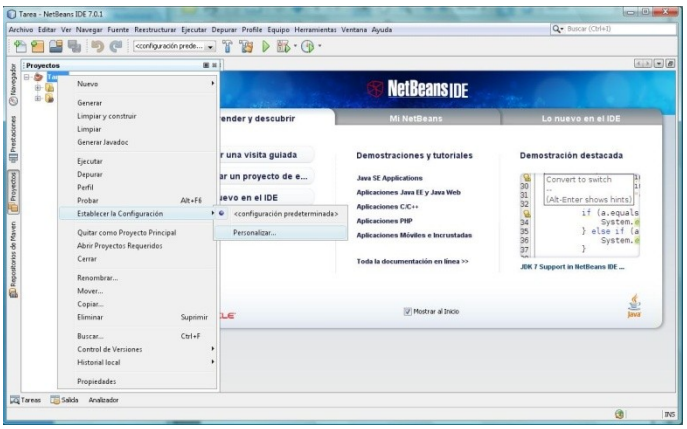
Pulsamos sobre *Archivo* en el menú principal y *Proyecto Nuevo*, con lo que accederemos a la ventana que nos aparece a la derecha.

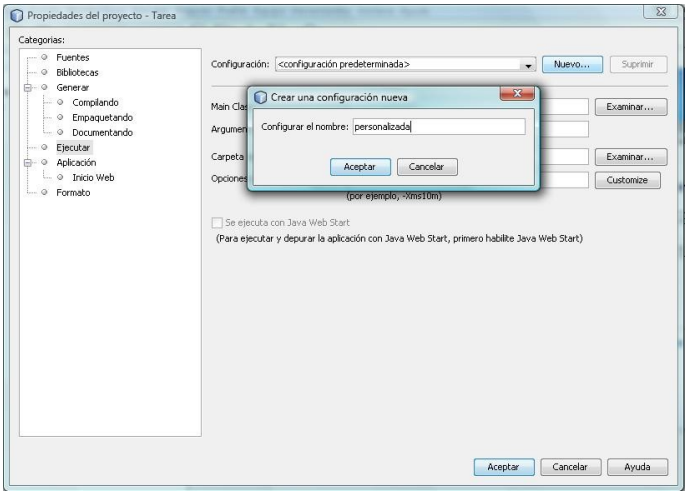




Llamamos *Tarea* al nuevo proyecto que vamos a crear y pulsamos *Terminar*

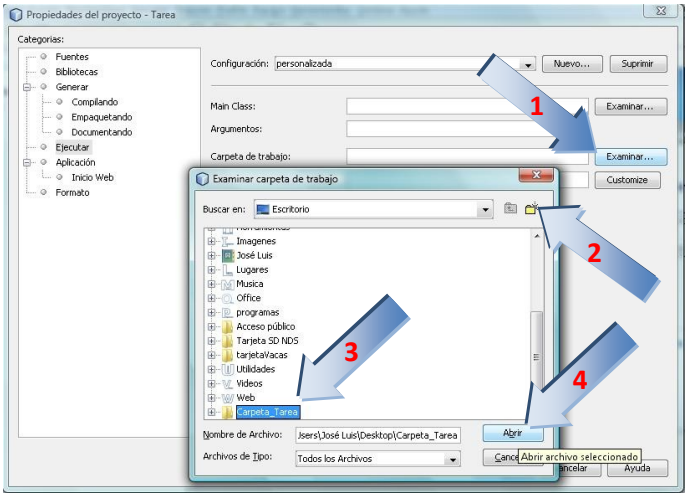
Pulsamos el botón derecho del ratón sobre el proyecto que hemos creado y seleccionamos *Establecer la configuración / Personalizar*

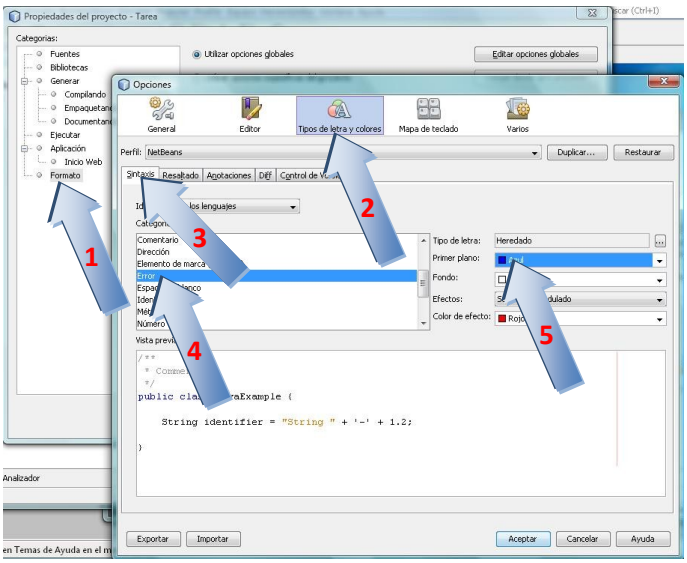




En el apartado *Ejecutar* pulsamos sobre el botón *Nuevo* y tecleamos *personalizada* como nombre para la nueva configuración.

Siguiendo en el apartado *Ejecutar* debemos asignar la carpeta de trabajo para nuestra nueva configuración, por lo que pulsaremos en el botón *Examinar*[1] del apartado *Carpeta de trabajo* y seleccionamos *Escritorio* en el apartado *Buscar en*, y a continuación pulsamos sobre el icono de crear nueva carpeta[2] y tecleamos *Carpeta_Tarea*[3] dándole finalmente a *Abrir*[4]

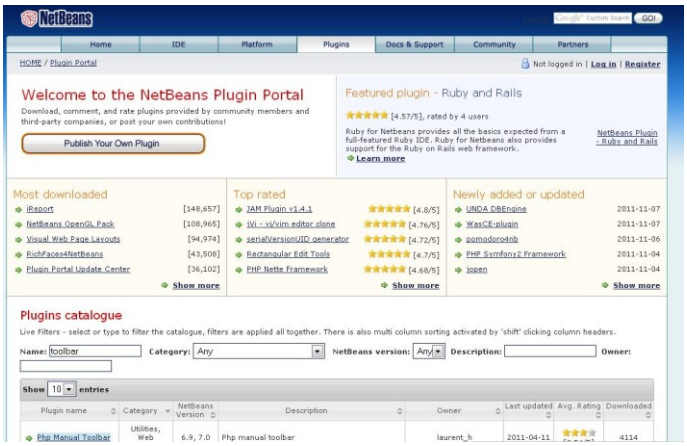




Para poner los mensajes de error en color azul debemos seleccionar *Fondo*[1] en *Categorías* y dentro de la nueva ventana que nos aparecerá seleccionaremos *Tipo de letra y colores*[2] y en la pestaña de *Sintaxis*[3] la categoría de *Error*[4] y Azul como color de *primer plano*[5].

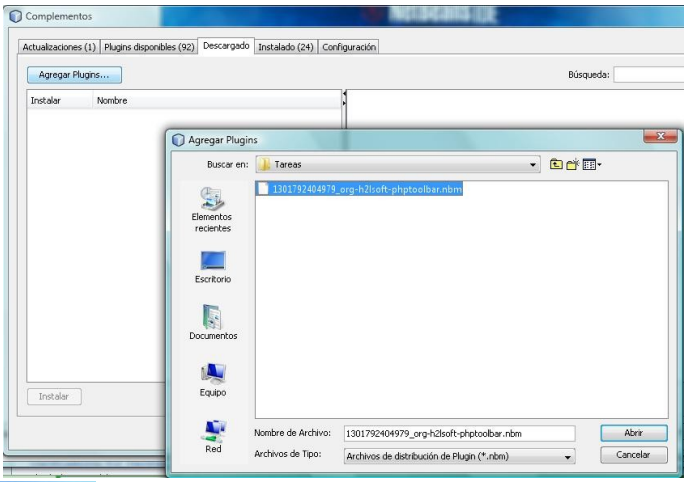
Ya sólo tendremos que darle al botón *Aceptar* para que se graben los cambios y tendremos finalizado este punto

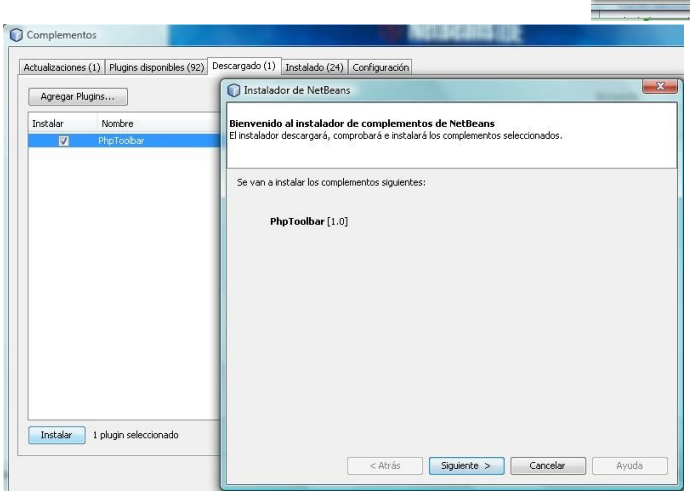
Ahora para instalar de forma *off-line* el plugin php-toolbar tendremos que ir de nuevo a la página de Netbeans y pulsar sobre la pestaña de *plugins*. Buscaremos plugins que tengan como parte de su nombre la palabra toolsbar y sólo nos aparece uno que se denomina *php manual toolbar* sobre el que pulsaremos para acceder a su descarga.



Pulsamos sobre *Dowload* y seleccionamos la carpeta donde deseamos guardar el fichero de instalación de dicho plugin.

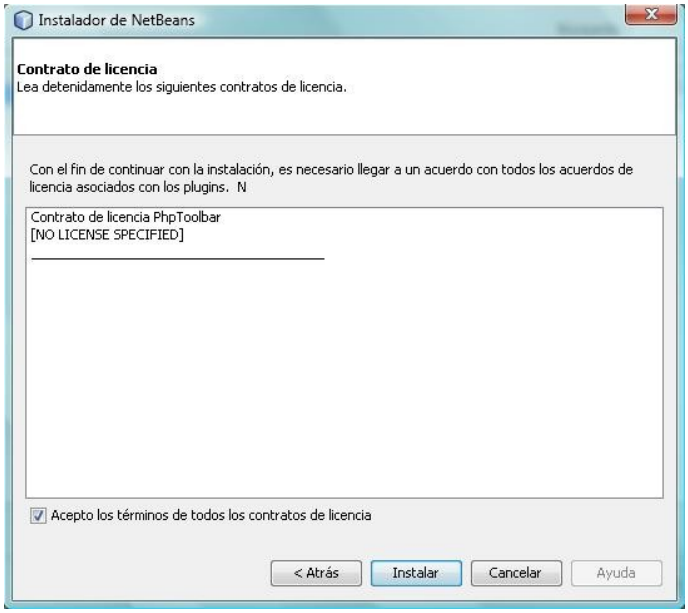
Después, dentro de Netbeans nos dirigimos al apartado de *Herramientas* y seleccionamos *Complementos*. Dentro de esta ventana nos dirigimos a la pestaña de *Descargado* y pulsamos sobre el botón *Agregar Plugins* y seleccionamos el fichero que hemos descargado anteriormente, y por último pulsamos sobre *Instalar*.

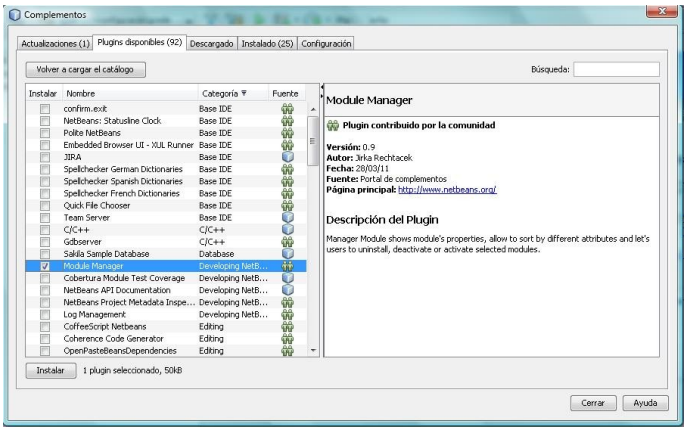




Le damos a siguiente en la nueva ventana que nos aparece y aceptamos los términos de la licencia.

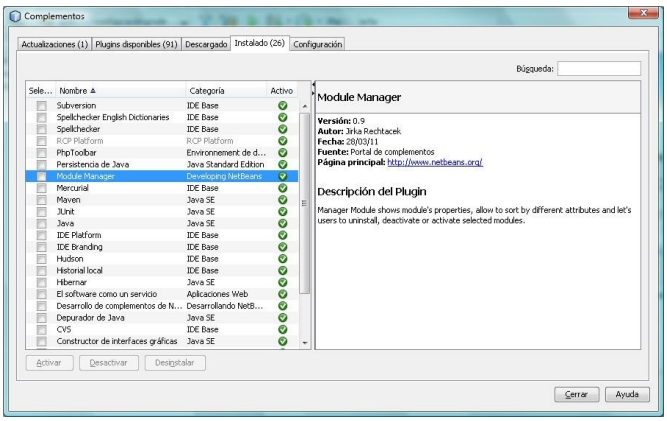
Pulsamos sobre *Instalar* y ya dispondremos del nuevo plugin *php toolsbar* instalado de forma off-line (lo hemos tenido que descargar para posteriormente instalarlo)





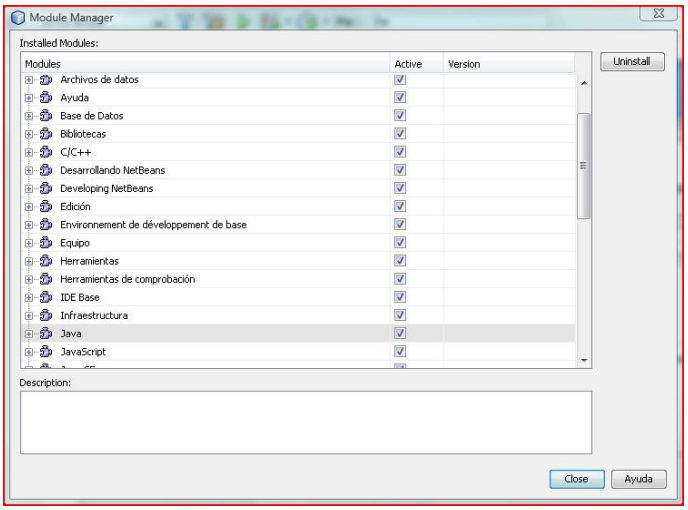
Ahora vamos a pasar a instalar, en esta ocasión de forma on-line, el plugin *Module Manager* para lo cual nos tendremos que dirigir de nuevo a *Herramientas* / Complementos y dentro de esta ventana nos situamos en la pestaña de *Plugins disponibles,* seleccionando *Module Manager* y pulsando sobre *Instalar*

Para comprobar que realmente lo hemos instalado bien nos dirigiremos ahora a la pestaña *Instalado* y veremos que nos aparece un nuevo plugin denominado *Module Manager* que está activo, por lo que la instalación se realizó correctamente.



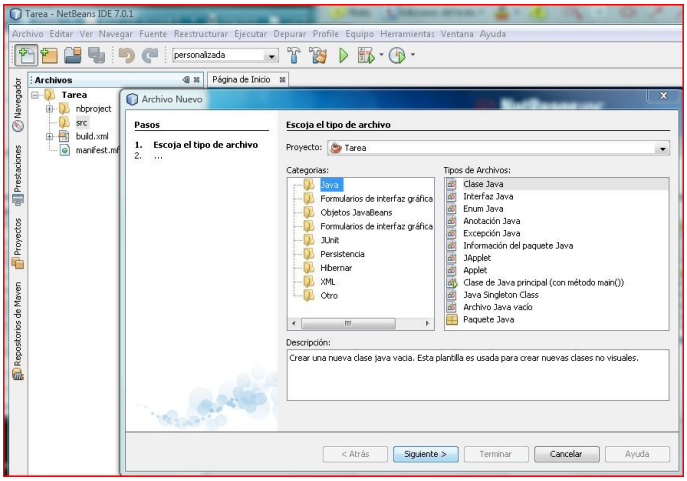
Con el primero de los plugins instalados (php toolsbar) conseguimos tener una especie de ayuda sobre cualquier mandato php, es decir, si teclea-mos cualquier orden del lenguaje php en la caja de búsqueda que nos aparece en la parte superior central precedida de la palabra php, conseguirémos acceder a una ayuda del uso de dicho término.

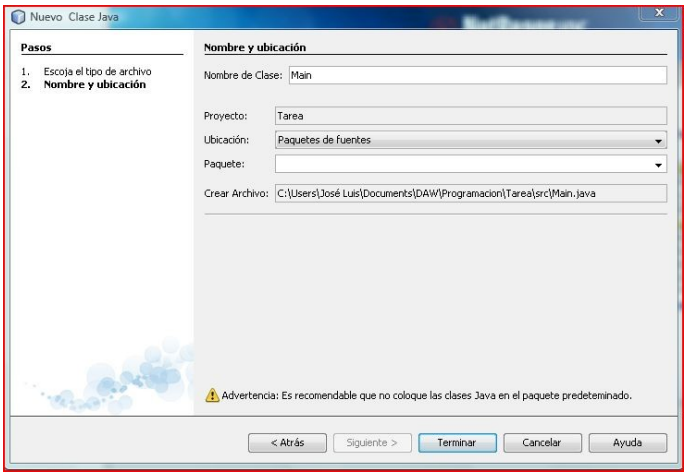
En el caso del plugin *Module Manager* tendremos un gestor de módulos que nos permitirá manejar de forma eficaz cualquier módulo instalado (Java, JavaScript, C++, etc), y que lo tendremos ubicado en el menú *Herramientas*.



Cuando accedemos a él nos aparece la ventana de la izquierda, donde podremos activar o desactivar cualquiera de los módulos que necesitemos

Ahora, para el apartado 5 de la tarea usaremos el proyecto creado para el segundo punto, por lo que pulsaremos sobre *Archivo / Archivo Nuevo* o con el atajo de teclado *Ctrl+N* o bien a través del icono  , tras lo cual aparecerá la pantalla de la derecha. Seleccionaremos *Clase Java* sobre el proyecto creado *Tarea* y pulsaremos sobre *Siguiente*.



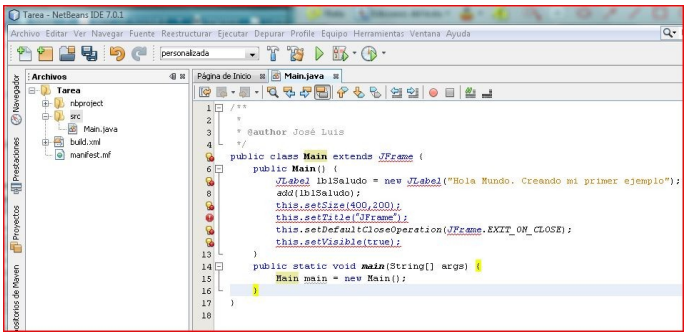


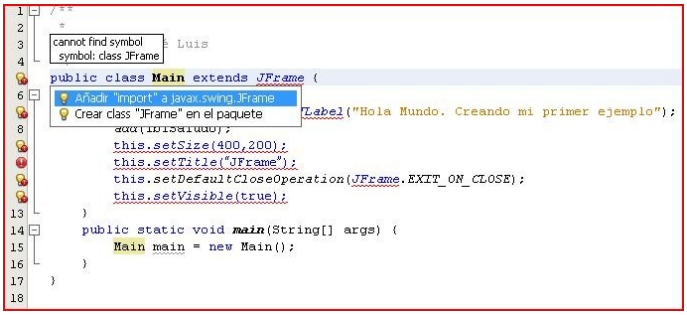
Pondremos como *Nombre de Clase* el nombre de *Main* ya que así es como se llama la clase del programa que hemos de realizar, dejando el resto de datos por defecto y pulsando sobre *Terminar*.

Al copiar el programa de la tarea nos aparecerá plagado de errores por varios motivos:

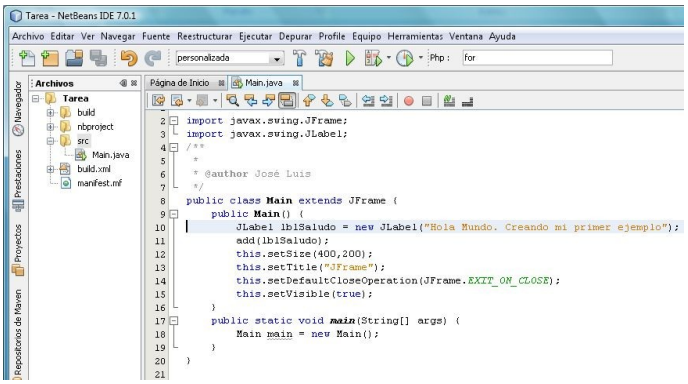
- No posee algunos paquetes
- Varias comillas son tipográficas

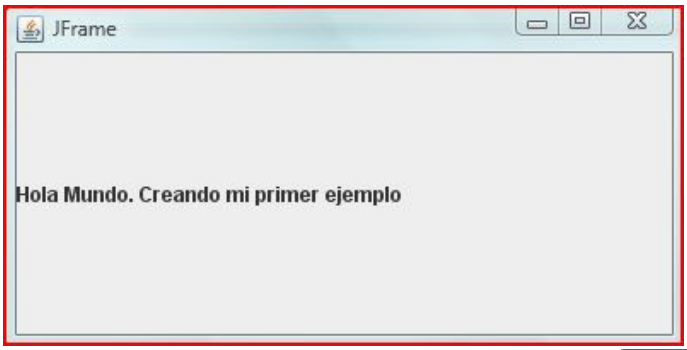
Por lo que tendremos que subsanar dichos errores antes de poder continuar.





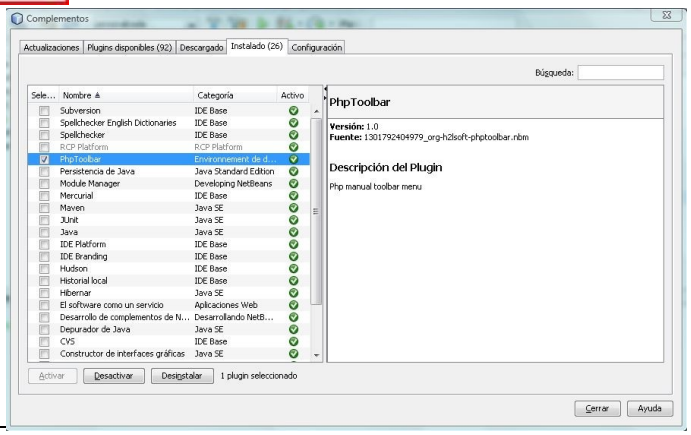
Si pulsamos con el ratón sobre el icono de la bombilla que nos aparece a la izquierda de ***public class Main extends Jframe{*** nos ofrece dos posibilidades para poder corregir el error de dicha línea. Seleccionamos la primera opción y nos añadirá de forma automática una línea al principio del código con el contenido ***import javax.swing.JFrame;*** con lo que se solucionará la mayoría de los problemas, aunque tendremos que hacer lo mismo para la etiqueta JLabel, y nos añadirá otra línea más con el contenido ***import javax.swing.JLabel;*** tras lo cual únicamente nos quedará cambiar las comillas tipográficas por comillas normales y tendremos nuestro programa terminado y sin errores como aparece en la imagen de la derecha

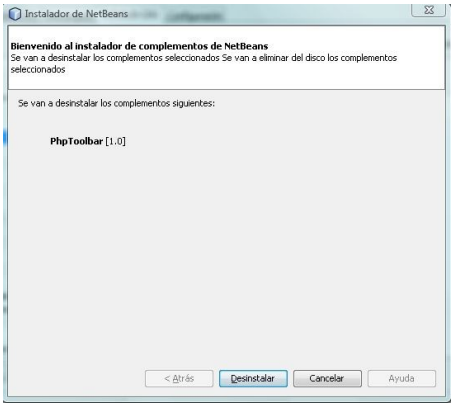




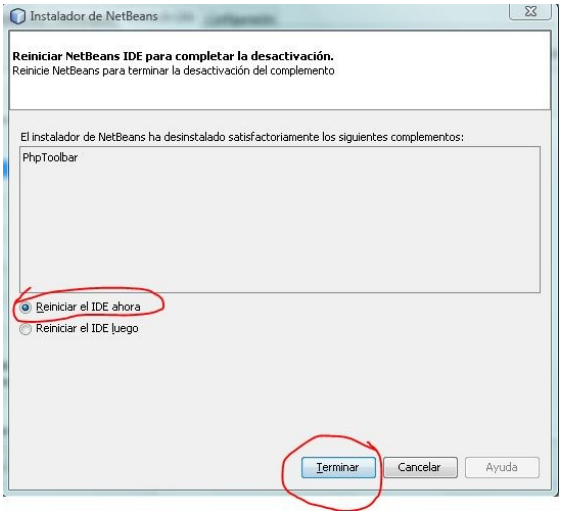
Ahora, con ***Ctrl+S*** grabaremos el programa y pulsaremos sobre la opción *Ejecutar* del menú superior y seleccionaremos *Compilar File* con lo que tendremos nuestro programa compilado y preparado para poder ejecutarlo cuando pulsemos ***Shift+F6*** apareciéndonos la pantalla de la izquierda.

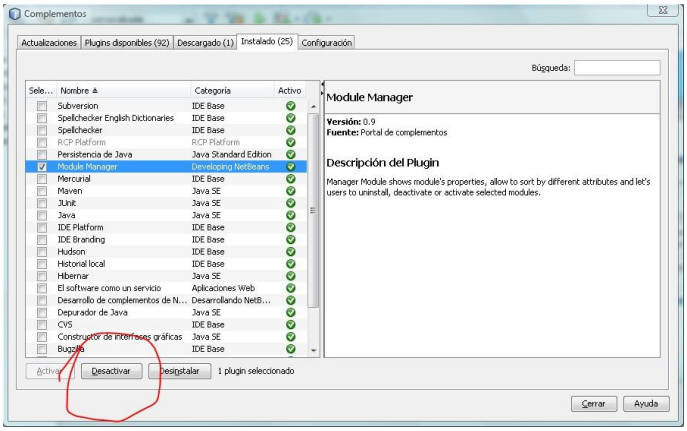
Ahora para desinstalar/desactivar los plugins de la presente tarea tendremos que ir a *Herramientas / Complementos* y en la pestaña *Instalados* seleccionamos primero la casilla de phpToolbar y pulsamos sobre desinstalar





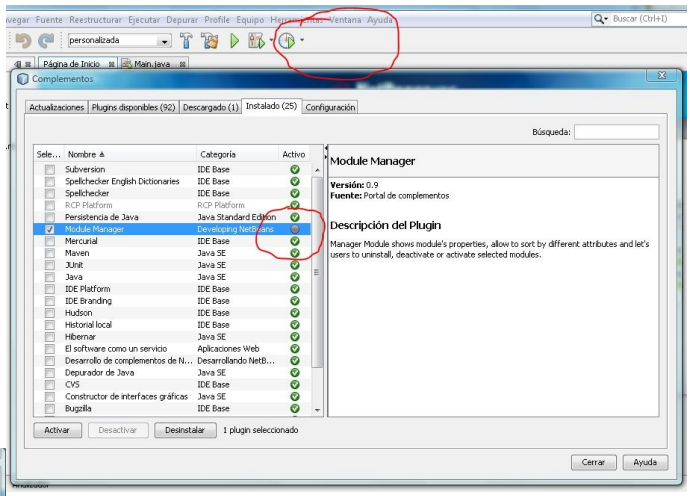
Pulsamos sobre ***Desinstalar*** y nos aparecerá la pantalla de la derecha. Seleccionamos *Reiniciar el IDE ahora* y pulsamos sobre *Terminar*, tras lo cual se reinicia NetBeans con el plugin phpTools eliminado de su configuración.

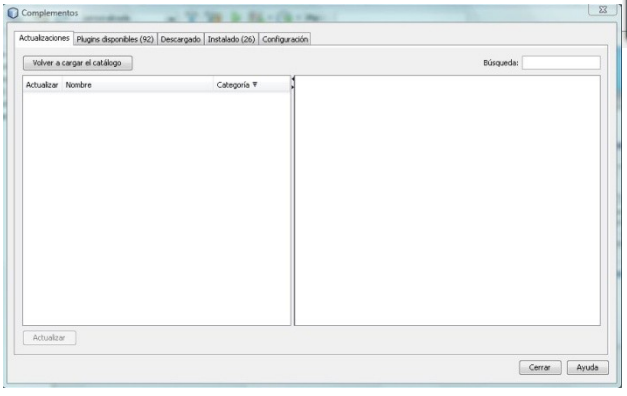




De nuevo nos dirigimos a *Herramientas / Complementos* y sobre la pestaña de *Instalado* vemos que ya no aparece el anterior plugin, por lo que sólo nos queda desactivar el Module Manager. Seleccionamos dicho plugin y en esta ocasión pulsamos sobre *Desactivar* y tras aceptar la pantalla que nos aparece a continuación sólo tendremos que decidir de nuevo reiniciar el IDE para que ya empiece a tener efecto la desactivación del plugin.

Ahora si nos vamos a *Herramientas / Complementos* veremos que no aparece ya phpToolsbar y que Module Manager está desinstalado por lo que si lo seleccionamos únicamente se nos activarán los botones de *Desactivar* y *Desinstalar*.





Como durante todo el proceso no nos ha aparecido ningún aviso de actualización, podremos comprobar si existe alguna disponible para cualquiera de los plugins que tenemos instalados volviendo a *Herramientas / Complementos* pero yéndonos en esta ocasión a la pestaña de *Actualizaciones* donde pulsaremos en *Volver a cargar el catálogo*. Si hubiese alguna

actualización disponible nos la presentaría para que pudiéramos marcarla y pulsar sobre el botón de *Actualizar*. En mi caso no existe ninguna disponible por lo que me aparece la pantalla en blanco.