



DIPLOMADO PROGRAMADOR EN JAVA.

Instructor: Giovanni Ariel Tzec Chávez



Perfil de ingreso

- ✓ Conocimientos básico de: Internet, HTML y SQL estándar.
- ✓ Conocimiento de la lógica de programación.
- ✓ Conocimiento en lenguaje de programación.
- ✓ Bases de programación orientada a objetos.

Diplomado programador JAVA

Java nivel 2 (Java intermedio)

Java nivel 3 (Java avanzado)

Java nivel 4 (J2EE)

Java nivel 5 (J2EE, hibernate)

Java nivel 6 (JSF)

Java nivel 7 (WebServices)

Java nivel 8 (Tecnologías JSON)

Java Intermedio(Modulo 2)

- ✓ Estándares de Programación en JAVA
- ✓ Acceso a bases de datos desde Java – JDBC
- ✓ Clases Driver, Connection, Statement, ResultSet
- ✓ Sentencias de SQL en JAVA
- ✓ Se agregará la temática de Java Swing y POO

Retroalimentando clase final

✓POO

Objetivo de la sesión

Aplicar estándares de programación de Java

Distribución de tiempo

Módulos	Duración
Java nivel 2	20 horas

Contenido

- ✓ Estándares de programación en JAVA propuestos por SUN Microsystem.
- ✓ Nombres y organización de Ficheros.
- ✓ Ficheros Fuente de Java
- ✓ Indentación - Comentarios- Declaraciones- Sentencias- Espacios en Blanco.
- ✓ Convenciones de nombrado.
- ✓ Prácticas de programación
- ✓ Prácticas Misceláneas
- ✓ Comentarios especiales
- ✓ Ejemplo Código Java

Nombre y organización para ficheros

Un fichero consiste de secciones que deben estar separadas por líneas en blanco y comentarios opcionales que identifican cada sección.

Los ficheros de más de 2000 líneas son incómodos y deben ser evitados.

Extensiones de los ficheros





Tipo de fichero	Extensión
Fuente Java	<code>.java</code>
Bytecode de Java	<code>.class</code>

Ficheros fuente Java

Cada fichero fuente Java contiene una única clase o interface pública.

Los ficheros fuentes Java tienen la siguiente ordenación:

- Comentarios de comienzo
- Sentencias package e import
- Declaraciones de clases e interfaces.

```
1   /*  
2      * To change this license header, choose License Headers in Project Properties.  
3      * To change this template file, choose Tools | Templates  
4      * and open the template in the editor.  
5  */  
6  package conocimientoprogramadorjava;  
7  
8   /**  
9      *  
10     * @author Giovanni Tzec  
11 */  
12 public class ConocimientoProgramadorJava {  
13  
14      /**  
15         * @param args the command line arguments  
16     */  
17      public static void main(String[] args) {  
18         // TODO code application logic here  
19     }  
20  
21 }  
22 |
```

Comentarios de inicio

Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha, y copyright:

```
/*  
* Nombre de la clase  
*  
* Información de la versión  
*  
* Fecha  
*  
* Copyright  
*/
```

```
1
2  package conocimientoprogramadorjava;
3
4  /**
5   * Nombre de la clase: Clase ConocimientoProgramadorJava
6   * Versión: 1.0
7   * Fecha: 27-06-2017
8   * Copyright ITCA-FEPADE
9   * @author Giovanni Tzec
10  */
11  public class ConocimientoProgramadorJava {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18      }
19
20  }
```


Sentencias package e import

La primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import. Por ejemplo:

```
1
2  package conocimientoprogramadorjava;
3  import java.util.Scanner;
4  import java.text.DecimalFormat;
5
```

Indentación

Se deben emplear cuatro espacios como unidad de indentación. La construcción exacta de la indentación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada 8 espacios (no 4).

Longitud de la línea

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

Nota: Ejemplos para uso en la documentación deben tener una longitud inferior, generalmente no más de 70 caracteres.

Romper líneas

Cuando una expresión no entre en una línea, romperla de acuerdo con estos principios:

- ✓ Romper después de una coma.
- ✓ Romper antes de un operador.
- ✓ Preferir roturas de alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").
- ✓ Alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.
- ✓ Si las reglas anteriores llevan a código confuso o a código que se aglutina en el margen derecho, indentar justo 8 espacios en su lugar. **Explicar ejemplo**

Formatos de comentarios de implementación

Los comentarios de bloque se usan para dar descripciones de ficheros, métodos, estructuras de datos y algoritmos. Los comentarios de bloque se podrán usar al comienzo de cada fichero o antes de cada método. También se pueden usar en otro lugares, tales como el interior de los métodos. Los comentarios de bloque en el interior de una función o método deben ser indentados al mismo nivel que el código que describen.

```
18  public static void main(String[] args) {  
19      /*  
20      *Comentario de bloque  
21      */  
22  }
```


Comentarios

Comentarios de una línea `/* Comentario */`

`//Comentario de una línea`

Comentarios de remolque

```
30      if(9>5)
31      {
32          System.out.println("Verdadero");           //Devolvera verdadero
33      }
34      else
35      {
36          System.out.println("Verdadero");           //Devolvera falso
37      }
```

Comentarios de fin de línea

Comentarios de documentación

Los comentarios de documentación describen clases Java, interfaces, constructores, métodos y atributos. Cada comentario de documentación se encierra con los delimitadores de comentarios `/**...*/`, con un comentario por clase, interface o miembro (método o atributo).

```
13
14  /**
15   * La clase ConocimientoProgramadorJava ofrece ...
16   */
17  public class ConocimientoProgramadorJava {
18
19      /**
20       * @param args the command line arguments
21       */
22      public static void main(String[] args) {
23          /*
```

Darse cuenta de que las clases e interfaces de alto nivel estan indentadas, mientras que sus miembros los están. La primera línea de un comentario de documentación (/**) para clases e interfaces no esta indentada, subsecuentes líneas tienen cada una un espacio de indentación (para alinear verticalmetne los asteriscos). Los miembros, incluidos los constructores, tienen cuatro espacios para la primera línea y 5 para las siguientes.

Declaraciones

Se recomienda una declaración por línea, ya que facilita los comentarios. En otras palabras, se prefiere

- `int nivel; // nivel de indentación`
- `int tam; // tamaño de la tabla`
- antes que: `int level, size;`
- No poner diferentes tipos en la misma línea.
- Ejemplo: `int foo, fooarray[]; //ERROR!`

Declaraciones de clases e interfaces

Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros

La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración

La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{"

Espacios en blanco

Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.

Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:

- Entre las secciones de un fichero fuente.
- Entre las definiciones de clases e interfaces.
- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
 - Entre métodos.
 - Entre las variables locales de un método y su primera sentencia.
 - Antes de un comentario de bloque o de un comentario de una línea
 - Entre las distintas secciones lógicas de un método para facilitar la lectura.

POO

Es un enfoque conceptual para diseñar programas, utilizando un lenguaje de programación orientado a objetos.

Sus propiedades más importantes son:

- Abstracción.
- Encapsulación y ocultación de datos.
- Polimorfismo, sobrecarga de métodos.
- Herencia.
- Reusabilidad o reutilización de código.

Este paradigma de programación supera las limitaciones de la programación tradicional o procedimental. Es otra forma de pensar....Otra forma de programar.!!!

Clases

- Clases
- Atributos
- Métodos u operaciones
- Visibilidad
- Constructores de la clase
- Métodos de acceso
- Objetos
- Relaciones
- Convenciones de nombres

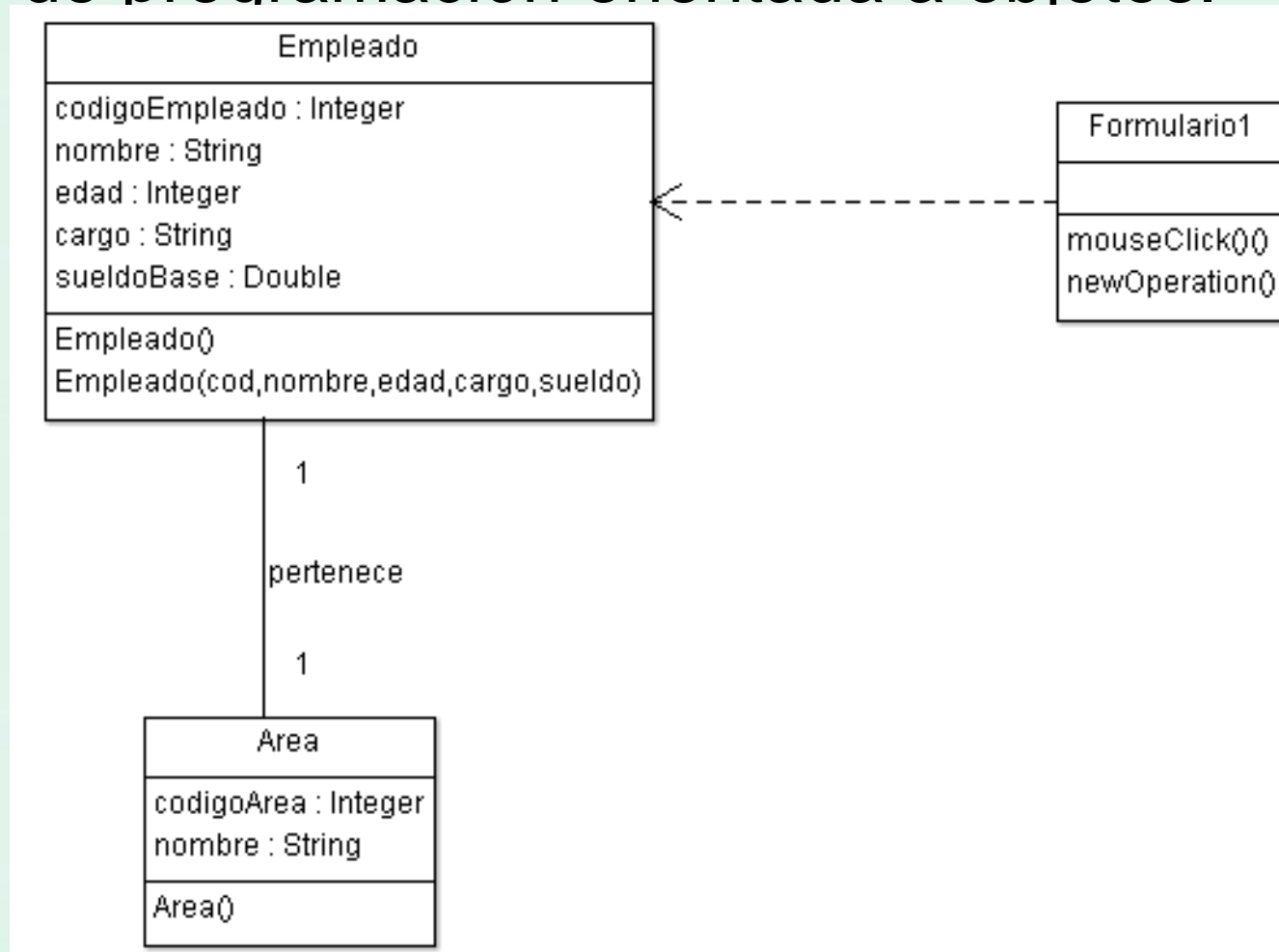
Persona
dui : Integer nombre : String edad : Integer
Persona() Persona(dui : Integer,nombre : String,edad : Integer) getDui() : Integer getNombre() : String getEdad() : Integer setNombre(nombre : String) setDui(dui : Integer) setEdad(edad : Integer) imprimir() imprimir(dui : Integer,nombre : String,edad : Integer) imprimir(edad : Integer)

Convenciones de nombres en Java

Tipo	Regla	Ejemplo
Paquetes	Se escriben en minúscula. Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. Dichas convenciones pueden especificar que algunos nombres de los directorios correspondan a divisiones, departamentos, proyectos o máquinas.	com.sun.es com.apple.quicktime.v2 edu.itca.finanzas.compras
Clases	Inicial con mayúscula. Si es palabra compuesta la siguiente con inicial mayúscula.	Empleado EmpleadoFinanzas
Interfaces	Regla de las clases	
Atributos y métodos	Inicial con minúscula. Si es palabra compuesta la siguiente mayúscula	nombre nombreParticipante insertar insertarEmpleado
Constantes	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separando las palabras con un subguión ("_").	static final int ANCHURA_MINIMA A = 4;

Ejercicio propuesto

Codificar el siguiente diagrama de clases tomando en cuenta los estándares de programación orientada a objetos.



Java Swing

Una interfaz gráfica de usuario (GUI) presenta un mecanismo amigable al usuario para interactuar con una aplicación. Una GUI proporciona a una aplicación una “apariencia visual” única. Al proporcionar distintas aplicaciones en las que los componentes de la interfaz de usuario sean consistentes e intuitivos, los usuarios pueden familiarizarse en cierto modo con una aplicación, de manera que pueden aprender a utilizarla en menor tiempo y con mayor productividad.

La mayoría de las aplicaciones que usamos a diario utilizan ventanas o cuadros de diálogo (también conocidos como diálogos) para interactuar con el usuario. Por ejemplo, los programas de correo electrónico le permiten escribir y leer mensajes en una ventana que proporciona el programa.

JOptionPane de Java (paquete javax.swing)

Proporciona cuadros de diálogo pre empaquetados, utiliza dos diálogos de entrada para obtener datos del usuario y un diálogo de mensaje para mostrar el resultado esperado para el usuario.

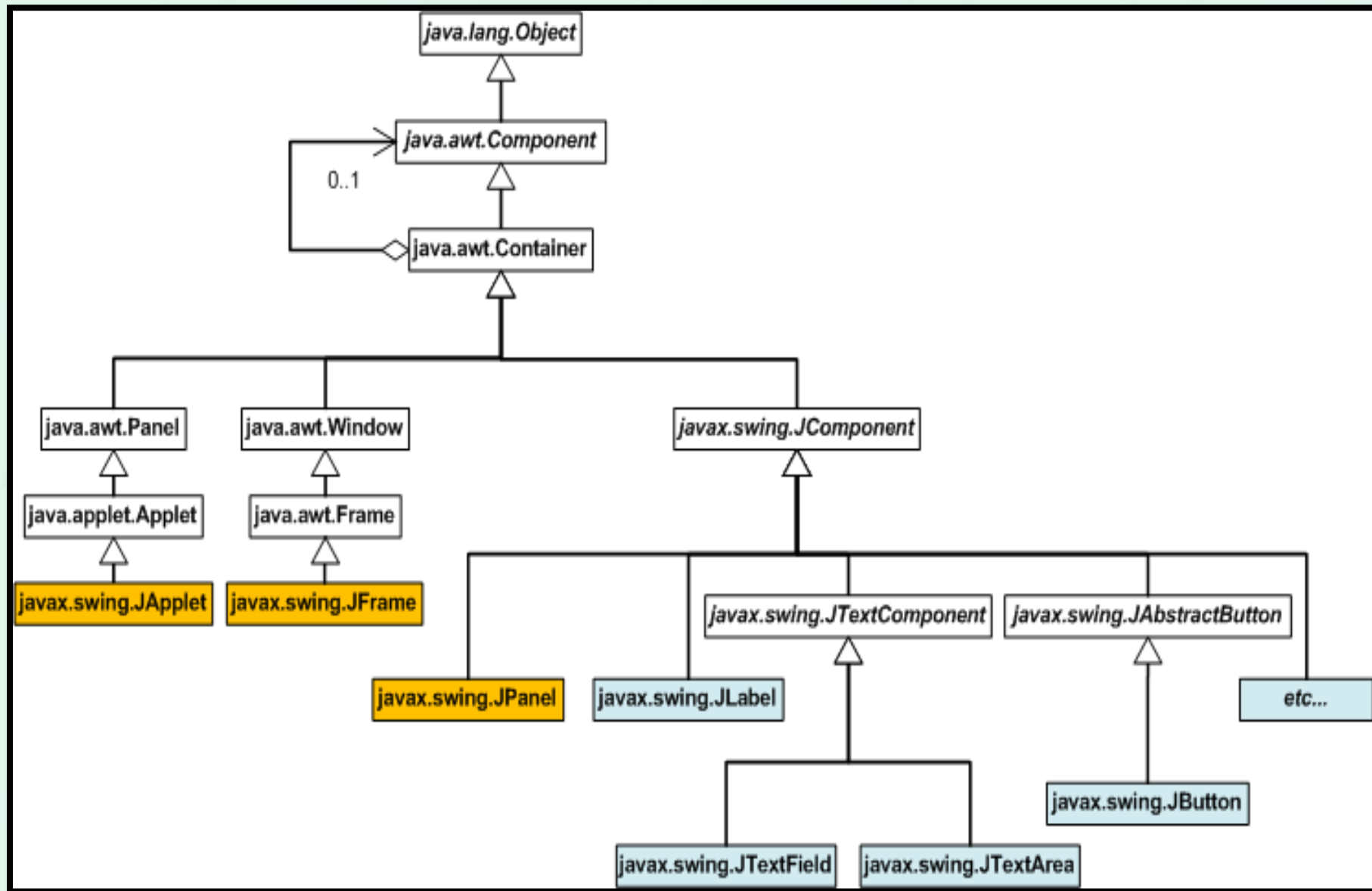
Ver ejemplo “IntroduccionGrafica”

Desarrollar ejercicio.

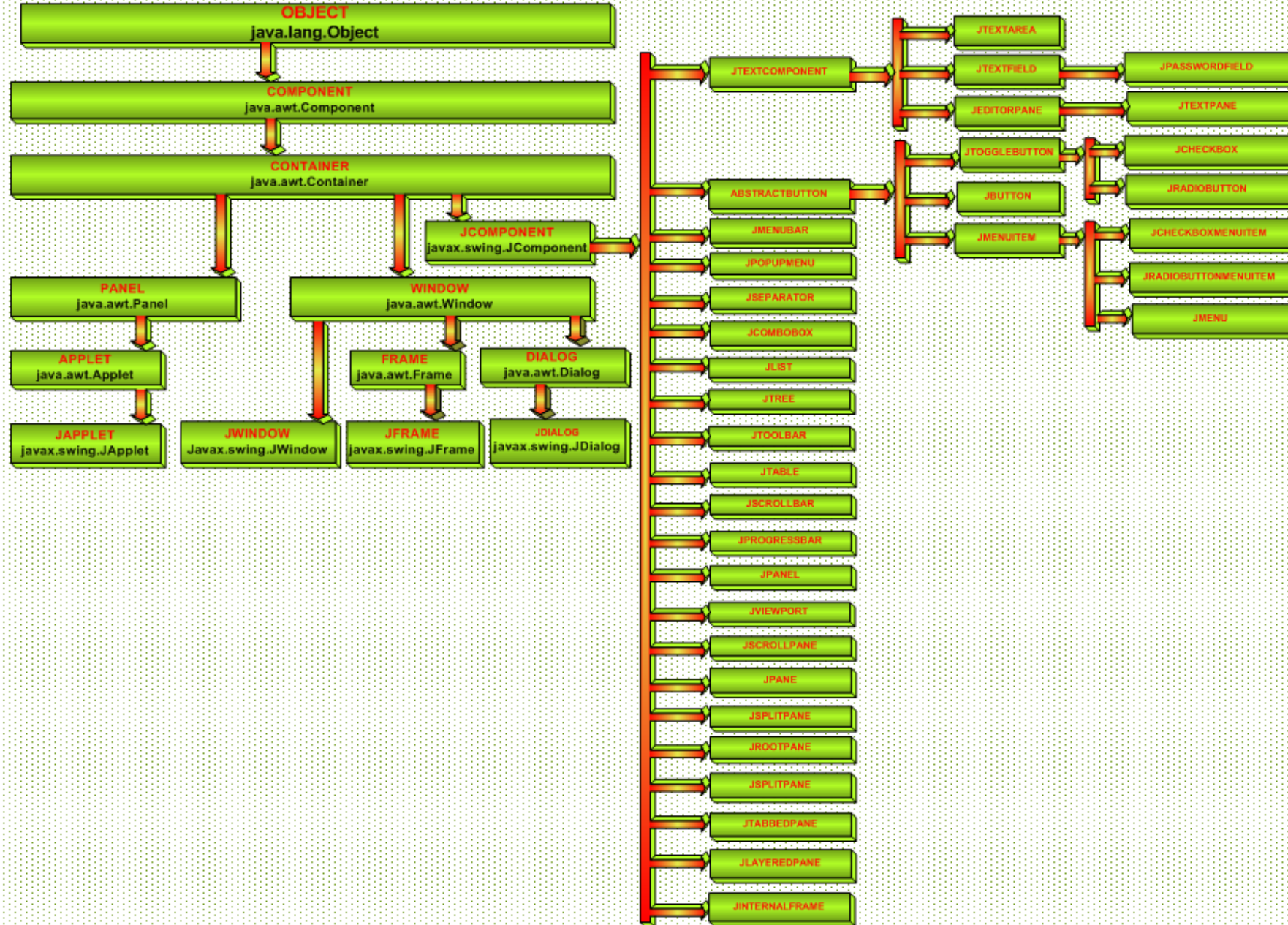
Escriba un programa en Java que reciba a través de un cuadro de diálogo los datos de un estudiante (código, nombre, carrera, edad, nota final), evaluar los siguientes criterios de la nota(1 a 2 NM, 2 a 4 Regular, 4 a 6 Bueno, de 6 a 8 Muy bueno, de 8 a 10 Excelente)

Comparación entre Swing y AWT

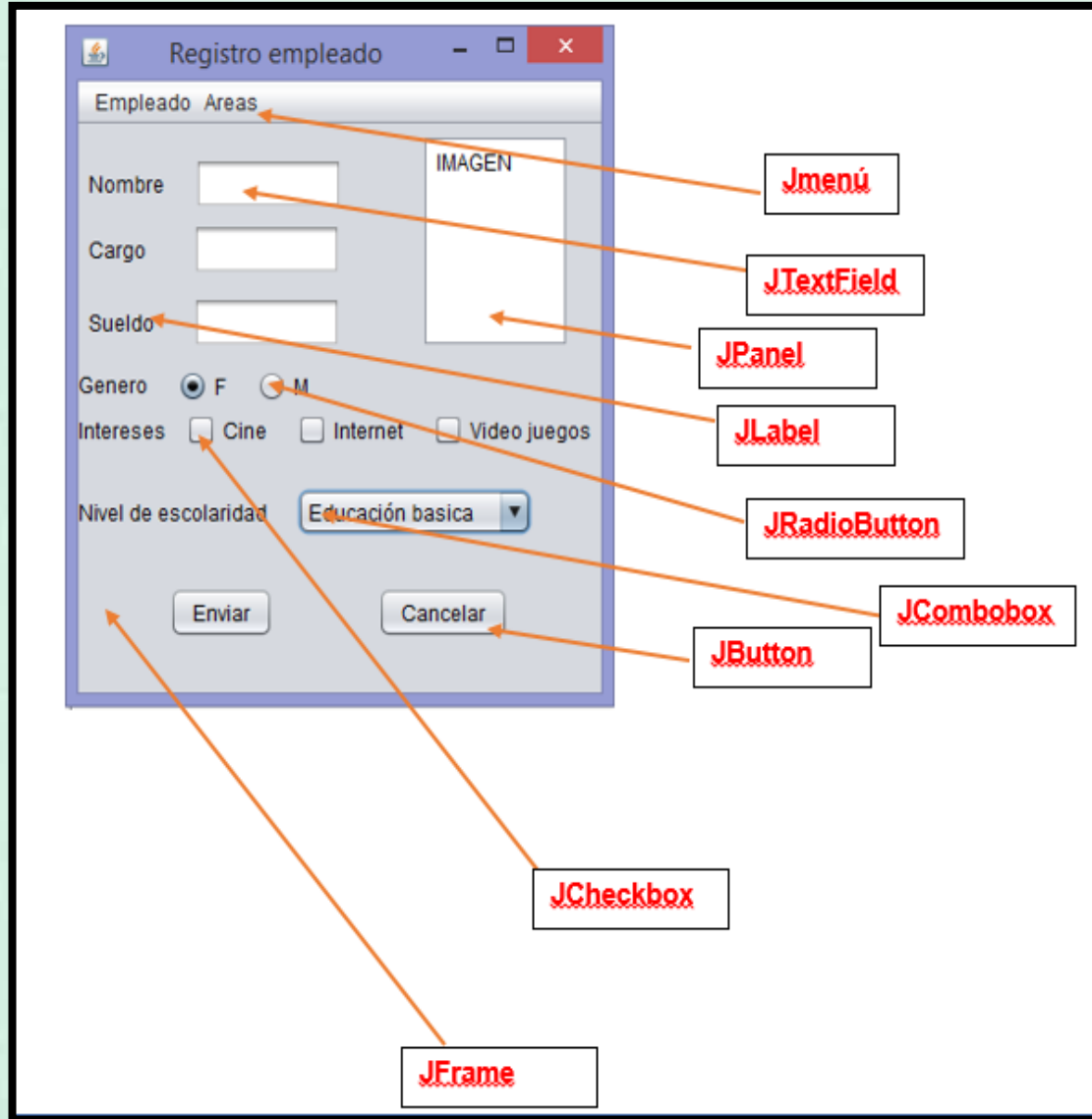
- En realidad hay dos conjuntos de componentes de GUI en Javas. Antes de introducir a Swing en Java SE 1.2, las GUIs de Java se creaban a partir de componentes del Abstract Window Toolkit (AWT) en el paquete `java.awt`.
- Cuando una aplicación de Java con una GUI del AWT se ejecuta en distintas plataformas, los componentes de la GUI de la aplicación se muestran de manera distinta en cada plataforma. Considere una aplicación que muestra un objeto de tipo `Button` (paquete `java.awt`).
- Los componentes de GUI de Swing nos permiten especificar una apariencia visual uniforme para una aplicación a través de todas las plataformas, o para usar la apariencia visual personalizada de cada plataforma.
- Incluso, hasta una aplicación puede modificar la apariencia visual durante la ejecución, para permitir a los usuarios elegir su propia apariencia visual preferida.



JAVA SWING COMPONENTS HIERARCHY



Ejemplo de formulario SWING



Forma de acceder a la documentación de JAVA puede ser consultada a través de Internet, en la dirección:

<http://docs.oracle.com/javase/7/docs/api/>

Swing Parte II

JFrame

JLabel

JTextField

JButton

JComboBox

JRadioButton

JCheckBox

Ver ejemplo con clases

Ejercicio propuesto.

Estudiante, Carrera.

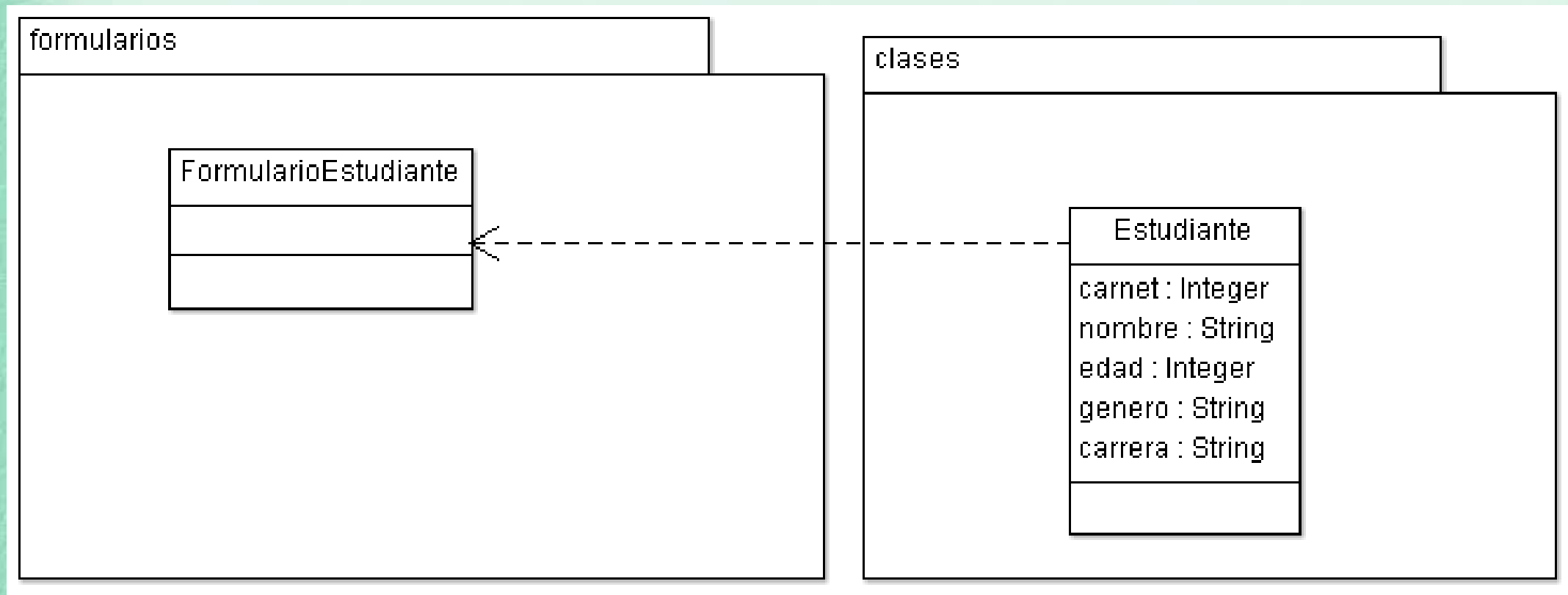
Estudiante(Carnet AA343318), nombre validado, edad, genero, facultad, teléfono móvil, teléfono fijo, dirección, cum de ciclo.

Imprimir resultados y evaluar que si el cum supera el 9.0 es sobresaliente, superior a 8.0 muy bueno, inferior a 7.0 debe de cursar materias adicionales para poder graduarse.

Carrera código, nombre.

Botón enviar y limpiar.

Codificar el siguiente diagrama de clases



Bibliografía.

Piensa en Java 2 Edicion Spanish

Aprenda Java Desde Cero.

Preguntas y respuestas

