Título: Técnico Superior en Desarrollo de Aplicaciones Web

Módulo profesional: 485- Programación

Unidad didáctica 9. Control de Errores. Excepciones. Entorno Gráfico

12 horas previstas

Instituto de Educación Secundaria "José Mor de Fuentes"
IFC INFORMÁTICA Y COMUNICACIONES

Unidad didáctica 9

- **Instrumentos de evaluación**
  - Unidad didáctica
    - Prácticas (20%)
    - Proyecto (20%)
    - Ejercicio práctico fin de unidad (60%)
      - Test 20%
      - Prueba conocimientos 20%
  - Prueba de conocimientos (fin de trimestre)

Materiales y métodos

- 01-UD9-ExcepcionesYGUI(transparencias de “teoría)
- 02-UD9-Tareas.pdf. Ejercicios
- Netbeans
- Debéis crear un directorio nuevo llamado UD9 y varios proyectos:
  - 01_EjemplosUD9. Para los ejemplos de clase y los que colguemos en Aramoodle
  - 02_EjerciciosUD9. Para los ejercicios de 02-UD9-Tareas.pdf (prácticas de la unidad, cuentan en nota final)

Parte I. GUI (Graphical User Interface

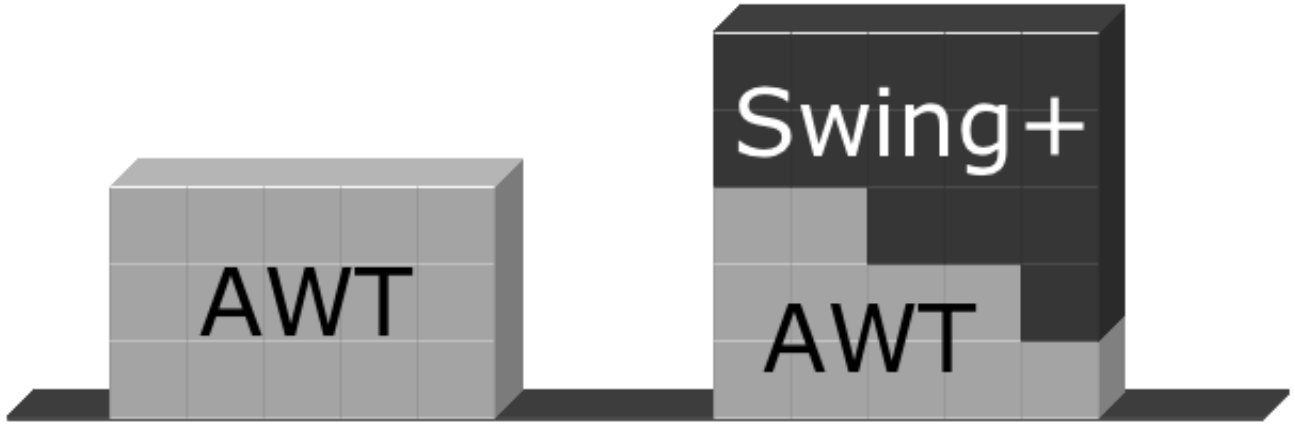
- GUI: Graphical User Interface
- Componentes de un GUI
- Disposiciones/Layouts
- Manejo de Eventos

- GUI: Graphical User Interface
- Componentes de un GUI
- Disposiciones/Layouts
- Manejo de Eventos

Principios de un GUI

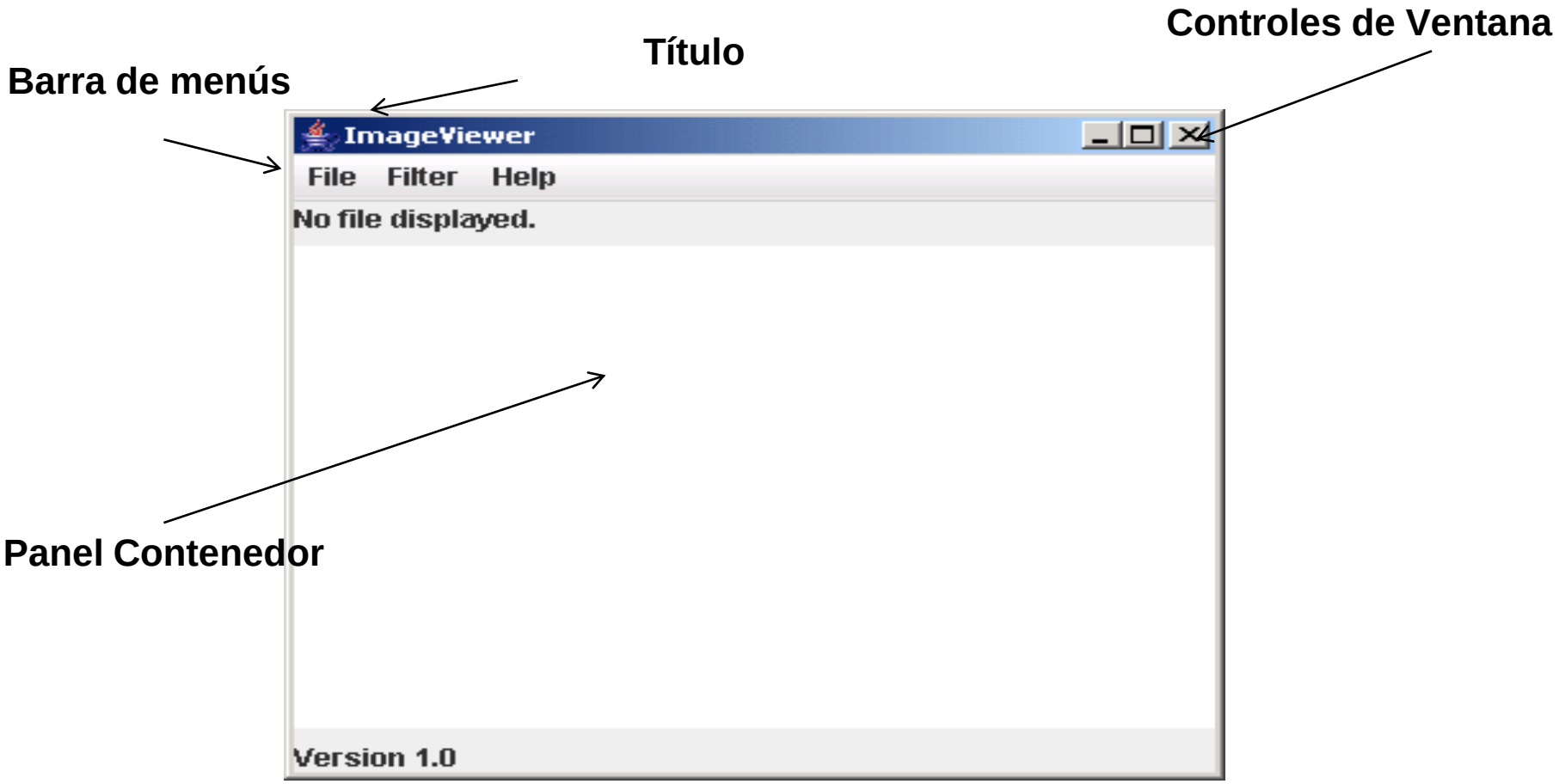
- Componentes:: Bloques para construir un GUI: botones, menús, barras de desplazamiento, etc,
- Layout/Disposición: Cómo disponer los componentes para formar un GUI usable (se utilizan los layout managers)
- Eventos: reacciones a la entrada/interacción del usuario (pulsación de un botón, selección de menú, etc,...)

AWT y Swing



- Swing Utiliza librerías de AWT
- Swing tiene algunas clases que tenía ya AWT (y que se pueden diferenciar porque empiezan por J. (por ejemplo Frame pertenece a AWT y Jframe a Swing)

Elementos de un frame (marco)



Creación de un frame

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ImageViewer
{
    private JFrame frame;

    /**
     * Crea un ImageViewer y lo muestra por pantalla
     */
    public ImageViewer()
    {
            makeFrame();
    }

    // resto de la clase omitido
}
```

Panel de contenido (content pane)

```
/**
 * Crea el frame Swing y su contenido
 */
private void makeFrame()
{
    frame = new JFrame("ImageViewer");
    Container contentPane = frame.getContentPane();

    JLabel label = new JLabel("I am a label.");
    contentPane.add(label);

    frame.pack();
    frame.setVisible(true);
}
```

Añadir menús

- JMenuBar
  - **Se muestra bajo el título**
  - **Contiene los menús**
- JMenu
  - **Contiene los items de menú**
- JMenuItem
  - **Contiene items de menú individuales**

Menús

```
private void makeMenuBar(JFrame frame)
{
    JMenuBar menubar = new JMenuBar();
    frame.setJMenuBar(menubar);

    // create the File menu
    JMenu fileMenu = new JMenu("File");
    menubar.add(fileMenu);

    JMenuItem openItem = new JMenuItem("Open");
    fileMenu.add(openItem);

    JMenuItem quitItem = new JMenuItem("Quit");
    fileMenu.add(quitItem);
}
```

Diálogos

- Los diálogos modales bloquean cualquier otra interacción con el usuario: fuerzan una respuesta del mismo
- Los diálogos no modales permiten otro tipo de interacción

JoptionPane diálogos estándar

- Message dialog
  - Mensaje de texto y un botón OK
- Dialogo Confirm
  - Si, No, Cancelar
- Dialogo de entrada
  - Mensaje de texto y un campo de entrada
- Puede haber variaciones

Un mensaje de diálogo

```
private void showAbout()
{
    JOptionPane.showMessageDialog(frame,
                "ImageViewer\n" + VERSION,
                "About ImageViewer",
                JOptionPane.INFORMATION_MESSAGE);
}
```

Bordes

- Se añaden para decorar componentes
- Definidos en javax.swing.border
- `BevelBorder`, `CompoundBorder`, `EmptyBorder`, `EtchedBorder`, `TitledBorder`.

Otros componentes

- Slider
- Spinner
- Tabbed pane
- Scroll pane

SWING

Tenemos dos tipos de elementos:
- Contenedores.
- Componentes / controles. Pueden ser también contenedores.

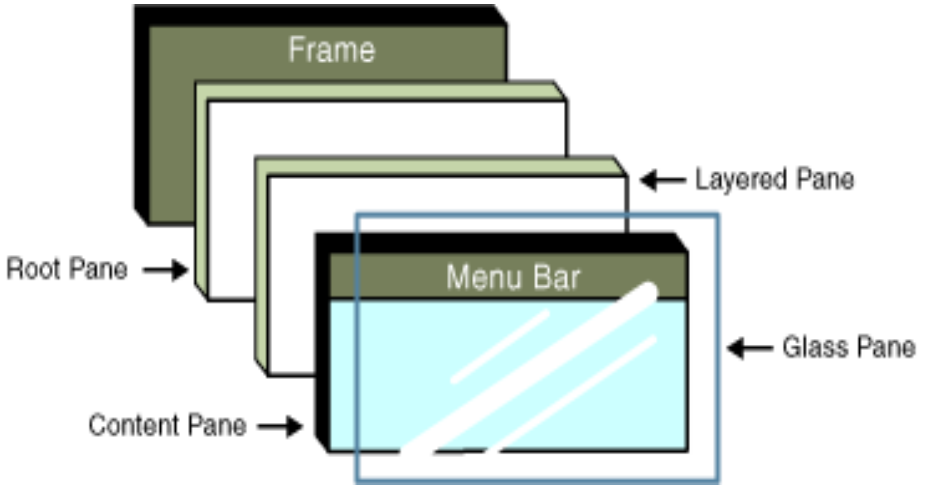
Los controles se organizan en su contenedor dependiendo de la disposición (layout) utilizado.

Diálogos

- JOptionPane.
  - Diálogos típicos (icono, mensaje, introducción de datos, botones)
- JFileChooser
  - Muestra un diálogo para abrir o guardar un fichero
- JColorChooser
  - Permite seleccionar un color.

Contenedores de alto nivel

- No necesitan de otro contenedor para mostrarse:
- - JFrame. Ventana con bordes, título, iconos para control
- - JWindow. Ventana sin bordes ni título
- - ~~JApplet. Ventana incrustable en un navegador.~~
- - JDialog.



Propiedades : Look And Feel

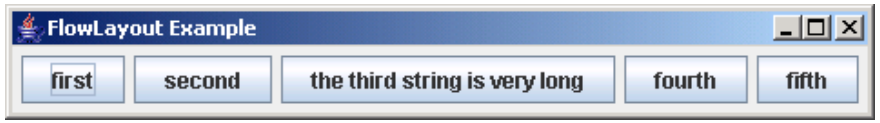
- El aspecto final de los controles depende del gestor de interfaz de usuario, que utiliza un objeto Look AndFeel.
- Se encarga de la apariencia de las ventanas y controles. Existen tres básicos, y se pueden crear más a partir de 1.5.
- - Metal, apariencia JAVA
- - Motif : UNIX
- - Windows: (propio del sistema)
- Para establecer la apariencia se usa :
- `UIManager.setLookAndFeel (nombreLookAndFeel)`
- `SwingUtilities.updateComponentTreeUI(frame)`

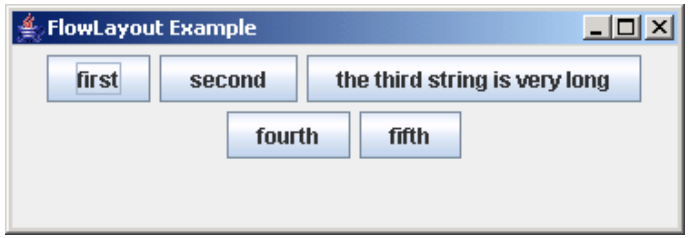
Elementos (algunos)

- JTextArea
  - Área de texto de más de una línea
  - Se pueden definir sus dimensiones, si es editable, o no,...
  - Tiene métodos cut, copy y paste
- JProgressBar, jSlider
  - Podemos saber su valor mediante getValue y especificar un valor con putValue
- JRadioButton.
  - Podemos ver si está seleccionado mediante el método isSelected().

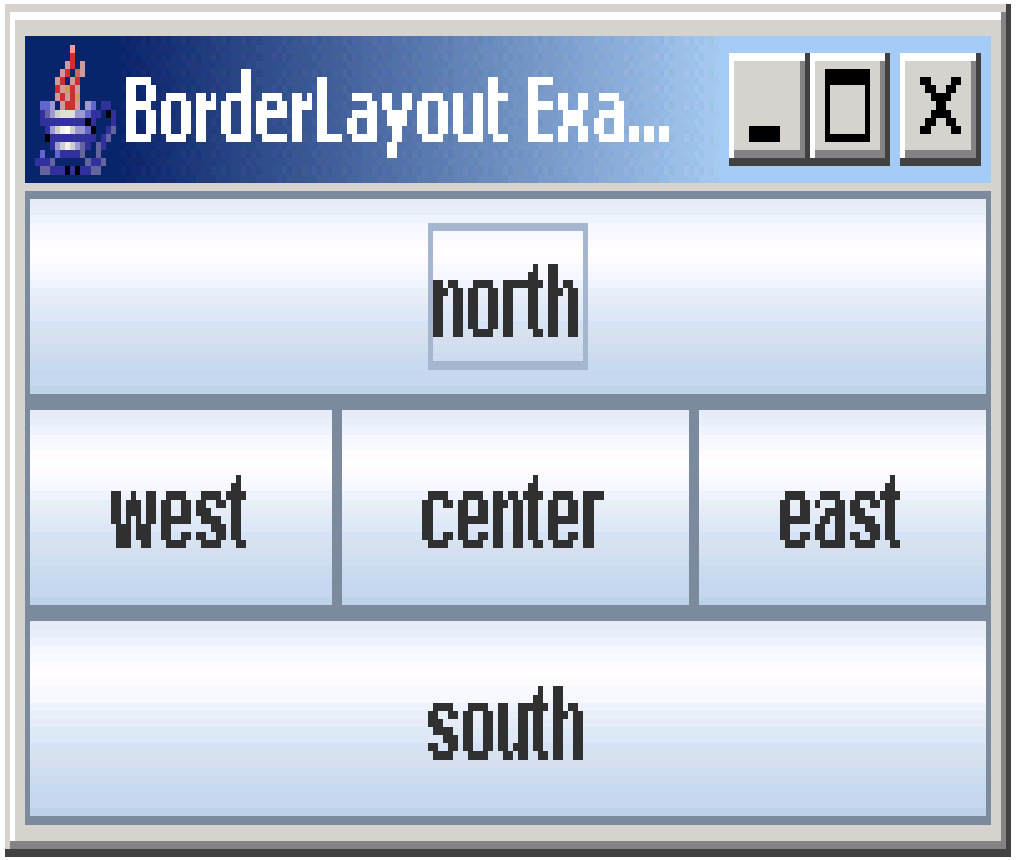
- GUI: Graphical User Interface
- Componentes de un GUI
- Disposiciones/Layouts
- Manejo de Eventos

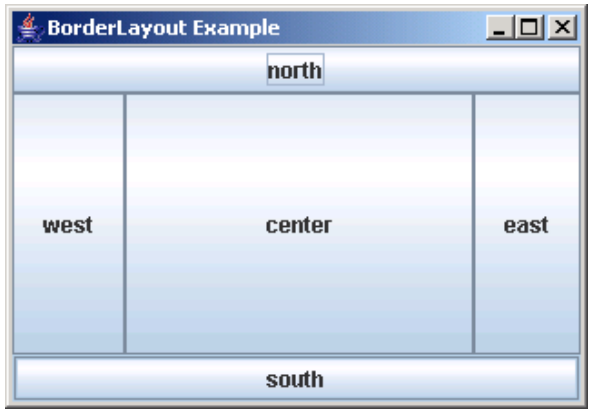
Flow Layout



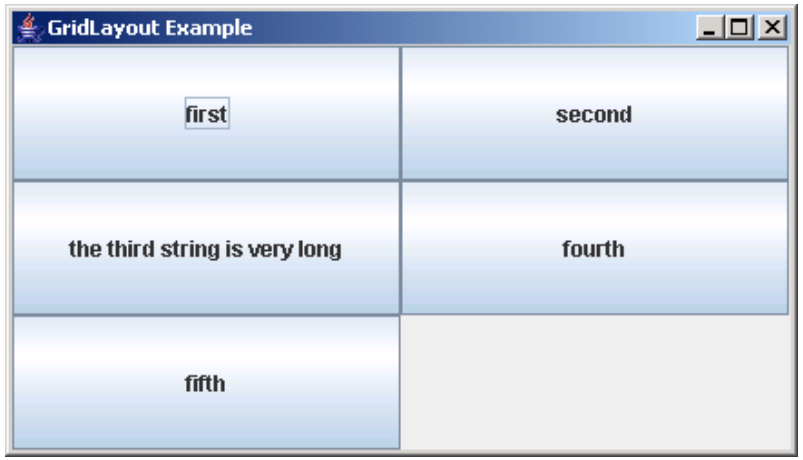


BorderLayout

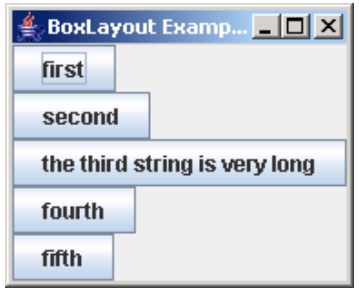


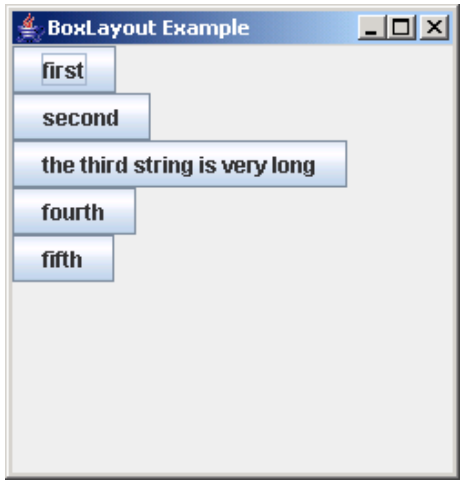


GridLayout



Box Layout





Layouts

- Disposicion de los elementos dentro del contenedor.
  - **FlowLayout** (de izquierda a derecha)
  - **BorderLayout** (bordes de la ventana, ocupando espacio libre). Alineación de control en 5 zonas diferenciadas (NORTH, SOUTH, WEST, EAST, CENTER).
  - **Null Layout**. Posiciones absolutas
  - **CardLayout.** Los controles se muestran uno cada vez. Con los métodos first(contenedor), previous(contenedor), next(contenedor), last (contenedor) se puede navegar entre las páginas el contendor. Con show (container, nombre) se puede mostrar la página nombre (la propiedad nombre se define como un atributo de cada control en este layout. En netbeans aparece como "Nombre de la tarjeta").
  - **GridLayout.** Agrupa los controles en forma de tabla (filas, columnas)
  - **GridBag Layout.** Agrupa los controles en forma de celdas con gran control sobre disposición en la tabla.
  - **BoxLayout.** Elementos a lo largo de un eje, puede ser vertical u horizontal.
  - **SpringLayout.** Forma una malla de componentes.
- Se pueden anidar contenedores con layouts distintos

- GUI: Graphical User Interface
- Componentes de un GUI
- Disposiciones/Layouts
- Manejo de Eventos

Modelo de Eventos

- Un evento que genera un usuario cuando se está ejecutando una aplicación (pulsar una tecla, mover el ratón,...) es recogido por el sistema operativo. El sistema operativo notifica a las aplicaciones que están ocurriendo estos eventos y ellas deciden si han de responder o no de algún modo a estos eventos.
- Para cada tipo de evento existe
  - El componente, u objeto que creamos y sobre cuyo área mostrada actúa el usuario
  - El evento, u objeto que Java crea automáticamente cuando el usuario interactúa con un componente
  - Un "listener", u objeto que creamos y sobre el que delegamos la tarea de estar atento a ciertos eventos,
  - El listener, a su vez, tiene un manejador (handler) que hace la acción apropiada para responder a estos eventos.

2 Modelos

- Centralizado
  - Cada componente permite agregar un ActionListener que está relacionado con el tipo del componente
  - Pero ejemplo si a un botón se le agrega un ActionListener el mismo es invocado cuando se oprime el botón ya sea con el mouse o con el teclado.
- Gestión de manejadores de eventos específicos

Parte II. Excepciones

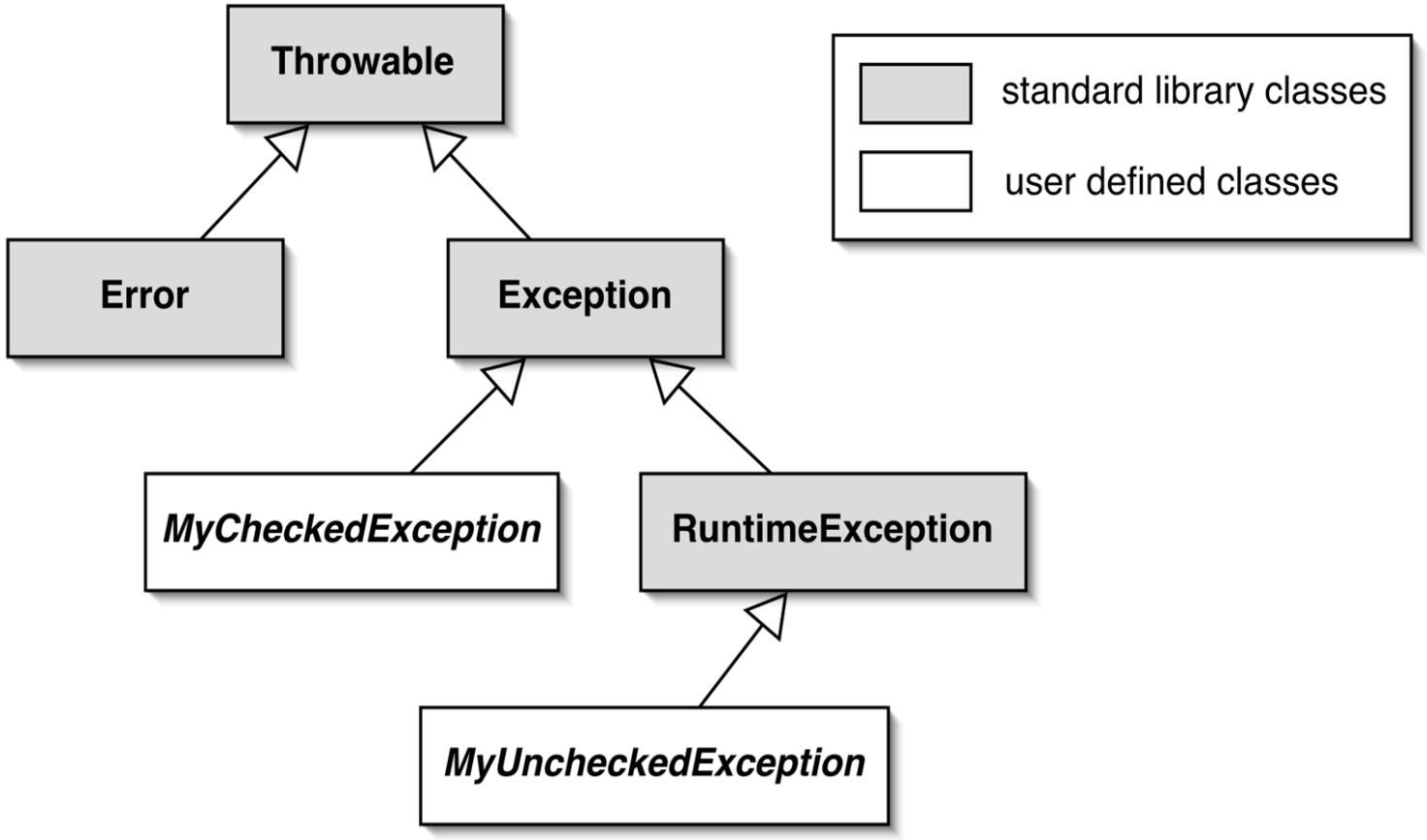
Ejemplo: AA_ExcepcionDivisionCero

Parte II. Excepciones

Si durante la ejecución de un programa se produce una situación anormal (un error) que interrumpe el flujo normal de ejecución, el método que se está ejecutando genera un objeto de la clase Throwable (“arrojable”) que contiene la información del error, de su causa y del contexto del programa en el momento en que se produce y lo entrega / arroja al programa en ejecución. Este evento

- Se puede capturar
- Si no se captura, el programa se interrumpe y el sistema de tiempo de ejecución muestra unos mensajes que describen el error

La jerarquía de la clase exception



- Error. Errores graves para los que es preferible que el programa se interrumpa

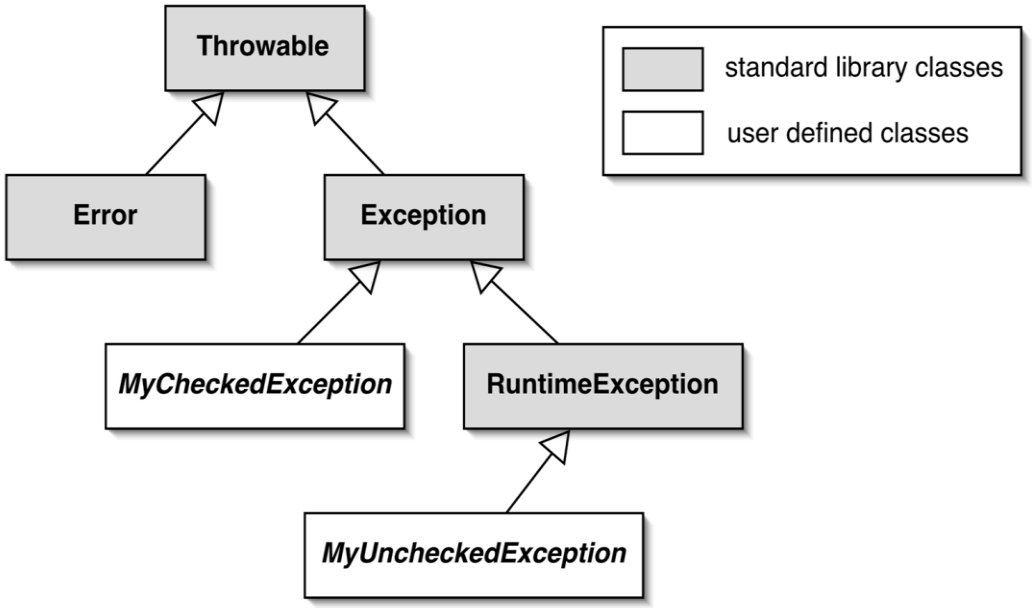
  Ejemplo: AB_Error

- Runtime exceptions. Código equivocado

  Ejemplo: AC_RunTime

Tipos de excepción

- **Excepciones Comprobadas**
  - Subclase de Exception
  - Uso para fallos anticipables
  - Cuando el programa se puede recuperar
- **Excepciones no Comprobadas**
  - Subclase de RuntimeException
  - Uso para fallos que no se pueden anticipar
  - Cuando la recuperación es improbable



Excepciones no comprobadas

- El uso de estas excepciones no es comprobado por el compilador
- Pueden producir la terminación del programa si no son capturadas (es el método normal)
- IllegalArgumentException es un ejemplo típico de excepciones de este tipo

Manejo de excepciones

- Se supone que se realizará la captura de las excepciones comprobadas
- El compilador asegura que se uso se va a controlar de forma correcta.
- Si se usan de forma correcta, los fallos que se han producido se pueden recuperar.

Efectos de una excepción

- El método que lanza la excepción termina de forma prematura
- No se devuelve ningún valor de retorno
- El control no vuelve al punto de la llamada de quien ha invocado el método que ha producido la excepción
- Quien ha invocado el método que ha producido la excepción (el cliente) debe "capturar" la excepción.

Lanzamiento de una excepción

- Se construye un objeto de tipo excepción
  - `new TipoDeExcepcion("...")`
- El objeto excepción se lanza
  - `throw...`
- Documentación Javadoc
  - `@throws TipoDeExcepcion razón`

```
If (clave == null) {
      throw new NullPointerException("clave nula ");
```

Sentencias

- Los métodos que lanzan una excepción comprobada deben incluir una sentencia throw.
- Los clientes que capturen una excepción deben proteger la invocación con una sentencia try
-
```
try {
    Proteger una o más sentencias en este bloque
}
catch (Exception e) {
    Informar y recuperarse de una excepción aquí
}
```

Ejemplo: AD_Excepciones

La sentencia try

1. La excepción se lanza desde aquí

```
try {
     agenda.salvarEnFichero(nombreDeFichero);
     IntentarDeNuevo = false;
}
catch(IOException e) {
     System.out.println("Incapaz de salvar " + filename);
     intentardeNuevo = true;
}
```

2. El control se transfiere aquí

Capturando múltiples excepciones

```
try {
    ...
    Proceso de referencia();
    ...
}
catch(EOFException e) {
    // Acciones ante una excepción de fin de fichero
    ...
}
catch(FileNotFoundException e) {
    // Acciones ante una excepción de fichero no encontrado
    ...
}
```

Ejemplo: AF_Excepciones

La sentencia finally

```
try {
    Protegemos una o varias sentencias en este bloque
}
catch(Exception e) {
        Informamos e intentamos recuperarnos de la excepción aquí
}
finally {
        Llevar a cabo aquí cualquier acción común a
   que se lance la excepción o no
}
```

La sentencia finally

- La sentencia finally se ejecuta incluso si la sentencia return se ejecuta en las sentencias try o catch
- Una excepción no capturada o propagada todavía existe via la sentencia finally

Comprobadas vs No comprobadas

AG_LanzaExcepcion

AH_LanzaExcepcionComprobada

Definiendo nuevas excepciones

- Extendemos RuntimeException para una excepción no comprobada o Exception para una excepción comprobada
- Definimos nuevos tipos para dar una mejor información de diagnóstico, incluyendo información de recuperación o de descripción de fallo.

Ejemplos02

Definiendo nuevas excepciones

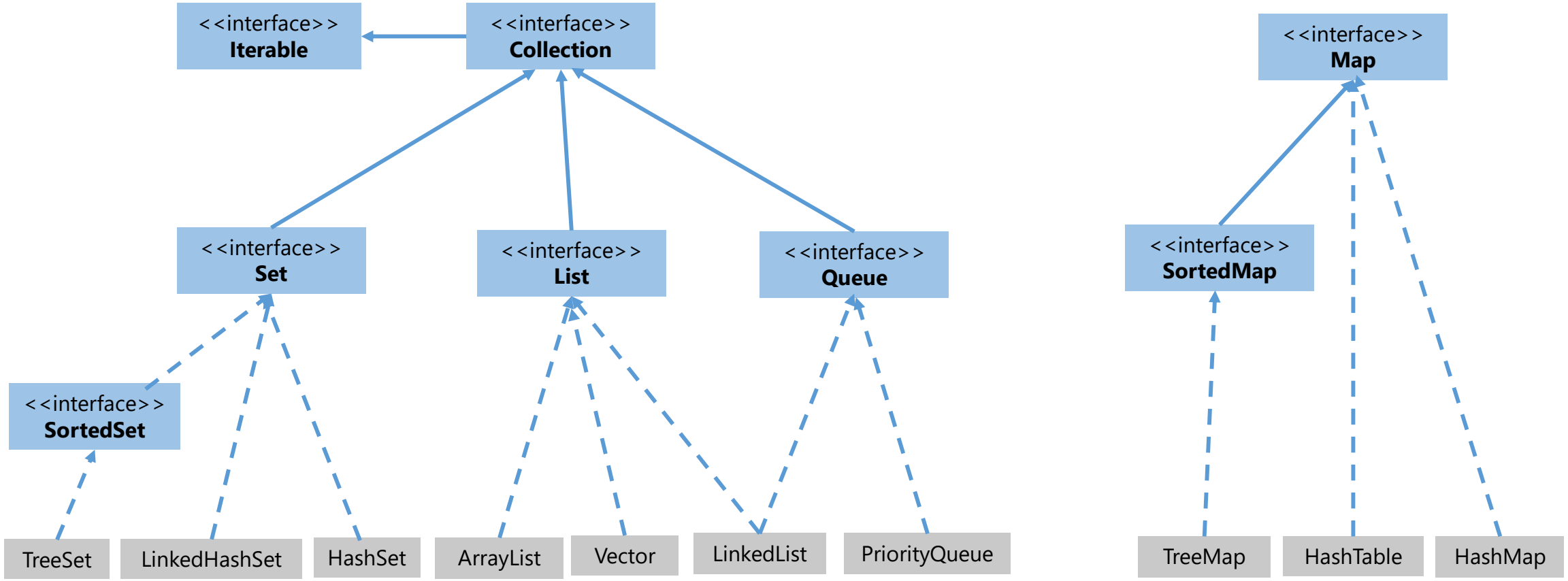
```
public class noHayInfoAsociada extends Exception
{
    private String clave;

    public noHayInfoAsociada(String clave)
    {
        this.clave = clave;
    }

    public String getClave()
    {
        return clave;
    }

    public String toString()
    {
        return ("No hay información asociada a la clave" + key);
    }
}
```

Colecciones (continuación)



Interfaz set

- Trata los datos como un conjunto matemático
  - No hay repeticiones
  - No hay orden
- Se pueden usar los métodos de la clase colección (ver UD6)
- Implementaciones
  - HashSet. Eficiente, no garantiza orden en la inserción
  - TreeSet. Eficiente, garantiza orden de elementos. Usa un comparador
  - LinkedHashSet, inserta elementos al final (garantiza orden basado en inserción)

ContColecciones. TreeSet

Interfaz map

- Trata los datos como un conjunto matemático
  - No hay repeticiones
  - No hay orden
- Se pueden usar los métodos de la clase colección (ver UD6)
- Implementaciones
  - HashSet. Eficiente, no garantiza orden en la inserción
  - TreeSet. Eficiente, garantiza orden de elementos. Usa un comparador
  - LinkedHashSet, inserta elementos al final (garantiza orden basado en inserción)

Interfaz map

"Pedro Martinez"	"686 33 44 55"
“María Sanchez"	"699 34 54 64"
“José Perez"	"679 23 24 25"

```
HashMap <String, String> guiaTelefonos = new HashMap<String, String>();

guiaTelefonos.put("Paco Martinez", "686 33 44 55");
guiaTelefonos.put(“María Sanchez", "699 34 54 64");
guiaTelefonos.put(“José Perez", "679 23 24 25");

String numeroDeTelefono = guiaTelefonos.get(“María Sanchez");
System.out.println(numeroDeTelefono);
```

ContColecciones. HashMap

Control de cambios

Fecha	Descripción
21/3/2023	Inicial

msierra@iesmordefuentes.com

Enlaces interesantes:

https://www.w3schools.com

https://docs.oracle.com/javase/tutorial/

https://www.tutorialcup.com/es/Java/conjunto-de-%C3%A1rboles-en-java.htm