



PRÁCTICA POO

Curso 2015/2016

Tutor: Roberto Centeno
(rcenteno@lsi.uned.es)

Tutoría práctica POO

1. Protocolo de entrega
2. Normas del enunciado
3. Práctica: Funcionalidades
4. Diseño
5. Diseño interfaz (textual vs gráfica)
6. Detalles de implementación
7. Dudas y estado de prácticas

1. Protocolo de entrega (I)

- Cuándo?
 - Fecha límite entrega junio: **viernes 13 de mayo**
 - Fecha límite entrega septiembre: **viernes 8 de julio**
- Dónde?
 - **Curso virtual (aLF)**: equipo docente
 - **Correo electrónico**: tutor (rcenteno@lsi.uned.es)
- Cómo?
 - **Fichero rar**: <DNI>.zip (ej. 12345678.zip **CONTRASEÑA 1234**). Directorio: 12345678/.../...
 - **Adjunto a correo electrónico**: rcenteno@lsi.uned.es
 - **Asunto**: PRACTICA POO 2016 <dni>.
 - **Cuerpo**: nombre, apellidos, **GRUPO Tutoría**, etc..
- Qué?
 - **Fichero LEEME.txt** (instrucciones ejecución práctica, clase main)
 - **Datos de prueba** (materiales, usuarios, préstamos, solicitudes préstamos interbibliotecarios,...) para probar el funcionamiento del SIGB
 - **Ficheros java y class** (compilados y funcionando correctamente)
 - **Memoria**: pdf (estructura del enunciado)

1. Protocolo de entrega (II)

- Calificaciones?
 - **Tutor** (Roberto Centeno)
 - **eMail**: misma dirección de entrega (ATENTOS CORREO)
- Revisiones?
 - **Tutor**: (Roberto Centeno)
 - **Límite**: 2 días después de recibir calificación (ATENTOS CORREO)
 - **eMail**: (rcenteno@lsi.uned.es)
 - **Asunto**: REVISION PRACTICA POO 2016 <dni>
 - **Cuerpo**: nombre, apellidos, **GRUPO TUTORÍA** y **MOTIVACIÓN**
revisión

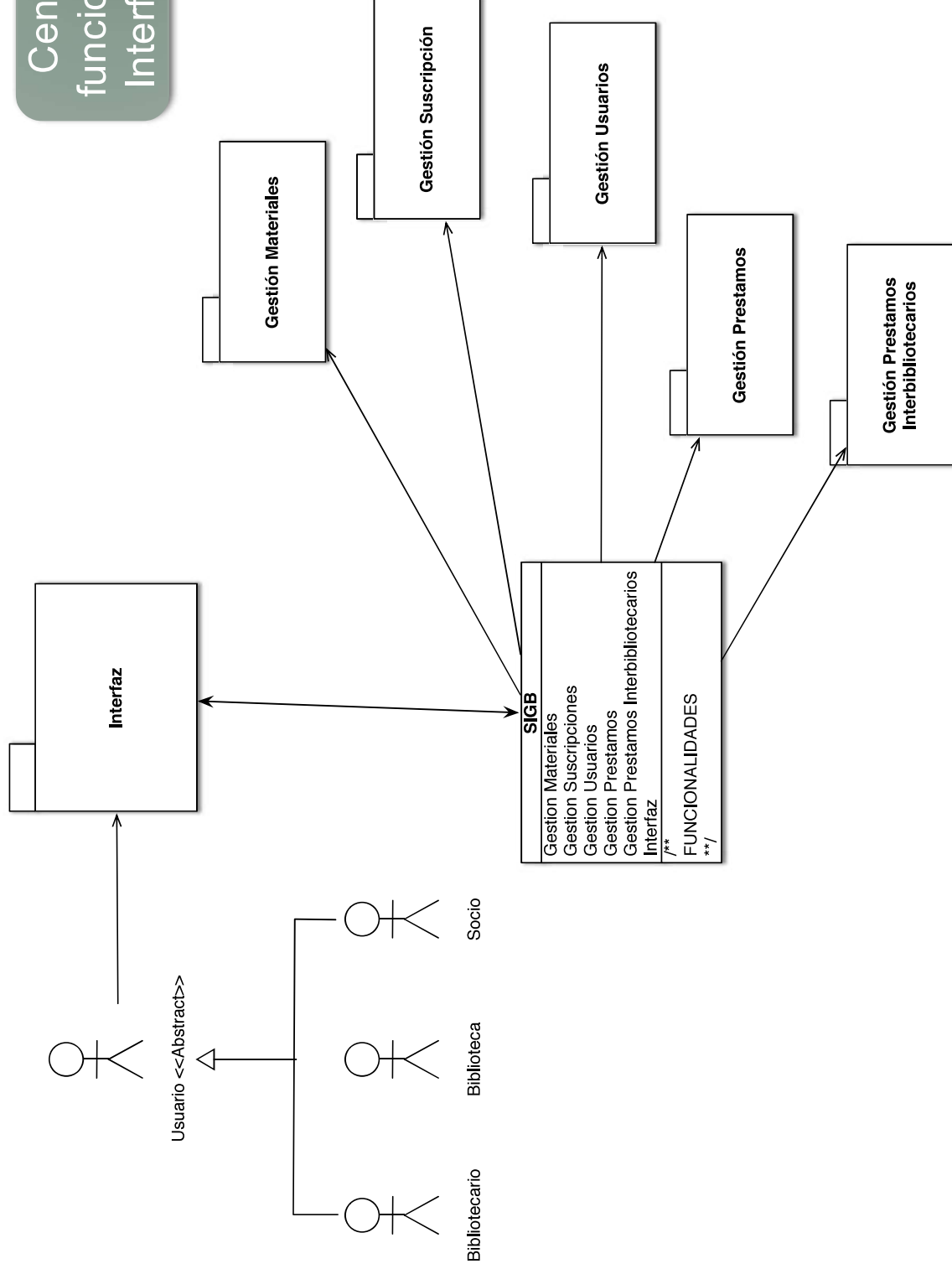
2. Normas del enunciado

- Estructura por niveles
 - Aprobar la práctica (5.0) => Superar **CORRECTAMENTE** niveles: 1, 2 y 3
 - Optar calificación superior implica superar nivel inferior **COMPLETO**
 - Implementar niveles en el orden indicado, no implementar niveles no consecutivos >>> No implementar GUI si no están TODAS las funcionalidades
 - Nivel 1: Transversal, se debe completar iterativamente (memoria, clases, relaciones, ficheros java, diagramas,...)

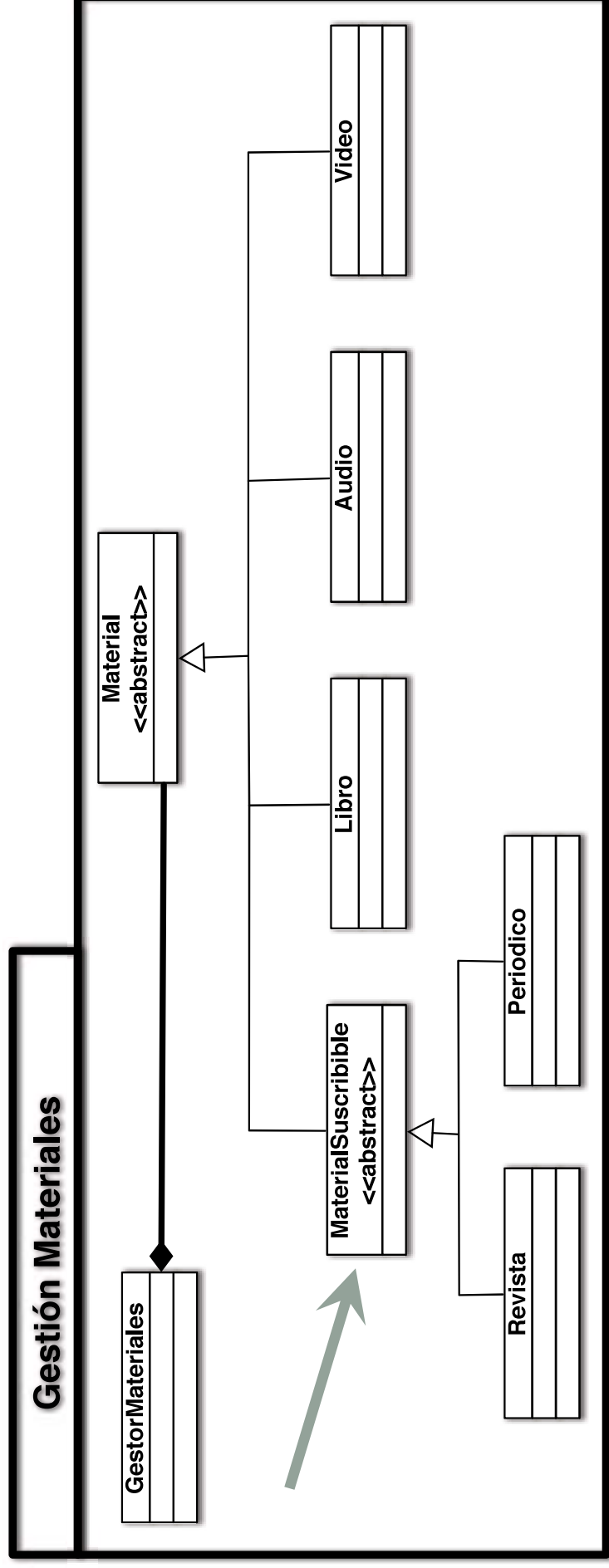
3. Funcionalidades SIGB

- **Gestión Materiales** (Libro, Revista, Periódico, Audio, Video, ...)
 - Añadir/Borrar/BuscarSencilla (1 campo) Materiales
 - BúsquedaFlexible (2 o más campos) / Listado Prestados
- **Gestión Suscripciones** (Temporalidad y Material: Revista o Periódico)
 - Añadir/Borrar/Buscar Suscripciones
- **Gestión Usuarios** (Socio/Bibliotecario)
 - Alta/Baja/GeneracionTarjetas/ControlAcceso/Buscar Usuarios (prestamos)
- **Gestión Prestamos**
 - Crear(fechaDevolución,NºMáximoMateriales)/AvisosCaducado/Multa(No prestamo en n dias)/Listados Prestamos
 - Crear/Borrar/Listar Reservas (Mensaje Usuario Prestamo)
 - Devolución Prestamo (Mensaje a usuario reserva)
- **Gestión Prestamos Interbibliotecarios**
 - Gestión Bibliotecas (Materiales) (Biblioteca un tipo de Usuario)
 - Búsqueda flexible
 - Crear Préstamo (Solicitud)
 - Generar Solicitudes (Exportar solicitudes a archivo texto)
 - Procesar Solicitudes (Importar fichero de solicitudes y crear préstamos)
- **Interfaz**
 - Texto / GUI

4. Diseño: SIGB (I)

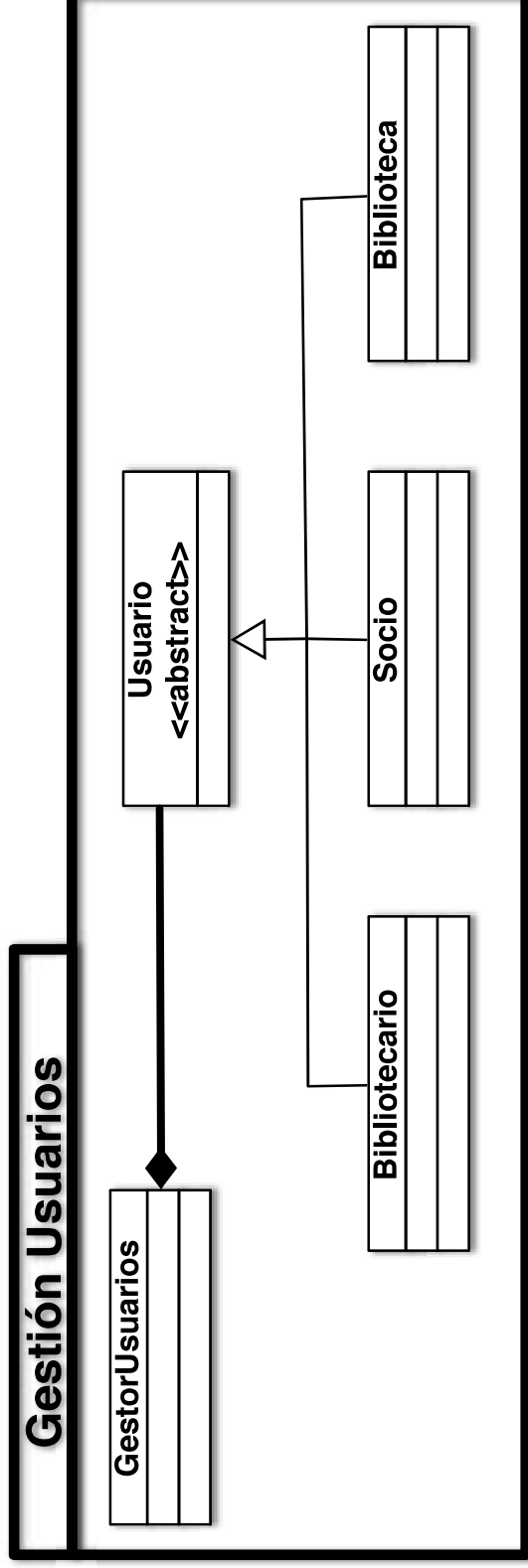


4. Diseño: Gestión Materiales (II)



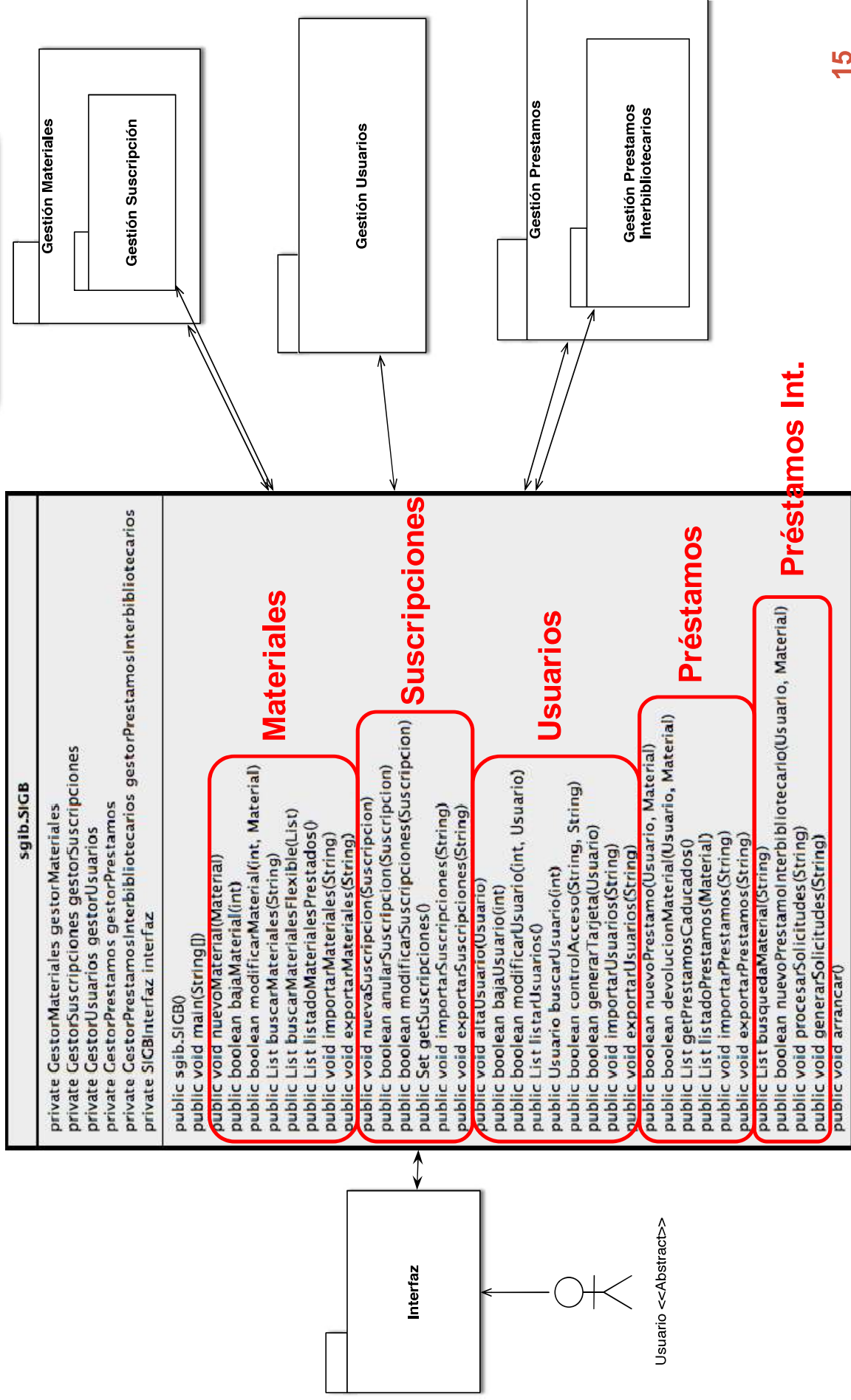
4. Diseño: Gestión Usuarios (II)

Bibliotecas como otro tipo de usuario (Prestamos Interbibliotecarios)



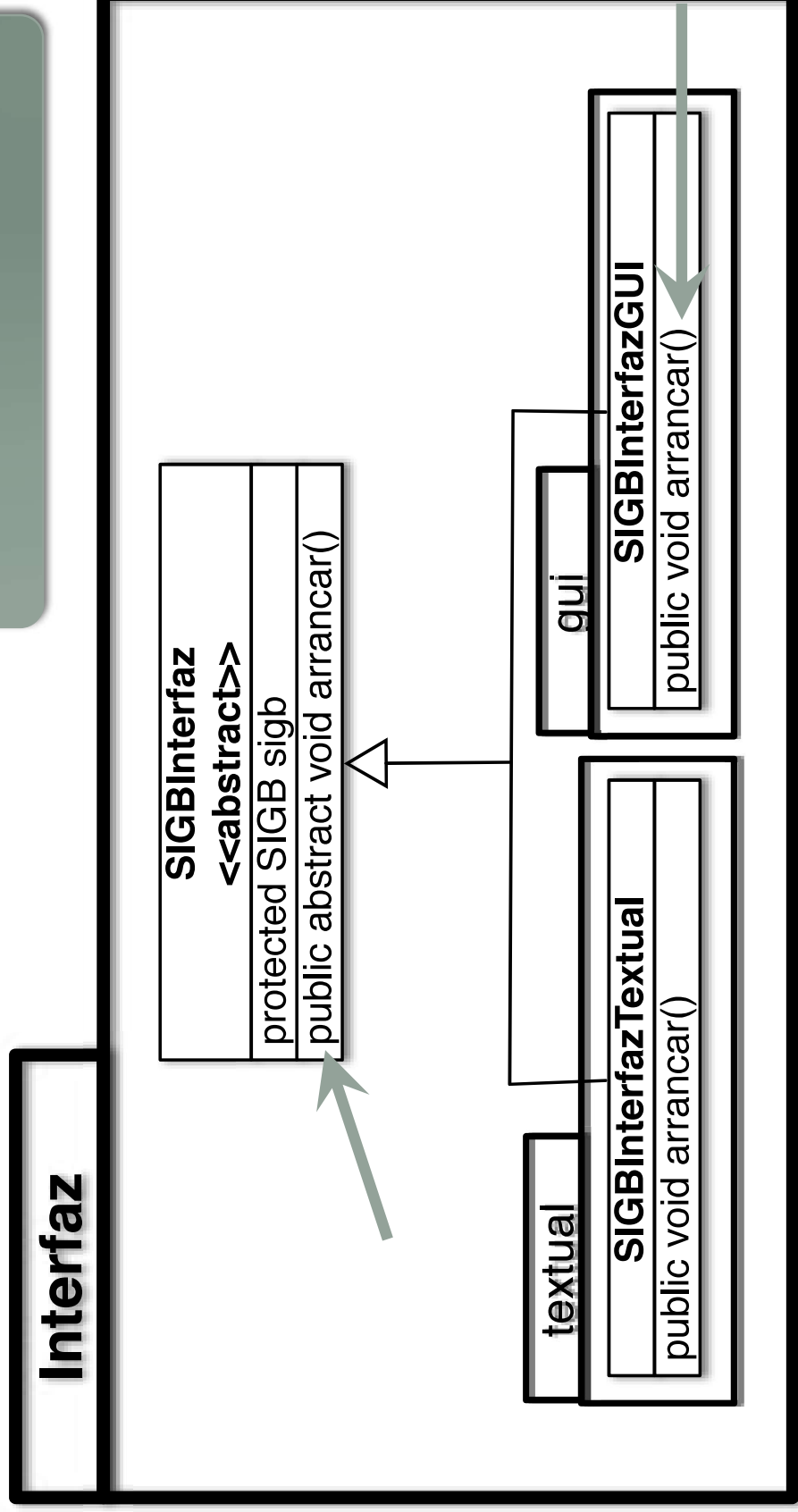
4. Diseño: SIGB

Re-
empaquetado



5. Diseño interfaz

Clase abstracta y herencia



5. Diseño interfaz (textual)

Introduzca Usuario: pepe
Introduzca Contraseña: *****

1. Materiales
2. Usuarios
3. Préstamos

Elija la opción que desee: 1

1. Nuevo material
2. Baja material
3. Buscar materiales
4. Modificar un material

En función del tipo de
usuario se mostrará el
menú de acciones

Pedir campos:

...


crear material (new Material(...))

Interfaz => sigb.nuevoMaterial(m);

SIGB => gestorMateriales.nuevoMaterial(m)

GestorMateriales => añadir el material e informar

5. Diseño interfaz (gráfica)

1. Librería Swing de java
2. Ejemplos de uso en <http://docs.oracle.com/javase/tutorial/uiswing/>
3. Pequeño tutorial de swing 

6. Detalles de implementación (I)

- Cómo leer del teclado y escribir por pantalla (interfaz textual)
- Cómo importar/exportar objetos a fichero
- Polimorfismo con `toString()`

6. Detalles de implementación (II)

- Cómo leer del teclado y escribir (interfaz textual)

```
@Override
public void arrancar() {
    this.menuPrincipal();
}

private void menuPrincipal() {
    Scanner in = new Scanner(System.in);
    System.out.println("0. Salir");
    System.out.println("1. Materiales");
    System.out.println("2. Usuarios");
    System.out.println("3. Prestamos");
    boolean salir = false;
    int menuItem;
    do {
        System.out.print("Elige la opción deseada: ");
        menuItem = in.nextInt();
        switch (menuItem) {
            case 1:
                System.out.println("Has elegido la opción #1");
                // do something...
                break;
            case 2:
                System.out.println("Has elegido la opción #2");
                // do something...
                break;
            case 3:
                System.out.println("Has elegido la opción #3");
                // do something...
                break;
            case 0:
                salir = true;
                break;
            default:
                System.out.println("Opción incorrecta.");
        }
    } while (!salir);
    System.out.println("Bye-bye!");
}
```

```
import java.util.Scanner;
```

6. Detalles de implementación (III)

- Importar/exportar objetos a fichero
 1. Implementar interfaz serializable en todas las clases que se quieren exportar/importar: Materiales, Usuarios, Préstamos, ...
 2. Re-escribir su método toString() para que genere el formato deseado.

```
import java.io.Serializable;

/* Write a description of class ProductoStock here. */
public abstract class Material implements Serializable {

    private int codigo;
    private String titulo;

    /*
     * (non-Javadoc)
     * @see java.lang.Object#toString()
     */
    @Override
    public String toString() {
        return "ProductoStock [codigo=" + this.codigo + ", titulo=" + this.titulo + "];"
    }
}
```


6. Detalles de implementación (IV)

- Exportar objetos a fichero
 1. Crear fichero con clase File
 2. Se lo pasamos a FileOutputStream para escribir
 3. Este se lo pasamos a ObjectOutputStream para que escriba
 4. Escribimos los objetos en el fichero
 5. Cerramos ObjectOutputStream

```
public class GestorPrestamosInterbibliotecarios {  
  
    public void generarSolicitudes(String rutaFichero) {  
        File fichero = new File(rutaFichero);  
        try {  
            FileOutputStream fos = new FileOutputStream(fichero);  
            ObjectOutputStream oos = new ObjectOutputStream(fos);  
            for (PrestamoInterbibliotecario p : this.prestamosInterbibliotecarios) {  
                oos.writeObject(p);  
            }  
            oos.close();  
        }  
        catch (FileNotFoundException e) {  
            e.printStackTrace();  
        }  
        catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

6. Detalles de implementación (V)

- Importar objetos a fichero

1. Cargar el fichero a leer y se lo pasamos a FileInputStream y a ObjectInputStream para que lea
2. Creamos un bucle para leer los objetos hasta el final de fichero EOF

```
public void procesarSolicitudes(String rutaFichero) {  
    File Fichero = new File(rutaFichero);  
    boolean eof = false;  
    try {  
        FileInputStream fis = new FileInputStream(Fichero);  
        ObjectInputStream ois = new ObjectInputStream(fis);  
        PrestamoInterbibliotecario p = (PrestamoInterbibliotecario) ois.readObject();  
        while (!eof) {  
            this.gestorPrestamos.nuevoPrestamo(p.getUsuario(), p.getMaterial());  
            System.out.println("Leo 1");  
            try {  
                p = (PrestamoInterbibliotecario) ois.readObject();  
            } catch (EOFException e) {  
                eof = true;  
            }  
        }  
        ois.close();  
    } catch (EOFException e) {  
        System.out.println("Fichero está vacío");  
    }  
    catch (FileNotFoundException e) {  
        e.printStackTrace();  
    }  
    catch (IOException e) {  
        e.printStackTrace();  
    }  
    catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```

Capturamos la excepción que se genera al fin del fichero (EOFException)

7. Dudas y estado prácticas

