

## **Intro POO (VII)**

- Archivos
- Excepciones

# Recordando

- Recordando clase anterior...
  - El método **toString()**
  - Métodos **get...()** y **set...()**
  - Operadores ***abstract*** y ***final*** para clases

# Los Package - implementación

File Path: ~/Desktop/ejPackage/usoPaquetes/mios/utiles.java

```
1 // *****
2 * Ejemplo de la creación de un paquete personalizado
3 * clase con utilitarios que serán llamados después desde
4 * otros programas mediante un import.
5 * Este package debe ser guardado en un directorio con el
6 * mismo nombre del package.
7 */
8
9 // se define el nombre del paquete
10 package mios;
11
12 public class utiles {
13
14     public int largo(String s) {
15         return s.length();
16     }
17
18     public boolean vacia(String s) {
19         return (s.length() == 0);
20     }
21
22     public boolean esPar(int n) {
23         return (n%2 == 0);
24     }
25 }
26
27
```

File Path: ~/Desktop/ejPackage/usoPaquetes/pruebaPaquete.java

```
1 // hacer un import de nuestro propio paquete
2 import mios.*;
3
4 public class pruebaPaquete {
5
6     /**
7      * @param args
8      */
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         String st = new String();
12         // se crea un objeto de la clase utiles
13         // de nuestro paquete mios
14         utiles u = new utiles();
15         int x = 3;
16         st = "hola";
17         // se invocan a operaciones de la clase
18         // utiles del paquete creado.
19         System.out.println(u.largo(st));
20         System.out.println(u.vacia(st));
21         System.out.println(u.esPar(x));
22     }
23
24 }
```

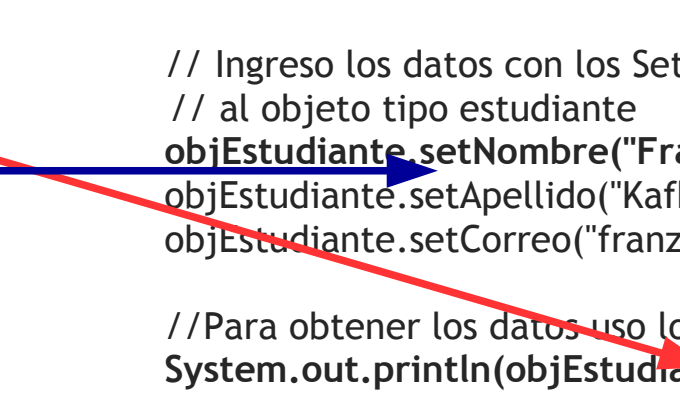
# Método toString

```
7 public class TestArreglo {
8     public static void main(String [] args) throws Exception {
9         InputStreamReader isr = new InputStreamReader(System.in);
10        BufferedReader br = new BufferedReader(isr);
11
12        System.out.println("\nTrabajo basico con Arreglos!!!\n");
13        System.out.println("Cuantos datos quieres mostrar? : ");
14
15        int largo = Integer.parseInt(br.readLine());
16
17        arreglo A = new arreglo(largo);
18        System.out.println("Largo total arreglo: "+A.vector.length);
19        A.mostrar(largo);
20
21        System.out.println(A);
22
23        // destruccion explicita del objeto
24        A = null;
25        if (A==null) System.out.println("Objeto destruido\n");
26    }
27 }
28
29 class arreglo {
30     int [] vector = new int[50];
31
32     public arreglo(int n) {
33         for (int i=0;i<n;i++)
34             this.vector[i] = i;
35     }
36
37     public String toString() {
38         String msj = "";
39         for (int i=0; i < vector.length ; i++ ) msj+= String.valueOf(vector[i]) + "-";
40         return msj;
41     }
```

# Métodos setter y getter

```
class Estudiante {  
    private String nombre;  
    private String apellido;  
    private String correo;  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public String getApellido() {  
        return apellido;  
    }  
  
    public void setApellido(String apellido) {  
        this.apellido = apellido;  
    }  
  
    public String getCorreo() {  
        return correo;  
    }  
  
    public void setCorreo(String correo) {  
        this.correo = correo;  
    }  
}
```

```
public class TestEstudiante {  
  
    public static void main(String args[]) {  
  
        Estudiante objEstudiante = new Estudiante();  
  
        // Ingreso los datos con los Setter  
        // al objeto tipo estudiante  
        objEstudiante.setNombre("Franz");  
        objEstudiante.setApellido("Kafka");  
        objEstudiante.setCorreo("franz.kafka@ufro.cl");  
  
        // Para obtener los datos uso los Getter  
        System.out.println(objEstudiante.getNombre());  
        System.out.println(objEstudiante.getApellido());  
        System.out.println(objEstudiante.getCorreo());  
    }  
}
```



# Uso de archivos

- Con Java es posible acceder a un archivo al igual que el resto de los demás LDP.
- Permite el acceso para archivos del tipo:
  - binario
  - secuencial
  - ***acceso aleatorio***

¿Qué es un archivo de tipo binario?  
¿Qué es un archivo de tipo secuencial?  
¿Qué es un archivo de tipo aleatorio?

# Uso de archivos

- Uso de archivos con acceso aleatorio
  - se abre el archivo (ruta, modo de apertura)
  - se procesa el contenido
  - se cierra el archivo
- ¿Cómo se implementa en Java?

# Archivos de acceso aleatorio - Java

- Para abrir el archivo

- Declarar una variable para manipular el archivo:

- ```
RandomAccessFile arch = new RandomAccessFile("archivo.txt","rw");
```

- Para leer su contenido línea a línea:

- ```
while ( ( cadena = arch.readLine( ) ) !=null ) {
```

- ```
    System.out.println( cadena ); }
```

- Para escribir contenido en el archivo:

- ```
arch.writeBytes( cadenaEntrada );
```

- Para cerrar el archivo:

- ```
arch.close();
```



# Manejo de Excepciones

- En Java, una Exception es un cierto tipo de error que se ha producido en ***runtime***.
- Un ***buen*** programa debe gestionar correctamente los errores que se pueden producir.
- Las secciones a monitorear en busca de excepciones, están contenidas en un bloque:  
***try { // código que queremos supervisar }***
- Si ocurre una excepción dentro de un try, ésta es lanzada y para capturarla se usa el bloque:  
***catch (Excepcion) { // código para gestionarla }***

# Manejo de Excepciones

De la figura se tiene que:

- para un mismo **try** se pueden capturar ***n*** excepciones, usando ***n*** **catch**.
- al cumplirse alguno de los **catch**, se ejecuta el código asociado a él y el resto de ellos es omitido.
- sino es lanzada ninguna excepción dentro del bloque **try**, el programa ignora todos los **catch** y sigue su funcionamiento.

```
try {  
    // bloque a controlar en busca de excepciones  
}  
  
catch (tipoException-1 exc) {  
    // código que permite manipular la excepción }  
  
.....  
  
catch (tipoException-N exc) {  
    // código que permite manipular la excepción N }
```

# Manejo de Excepciones

- Existen diversos tipos de excepciones que pueden ser capturadas, entre ellas:
  - ***IOException***: clase general usada para capturar errores en operaciones de E/S. Se encuentra dentro del ***package java.io***
    - ***FileNotFoundException***: clase que permite manipular el hecho de que se trató de abrir un archivo en una ruta no válida.
  - ***AWTException***: clase que indica un error de Abstract Window Toolkit ha sido generado. Se encuentra dentro del ***package java.awt***
- Más detalles en:

<http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Exception.html>

Ver archivo Excepciones.java

# Caso personasUFRO

- A partir del caso iniciado la clase anterior, se le pide:
  - Agregar el uso de archivos para almacenar los datos de cada alumno, profesor y curso
  - Agregar el manejo de excepciones según corresponda

# Resumiendo

- Acceso de archivos con Java
  - Procesando archivos de acceso aleatorio
  - Ejemplo de uso de archivos
- Manejo de Excepciones en Java
  - Bloques try-catch
  - Ejemplo de uso de excepciones