8.- Validación de información.

8.1.- Necesidad de la revisión y validación de los datos.

Cuando se lee información tecleada por un usuario humano (hay usuarios que no son humanos, que son otros programas también llamados robots) existe la posibilidad de cometer errores al teclear la información, eso puede ocasionar que los programas no funcionen, no porque estén mal escritos, sino por la calidad de la información que procesan, por ejemplo una variable que espera un número entero puede recibir una letra lo cual ocasionaría un error en el programa receptor. Hay un dicho en el argot de los sistemas de información que dice "basura entra basura sale". Al proceso de revisar la información se le llama validación. Es un proceso rutinario, tedioso y repetitivo que vale la pena estandarizar y automatizar.

8.2.- Tipos de validación.

Hay varios tipos de validaciones:

1. Validación por configuración. En este tipo de validación se revisa que si se espera un número, se teclee algo que puede ser convertido a número, si se espera una fecha y el dato recibido pueda ser convertido a una fecha correcta. Por ejemplo un nombre de persona no esperamos que contenga números o caracteres especiales, no es posible tener un Ricardo 56 como nombre de pila. En otras palabras esta validación revisa que sean datos que puedan convertirse y operarse según el tipo de variable que los va a almacenar.
2. Validación por congruencia. En este tipo de validación se revisa el contenido de la variable para comprobar que es un dato congruente, por ejemplo un peso de una persona se espera que no sea mayor a 1000 kilogramos, o bien su altura mayor a 5 metros. En esta validación se revisa que los datos se encuentren en rangos congruentes de valores o conjuntos de valores discretos congruentes por ejemplo si se solicita el sexo de una persona se espera que la respuesta sea masculino o femenino.
3. Validación por existencia. En este tipo de validación se revisa que el dato exista en un conjunto de valores que el mismo sistema captura y actualiza dinámicamente por ejemplo la especialidad de un médico se revisa contra un catalogo de especialidades que sería diferente según el contexto, si estamos en una clínica odontológica tendríamos unas especialidades muy distintas a las que se utilizarían en una clínica oftalmológica.

Los tres tipos de validaciones no son excluyentes, al contrario a un mismo dato puede ser necesario efectuarle los tres tipos.
En esta sección cubriremos las validaciones por configuración.

8.2.- Procedimientos comunes de validación.

Hay varias maneras dos maneras de llevar a cabo una validación, en tiempo real y en forma diferida.
Cuando se valida en tiempo real, al momento en que el usuario trata de teclear un campo inválido el programa que también se le llama GUI por sus siglas en inglés (Graphical User Interface) responde con un mensaje de error, de tal manera que no le permite cometer errores al capturar la información.
Cuando se valida en forma diferida el programa revisa todos los datos capturados como un conjunto y se encuentra un error lo reporta y evita el proceso de la información.
Cada uno de estos procedimientos tiene su aplicación según la situación en que se utilicen.

8.3.- Validación de tipos primitivos.

El paquete inovaProg proporciona validaciones para los tipos de datos primitivos más comunes.

Asumiendo que tenemos las siguientes variables declaradas al principio del programa:

```
boolean[] existeError = new boolean[]{false};
String[] mensaje  = new String[]{""};
float datoFlotanteValidado = 0;
int   datoEnteroValidado   = 0;
String datoCadenaValidado  = "";
Date   datoFechaValidado   = new Date();
String datoleido = "";
```

Y suponiendo que la variable de datoleido será utilizada para contener el dato tecleado por el usuario:

Tenemos que la validación de un dato flotante sería de la siguiente manera:

```
datoFlotanteValidado = Ip.validarDato(
                           datoFlotanteValidado,
                           datoleido,
                           mensaje,
                           existeError);
if(existeError[0])
    //Aquí se procesa el error
    {Ip.wescribeMensaje(mensaje[0]);}
```

La validación para un entero es muy similar a la anterior:

```
datoEnteroValidado = Ip.validarDato(
                           datoEnteroValidado,
```


```
                              datoleido,
                              mensaje,
                              existeError);
if(existeError[0])
     //Aquí se procesa el error
     {Ip.wescribeMensaje(mensaje[0]);}
```

Validar un dato que contiene una cadena de caracteres revisando que no esté vacía:

```
datoCadenaValidado = Ip.validarDato(
                              datoCadenaValidado,
                              datoleido,
                              mensaje,
                              existeError);
if(existeError[0])
     //Aquí se procesa el error
     {Ip.wescribeMensaje(mensaje[0]);}
```

Note como todas las validaciones son casi iguales, solo varía la variable con el dato validado lo que hace extremadamente sencillo el utilizar estas rutinas.

El programa 801OperacionesDeValidacionDeDatos.java ilustra con más detalles estos procesos.

8.4.- Validación de fechas.

La validación de una fecha comprende la revisión de los números de días en los meses y validar por años bisiesto. Es indispensable cubrir estas revisiones de otra forma las validaciones no son completas.

La validación queda como se muestra a continuación:

```
datoFechaValidado = Ip.validarDato(
                              datoFechaValidado,
                              datoleido,
                              mensaje,
                              existeError);
if(existeError[0])
     //Aquí se procesa el error
     {Ip.wescribeMensaje(mensaje[0]);}
```

Como se puede observar esta validación conserva la estructura de la validación de los tipos primitivos.

8.5.- Actividades
Compile y ejecute el ejemplo adjunto agregando variables y validándolas.

8.6.- Ejercicio propuesto.
Escriba un programa que solicite una lista de artículos y que para cada artículo lea su código, nombre, existencia y precio, valide la información antes de guardarla en el archivo, escriba otro programa que lea y muestre el archivo capturado en el primer programa.