

UNIVERSIDAD DE GUADALAJARA

**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E
INGENIERÍAS**

INGENIERÍA EN COMPUTACIÓN(INCO)



MATERIA: Computación Tolerante a Fallas.

SECCIÓN: D06.

DOCENTE: Lopez Franco, Michel Emanuel.

ALUMNO: Izmael Guzmán Murguía.

CÓDIGO: 216728179.

Tarea: (Par. 2) Otras herramientas para el manejo de errores.

FECHA DE ENTREGA: Miércoles 9 de febrero de 2022.

Índice

Introducción	3
Validación de enteros	4
Pruebas	5
Entrada de string	5
Entrada combinada	5
Entrada con punto	6
Entrada con espacio y número	6
Entrada con espacio	7
Entrada vacía	7
Entrada válida int	8
Validación de flotantes	9
Pruebas	9
Entrada de string	10
Entrada combinada	10
Entrada con punto	11
Con más de un punto	11
Entrada con espacio y número	12
Entrada con espacio	12
Entrada vacía	13
Entrada válida float	13
Validación de ficheros con la estructura try-catch	14
Prueba	15
Validación para entrada de tipo de archivo especificado	15
Prueba	16
Validación de ficheros mediante throws	16
Prueba	17
Repositorio de github	17
Conclusión	18
Bibliografía	19

Introducción:

Para esta actividad se pretende conocer y aplicar técnicas de validación ante errores que pudieran acaecer en nuestros programas, en esta ocasión se ejemplifica mediante capturas de pantalla algoritmos de elaboración propia y la estructura try-catch que es muy utilizada en diferentes lenguajes de programación para validar ciertos tipos de errores, por decirlo de alguna manera aquellos que pueden causar grandes inconsistencias, además en el trabajo de ejemplo esta desarrollado en el lenguaje Java pero estos mismos algoritmos los podemos aplicar en lenguajes muy diferentes como python y de mas, la finalidad de realizar estos algoritmos es aprender técnicas que puedan llevar a crear un sistema robusto, un buen producto de software no solo es aquel que cumple con los requerimientos funcionales o lo que el cliente necesita o pide, un buen software debe ser seguro, robusto, que permita la escalabilidad y de más, robustez implica que nuestro software será capaz de seguir funcionando incluso cuando se produce un evento que no estaba planeado que sucediera podemos por ejemplo tener el caso en que en un software se solicite un dato entero y el usuario ingresa un caracter o cadena de caracteres por accidente, esto ocasiona naturalmente que el programa se detenga, o tenga un comportamiento aleatorio, es precisamente esto lo que debemos evitar, comportamientos de los cuales no queremos que haga el software, dicho de otra manera queremos que sea robusto.

Validación de enteros:

Para este tipo de validación se implementará un algoritmo que utiliza el código ascii en conjunto con el recorrido de una cadena evaluando el orden en que se ponen los números pero además se consideran los espacios, esto significa que si se ingresa un espacio al principio, al final o intermedios no se considerará como un número entero, para que sea considerado un número entero tiene que escribirse como tal, a continuación se muestra el algoritmo implementado con el código comentado para su mejor entendimiento:

```
private boolean v(int f,String value){//Funcion para validar datos enteros y
// flotantes, recibe 0 si es entero y
//1 si es flotante, como segundo parametro
//el string a validar para fialmente
//retornar un voleano.

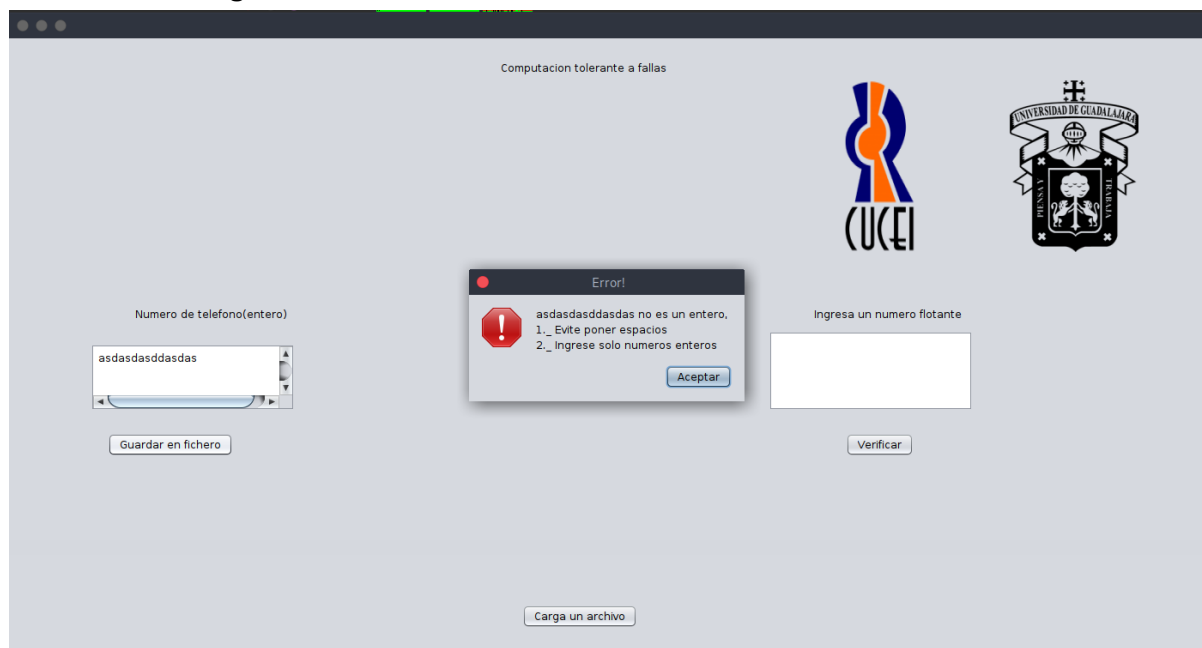
String B;//Cadena que se validara
int i,o;//Posicion en la cadena y vander
i=0;//iniciaizacion en la primera posicion
B=value;//Asignacion de el String al String evaluado
o=0;//Vandera se inicializa en 0 o como si cumple
int token=0;//Variable para conteo de puntos en caso de ser flotante
while(i<B.length()){//Bulce para recorrer la cadena
    if(f==0){//Si se pide validar un entero
        if(B.charAt(i)>47 && B.charAt(i)<58)i++;//Si es un numero
        else {//Si no es un numero
            o=1;//No es un entero
            break;//Terminamos con el recorrido de la cadena
        }//Fin de si no es un numero
    }//Fin de validacion de entero

    if(f==1){//Si se pide validar un flotante
        if(B.charAt(i)>47 && B.charAt(i)<58|B.charAt(i)=='.' ){//SI es
//un numero o punto
            if(token==1 && B.charAt(i)=='.' ){//Si ya se registro un punto
//y se detecta otro
                o=1;//No es un numero flotante
                break;//termina el recorrido
            }
            if(token==0 && B.charAt(i)=='.' ){//si es el primer punto
                token=1;//Se registra que ya se encontro un punto
                i++;//Incremento de posicion en la cadena
            }//Fin de if en validacion de float
        }else {//Si no es un numero ni un punto
            o=1;//No es un flotante si no un caracter distinto
            break;//Termina el recorrido de la cadena
        }//Fin de no flotante
    }//Fin de if para validacion flotante
} //Fin de recorrido de cadena
if(o==1)//Si no cumple con el tipo de dato que se pide
    return false;//Retorno no cumple con el tipo de dato
return true;//De lo contrario si cumple con el tipo de dato
}
```

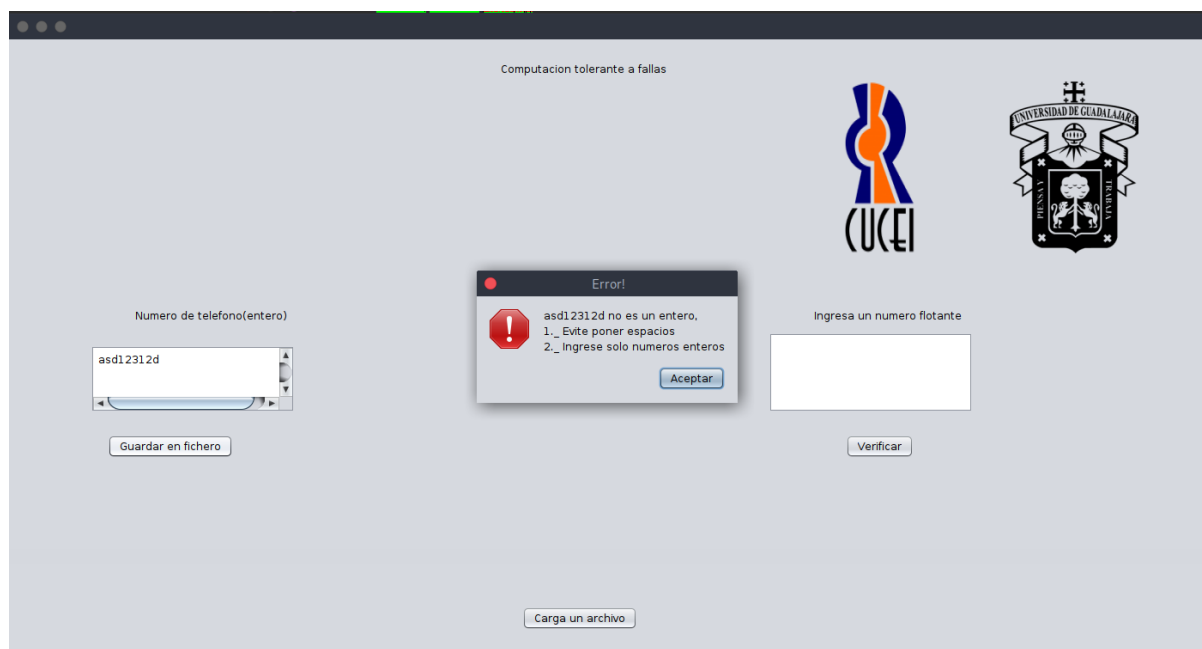
Pruebas:

En esta fase se pretende poner entradas diferentes para hacer que el sistema falle, se implementara como ejemplo entradas únicamente string, cadenas combinadas con números, entrada con punto, entrada con espacio y número, entrada con solo espacios, entrada vacía y finalmente una entrada válida.

Entrada de string:





Entrada combinada:



Entrad con punto:

Computacion tolerante a fallas

Numero de telefono(entero)

123.4

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

123.4 no es un entero,
1._ Evite poner espacios
2._ Ingrese solo numeros enteros

Aceptar

Entrada con espacio y número:

Computacion tolerante a fallas

Numero de telefono(entero)

123 2342 234

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

123 2342 234 no es un entero,
1._ Evite poner espacios
2._ Ingrese solo numeros enteros

Aceptar

Entrada con espacio:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

no es un entero,
1._ Evite poner espacios
2._ Ingrese solo numeros enteros

Aceptar

Entrada vacía:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

no es un entero,
1._ Evite poner espacios
2._ Ingrese solo numeros enteros

Aceptar

Entrada válida int:


Computacion tolerante a fallas



Numero de telefono(entero)

Guardar en fichero

Mensaje

 Objeto guardado correctamente con id = 321

Aceptar

Ingresa un numero flotante

Verificar

Carga un archivo

Validación de flotantes:

Este algoritmo utiliza la misma función que la validación de entero solo que recibe un parámetro diferente 1 cuando en int es 0, esta validación consiste en recorrer la cadena verificando mediante ascii que se ingresen números y en su caso un punto, cabe destacar que en caso de ingresar puntos solo sera uno, si se ingresa alguno mas se considerará como error y por supuesto también se consideran los espacios.

```
private boolean v(int f,String value){//Funcion para validar datos enteros y
// flotantes, recibe 0 si es entero y
//1 si es flotante, como segundo parametro
//el string a validar para fialmente
//retornar un voleano.

String B;//Cadena que se validara
int i,o;//Posicion en la cadena y vander
i=0;//iniciaizacion en la primera posicion
B=value;//Asignacion de el String al String evaluado
o=0;//Vandera se inicializa en 0 o como si cumple
int token=0;//Variable para conteo de puntos en caso de ser flotante
while(i<B.length()){//Bulce para recorrer la cadena
    if(f==0){//Si se pide validar un entero
        if(B.charAt(i)>47 && B.charAt(i)<58)i++;//Si es un numero
        else {//Si no es un numero
            o=1;//No es un entero
            break;//Terminamos con el recorrido de la cadena
        }//Fin de si no es un numero
    }//Fin de validacion de entero



    if(f==1){//Si se pide validar un flotante
        if(B.charAt(i)>47 && B.charAt(i)<58|B.charAt(i)=='.'){//SI es
//un numero o punto
            if(token==1 && B.charAt(i)=='.'){//Si ya se registro un punto
//y se detecta otro
                o=1;//No es un numero flotante
                break;//termina el recorrido
            }
            if(token==0 && B.charAt(i)=='.')//si es el primer punto
                token=1;//Se registra que ya se encontro un punto
            i++;//Incremento de posicion en la cadena
        }//Fin de if en validacion de float
        else {//Si no es un numero ni un punto
            o=1;//No es un flotante si no un caracter distinto
            break;//Termina el recorrido de la cadena
        }//Fin de no flotante
    }//Fin de if para validacion flotante
}//FIN de recorrido de cadena
if(o==1)//Si no cumple con el tipo de dato que se pide
    return false;//Retorno no cumple con el tipo de dato
return true;//De loc contrario si cumple con el tipo de dato
}
```

Pruebas:

Esta fase consiste en verificar que nuestro algoritmo funciona, para ello se hace una serie de pruebas con distintos tipos de entradas intentando que el programa falle, se intenta por ejemplo entrada de con string de letras y simbolos, entrada combinada de simbolos y numeros, entrada con punto, con más de un punto, entrada con espacios y numeros, entrada vacía y finalmente una entrada válida o float como se muestra a continuación.

Entrada de string:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

pruba no es un flotante,
1._ Evite poner espacios
2._ Ingrese solon umeros con formto entero o flotante

Aceptar

Entrada combinada:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

1234.5 asd no es un flotante,
1._ Evite poner espacios
2._ Ingrese solon umeros con formto entero o flotante

Aceptar

Entrada con punto:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

Carga un archivo



Error!

! no es un flotante,
1._ Evite poner espacios
2._ Ingresa solon umeros con formto entero o flotante

Aceptar

Con más de un punto:

Computacion tolerante a fallas

Numero de telefono(entero)

Guardar en fichero

Ingresa un numero flotante

Verificar

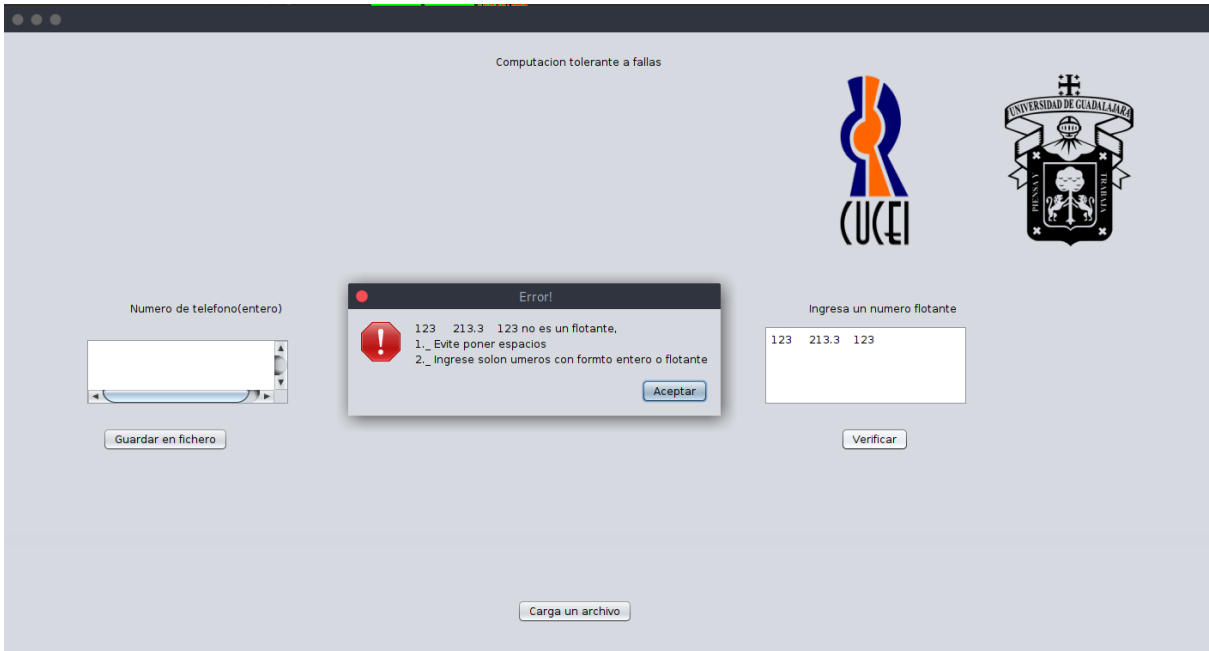
Carga un archivo

Error!

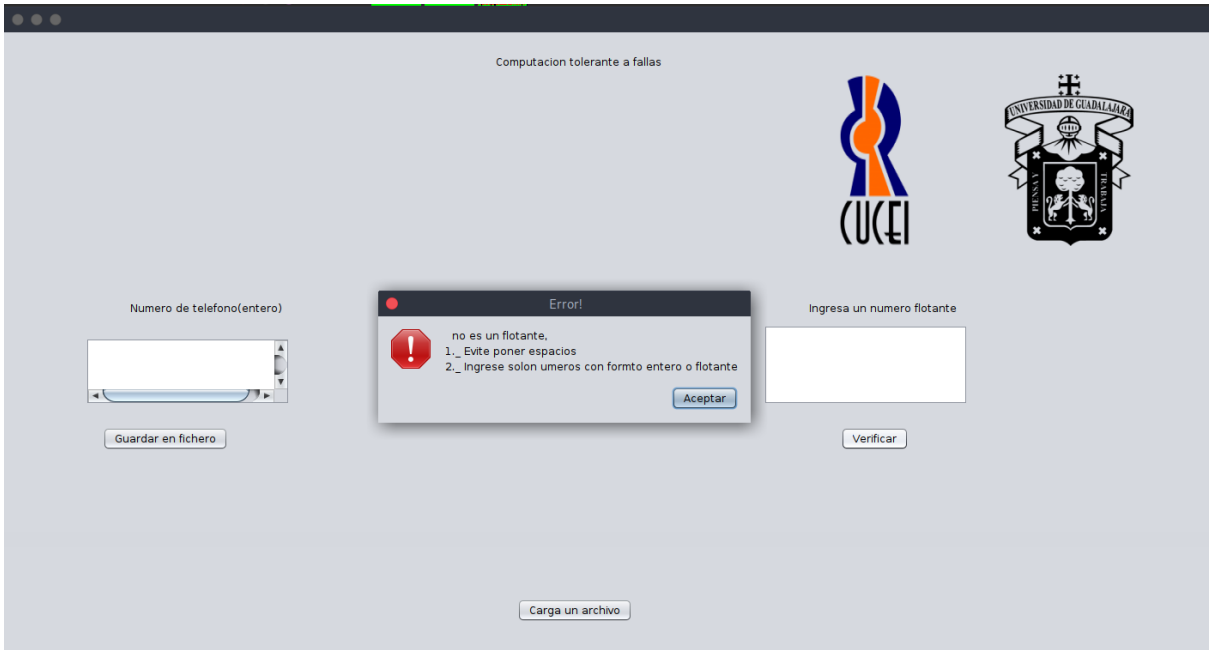
! 123..4 no es un flotante,
1._ Evite poner espacios
2._ Ingresa solon umeros con formto entero o flotante

Aceptar

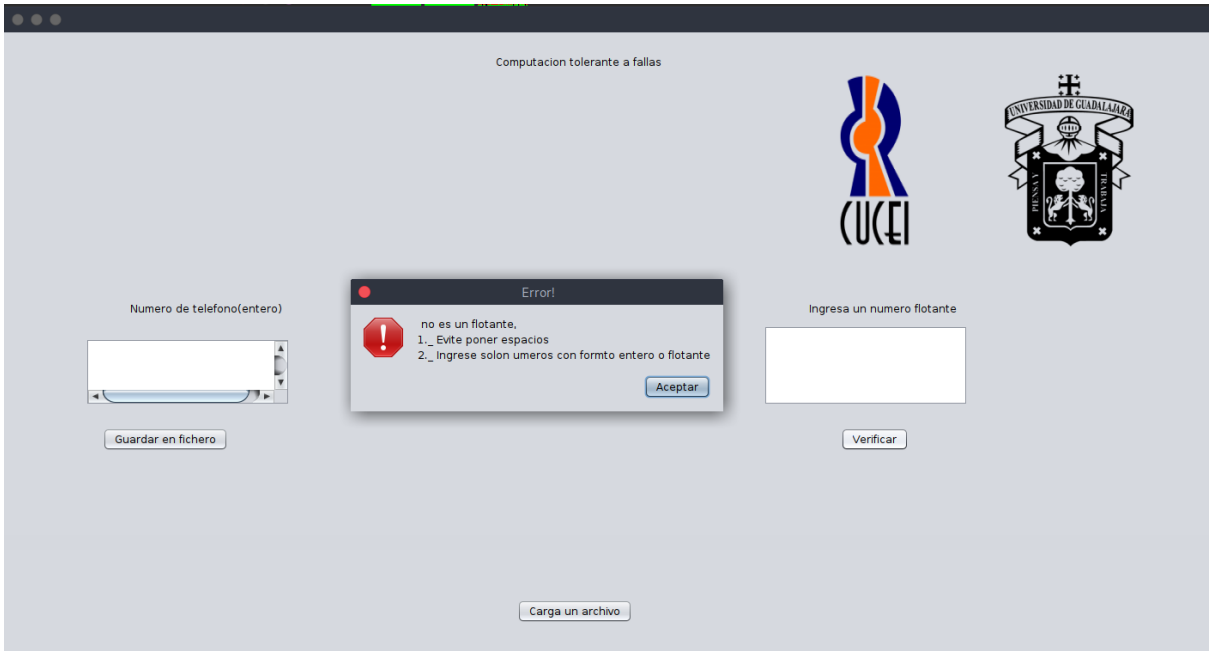
Entrada con espacio y número:



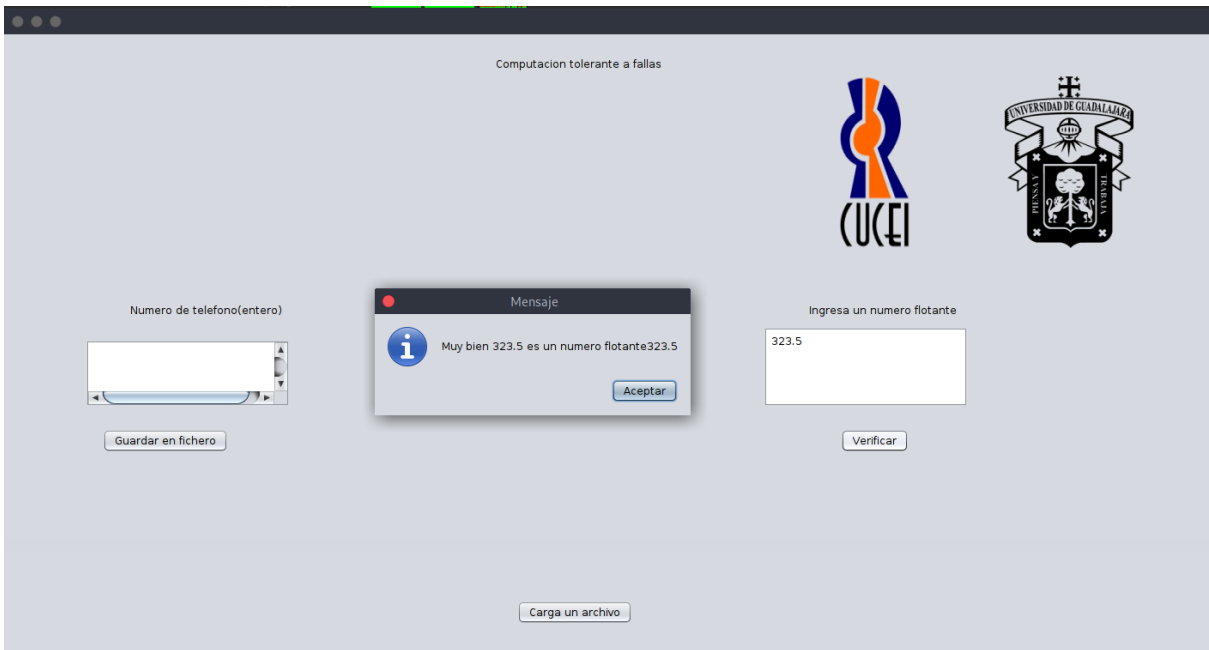
Entrada con espacio:



Entrada vacía:



Entrada válida float:



Validación de ficheros con la estructura try-catch:

Este tipo de validación es bastante común y nos permite atrapar errores para que el script pueda en lugar de morir hacer algo más razonable que consiste en capturar el error y hacer lo que nosotros le indiquemos, la estructura de un try-catch se basa en los siguientes puntos:

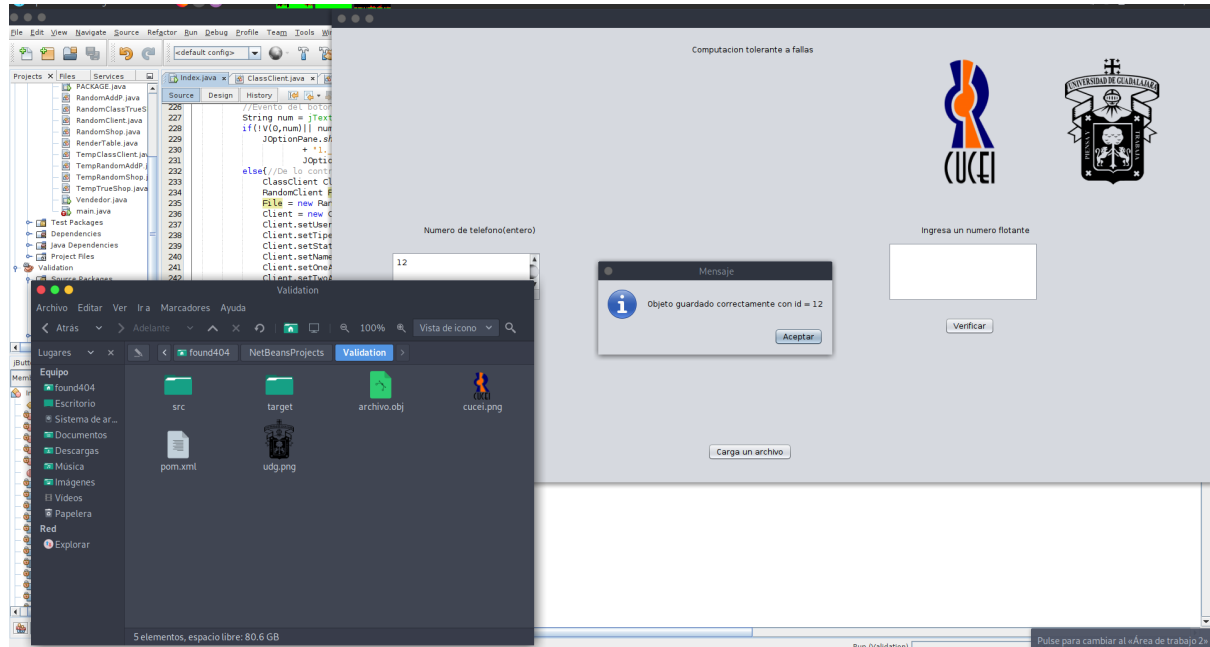
- 1) Primero es ejecutado el código dentro de try.
- 2) Si no se producen errores se ignora el bloque catch, la ejecución llega al final del try y continúa omitiendo el catch.
- 3) En el caso de que se produzca un error la ejecución try se detiene y el control fluye al comienzo de catch, la variable err contendrá un objeto de error con detalles sobre lo que ha ocurrido.

Para ejemplificar este método se trabaja con ficheros que usualmente generan errores, el código se muestra a continuación incluyendo el comentario de cada línea de código para su mejor entendimiento.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    //Evento del boton validar
    String num = jTextArea1.getText();//Captura del texto ingresado
    if(!V(0,num)|| num.length()==0)// Si no se ingresa un entero o si no se ingresa nada
        JOptionPane.showMessageDialog(this, num+" no es un entero, \n"//Mensaje de error, no
            + "1._ Evite poner espacios\n2._ Ingrese solo numeros enteros", "Error!",//es
            JOptionPane.ERROR_MESSAGE);//un numero
    else{//De lo contrario significa que es un numero.
        ClassClient Client;//Creacion de objeto cliente
        RandomClient File;//Creacion de objeto para manejo de ficheros
        File = new RandomClient();//Inicializacion de objeto File
        Client = new ClassClient();//Inicializacion de objeto cliente
        Client.setUserName("Izmael21");//Seteo de nombre al objeto cliente
        Client.setTipeUser("Estudiante");//Seteo de tipo de usuario al objeto cliente
        Client.setStatus("Activo");//Seteo de estatus al objeto cliente
        Client.setName("Izmael");//Seteo de nombre al objeto cliente
        Client.setOneApe("Guzman");//Seteo de primer apellido al objeto cliente
        Client.setTwoApe("Murguia");//Seteo del segundo apellido al objeto cliente
        Client.setId(Integer.parseInt(num));//Seteo de id al objeto cliente obtenido de
            //el campo de texto
        Client.setPassword("asdasdqw213dass");//Seteo de contrasena al objeto cliente
        Client.setAddress("Morelos 14");// Seteo de direccion al objeto cliente
        Client.setEmail("izmael.guzman4312@alumnos.udg.mx");//Seteo de email al objeto cliente
        Client.setPhone(12121);//Seteo de numero de telefono al objeto cliente
        try {//codigo a prueba de fallo
            File.AddEnd(Client);//Guardado del objeto en el fichero, posible fallo
            JOptionPane.showMessageDialog(this,"Objeto guardado correctamente con id = " + num);
            //Todo ok si se muestra el mensaje
        } catch (Exception ex) {//Ejecucion de instrucciones en caso de error
            JOptionPane.showMessageDialog(this, " Error con el fichero, el message de error es "
                +ex.getMessage(), "Error!", JOptionPane.ERROR_MESSAGE);
            //Mensaje informativo de error
        } //Fin de caso de error
    } //Fin de numero valido
}
```

Pruebas:

En esta fase se pretende poner a prueba el algoritmo o método de validación para hacer el software tolerante a fallos, para esto se proporciona solo un tipo de entrada ya que utiliza la validación de enteros que ya se mostró anteriormente, este tipo de validación se implementa en el manejo de archivos.



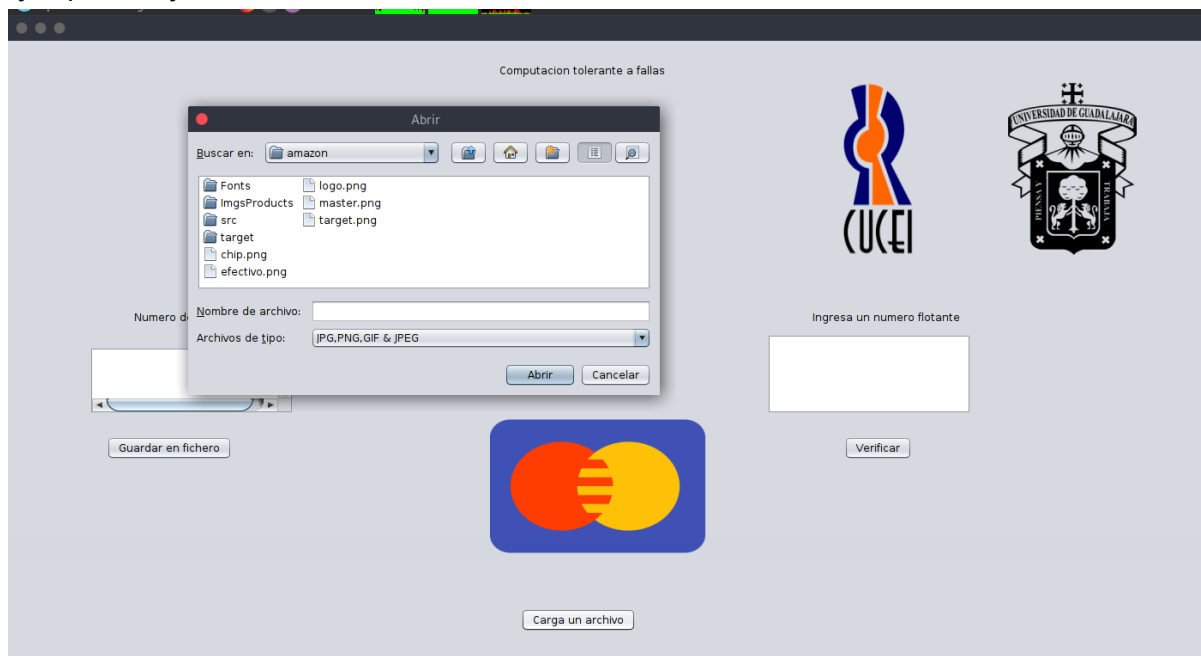
Validación para tipo de archivo especificado:

En ocasiones necesitamos específicamente un cierto tipo de archivos, no es posible asegurar que el usuario hará siempre todo bien, por ello se requerirá de un sistema que facilite o evite que el usuario se equivoque, es entonces cuando se hace un filtro asegurando así que si le pedimos una imagen nos de una imagen y no otro tipo de archivo como un .rar, lo que se utiliza en esta ocasión es un `filtered` que admite únicamente archivos del tipo, o con extensión que le indiquemos por ejemplo imágenes como se muestra en el código implementando únicamente archivos de tipo imagen.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser JFileChooser = new JFileChooser(); // Obtener que nos permita abrir un explorador de archivos
    FileNameExtensionFilter Filtred = new FileNameExtensionFilter("JPG,PNG,GIF & JPEG","jpg","png","gif","jpeg"); // Filtro de tipo de archivos.
    JFileChooser.setFileFilter(Filtred); // Le aplicamos el filtro al explorador de archivos
    int RequestImage = JFileChooser.showOpenDialog(this); // Verificador de elección de archivos
    if(RequestImage == JFileChooser.APPROVE_OPTION){ // Se compara si se seleccionó algo o no
        String urlImage = JFileChooser.getSelectedFile().getPath(); // Se obtiene la url de la imagen
        Image imageShow = new ImageIcon(urlImage).getImage(); // Se crea la imagen con la url
        ImageIcon MyImage = new ImageIcon(imageShow.getScaledInstance(jLabel3.getWidth(), jLabel3.getHeight(), Image.SCALE_SMOOTH)); // Se definen las
        jLabel3.setIcon(MyImage); // Se setea la imagen para que sea visible en la interfaz.
    } // Fin de caso de elección de imagen
}
```

Prueba:

Se hará una prueba cargando una imagen y veremos que efectivamente solo permite la selección de este tipo de archivo con extensión .png, .jpg, .jpeg y de mas pero siempre evitando errores de tipo de archivo, además se valida en caso de cancelar con el if ya que si no se hace genera una excepción al tener el campo url vacío, a continuación se muestra un ejemplo de ejecución.



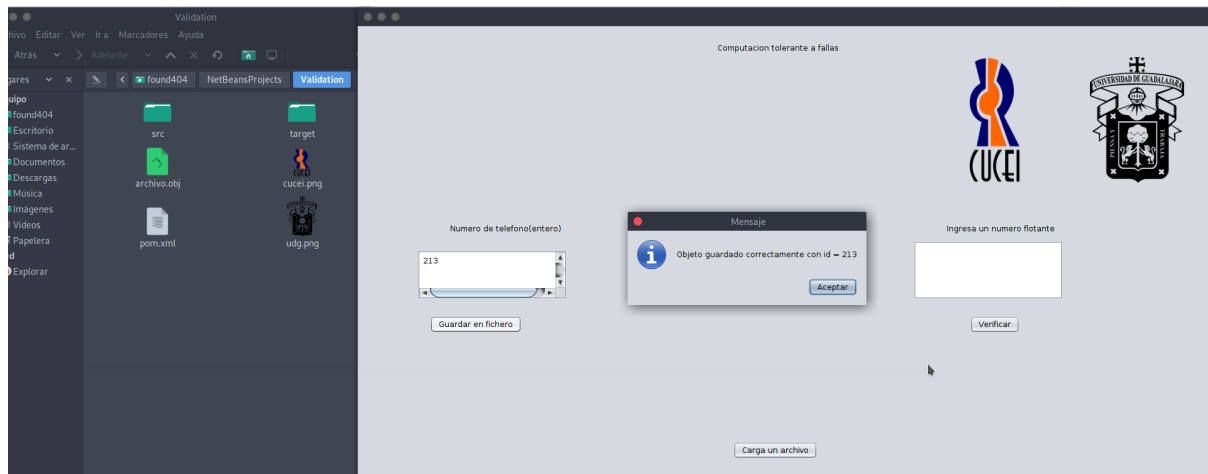
Validación de ficheros mediante throw:

La validación mediante throw es otro método muy sencillo que permite prevenir posibles errores, la diferencia es que ahora en caso de error ya no tendremos el código de mensaje, en base a esto podemos afirmar que debe utilizarse cuando ya conocemos el error posible y deseamos simplemente ignorarlo ya que no tenemos algo que hacer al respecto por que no lo consideramos conveniente, en resumidas palabras solo lo saltamos sin que se rompa el sistema, el código se muestra a continuación con sus respectivos comentarios para su mejor entendimiento.

```
public void AddEnd(ClassClient UserI) throws Exception { //Esta funcion
    //agrega un nuevo objeto al final pero ademas incorpora una posible
    //excepcion con throws Exception
    raf = new RandomAccessFile(file, "rw"); //Creacion de objeto
    //para leer
    raf.seek(raf.length()); //Se posiciona al final
    raf.writeUTF(UserI.getUserName()); //Escribe un string nombre de usuario
    raf.writeUTF(UserI.getTypeUser()); //Escribe un string tipo de usuario
    raf.writeUTF(UserI.getStatus()); //Escribe un string estatus de usuario
    raf.writeUTF(UserI.getName()); //Escribe un string nombre de usuario
    raf.writeUTF(UserI.getOneApe()); //Escribe un string primer apellido
    raf.writeUTF(UserI.getTwoApe()); //Escribe un string segundo apellido
    raf.writeInt(UserI.getId()); //Escribe un entero id de usuario
    raf.writeUTF(UserI.getPassword()); //Escribe un string contraseña
    raf.writeUTF(UserI.getAddress()); //Escribe un string direccion
    raf.writeUTF(UserI.getEmail()); //Escribe un string email
    raf.writeLong(UserI.getPhone()); //Escribe un long numero de telefono
    raf.close(); //Cierre del archivo de lectura
} //Fin de la funcion
}
```

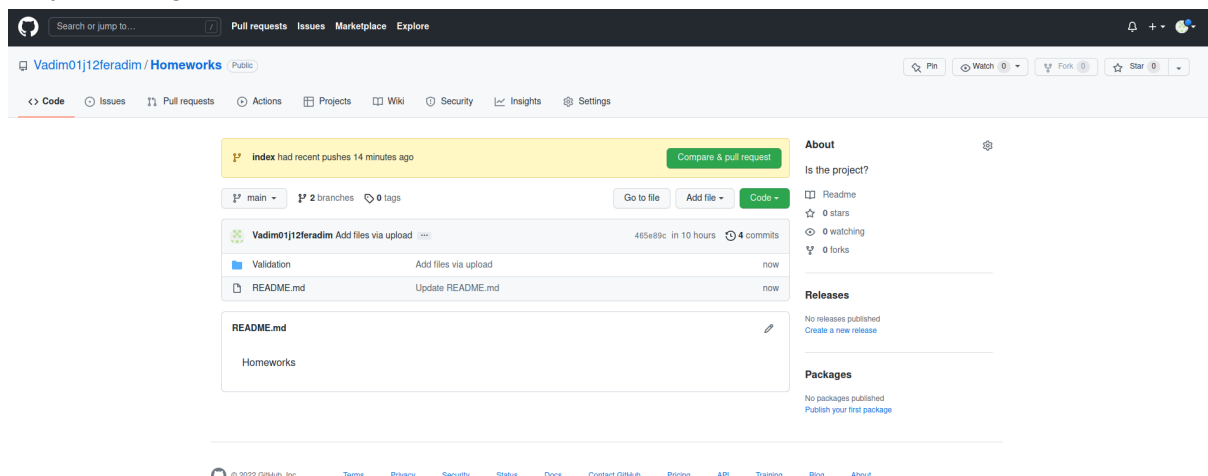

Prueba:

En esta fase se implementará solo una prueba ya de lo que se trata básicamente es de la escritura de los diferentes datos de un objeto en un fichero, para ello se utiliza una operación muy similar al try-catch pero la diferencia es que ahora no se captura el código de error, solo se evita, a continuación se muestra un ejemplo donde se utiliza esta operación del objeto File.



Repositorio de github:

Proyecto en github.



link: <https://github.com/Vadim01j12feradim/Homeworks.git>

Conclusión:

En esta actividad se implementaron algoritmos y estructuras que permiten validar las entradas y ciertos flujos del programa que pudieran ocasionar un fallo o error, tambien, ademas de las validaciones correspondientes se aplica también una método o modelo de programación lo más entendible posible mediante variables y comentarios que ayudan a estudiar el código de una manera correcta, además se practica en este programa el uso de indentado, para darle forma al código, esto es muy importante cuando se desea buscar errores e incluso actualizar el sistema añadiendo nuevas características o arreglando errores, principalmente se utiliza la estructura try-catch pero además algunos algoritmos propios para la validación de entradas de imágenes, campos en blanco y de más, en este sentido podemos decir que entre más metodologías eficaces apliquemos más robusto será nuestro sistema y por consiguiente para encontrar o provocar un error sera practicamente imposible, algo que valorara sin duda el cliente, recordando que uns sistema robusto es aquel capaz de funcionar o seguir operando incluso cuando se le proporcionan datos deentrada que idealmente no deberían o también en su caso ciertas condiciones externas que puedan provocar un comportamiento que no deseamos. Cabe mencionar que aunque en esta ocasión se utilice un try-catch existen muchas maneras de validar nuestros programas y hacerlos más robustos, algo que se menciona comúnmente en el área de desarrollo es que ningún software está exento de error, lo que sí es posible es reducirlos en gran medida si aplicamos ciertas técnicas como las mencionadas anteriormente, pero también es importante que se tenga una buena fase de elicitación de requisitos, esto debido a que si no se tiene este punto lo suficientemente correcto aunque tengamos nuestro software bien validado contendrá errores en cuanto a que no le estaremos dando al cliente lo que él nos pide, recordando que lo que él nos pide es lo que necesita y el producto no solo pudiera llegar a ser inapropiado si no que en su totalidad no útil para su fin solicitado por los usuarios.

Bibliografía:

- Manejo de errores, &., 2022. *Manejo de errores, "try...catch"*. [online] Es.javascript.info. Available at: <<https://es.javascript.info/try-catch>> [Accessed 3 February 2022].
- Swami, S., 2022. *Validación de correo electrónico en Java*. [online] Delft Stack. Available at: <<https://www.delftstack.com/es/howto/java/email-validation-method-in-java/>> [Accessed 3 February 2022].
- Elcodigoascii.com.ar. 2022. *El código ASCII Completo, tabla con los codigos ASCII completos, caracteres simbolos letras ascii, ascii codigo, tabla ascii, codigos ascii, caracteres ascii, codigos, tabla, caracteres, simbolos, control, imprimibles, extendido, letras, vocales, signos, simbolos, mayusculas, minusculas, alt, teclas, acentos, agudo, grave, eñe, enie, arroba, dieresis, circunflejo, tilde, cedilla, anillo, libra, esterlina, centavo, teclado, tipear, escribir, español, ingles, notebook, laptop, asccii, asqui, askii, aski,20220202*. [online] Available at: <<https://elcodigoascii.com.ar/>> [Accessed 3 February 2022].
- Dis.um.es. 2022. *Tutorial de Java - Captura de Excepciones*. [online] Available at: <<http://dis.um.es/~bmoros/Tutorial/parte9/cap9-3.html>> [Accessed 3 February 2022].
- Aprenderaprogramar.com. 2022. *Igual a cero validar si JTextField java está vacío como comprobar campo texto*. [online] Available at: <<https://aprenderaprogramar.com/foros/index.php?topic=1235.0>> [Accessed 3 February 2022].
- Atlassian. 2022. *Instalación de Git | Atlassian Git Tutorial*. [online] Available at: <<https://www.atlassian.com/es/git/tutorials/install-git>> [Accessed 3 February 2022].