

LABORATORIO # 4
PROGRAMACIÓN ORIENTADA A OBJETOS:
EXCEPCIONES

PRESENTADO A:
MARIA IRMA DIAZ ROSO

PRESENTADO POR:

CAROLINA CEPEDA

UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO

BOGOTA D.C

2025-1

PRACTICANDO MDD y BDD con EXCEPCIONES

En este punto vamos a aprender a diseñar, codificar y probar usando excepciones. Para esto se van

a trabajar algunos métodos de la clase Complete

1. En su directorio descarguen los archivos contenidos en units.zip revisen el contenido y

estudien el diseño estructural de la aplicación (únicamente la zona en azul).

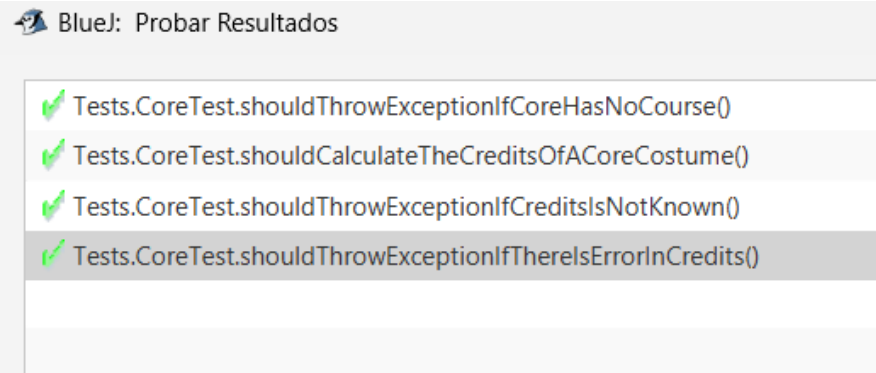
2. Expliquen por qué el proyecto no compila. Realicen las adiciones necesarias para lograrlo.

El proyecto no compila debido a que la clase de excepciones Exception15 no está planteada. Se crea la clase Exception15 con sus respectivos mensajes:

```
9 public class Plan15Exception extends Exception
0 {
1
2     public static final String IMPOSSIBLE= "imposible";
3     public static final String CREDITS_UNKNOWN = "creditos no conocidos";
4     public static final String CREDITS_ERROR = " error en los creditos";
5     public static final String IN_PERSON_UNKNOWN = "persona no reconocida";
6     public static final String IN_PERSON_ERROR = "error de persona";
7
8     public Plan15Exception(String message)
9     {
```

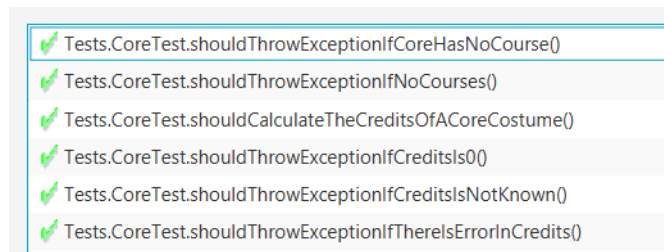
3. Dado el diseño y las pruebas documenten y codifiquen el método credits().

```
/*
 * Sums the credits of the courses that the core has
 *
 * @return int
 *
 * @throws Plan15Exception IMPOSSIBLE, if there are no courses
 */
@Override
public int credits() throws Plan15Exception {
    if (courses.isEmpty()) {
        throw new Plan15Exception(Plan15Exception.IMPOSSIBLE);
    }
    int totalCreditos = 0;
    for (Course c : courses) {
        totalCreditos += c.credits();
    }
    return totalCreditos;
};
```

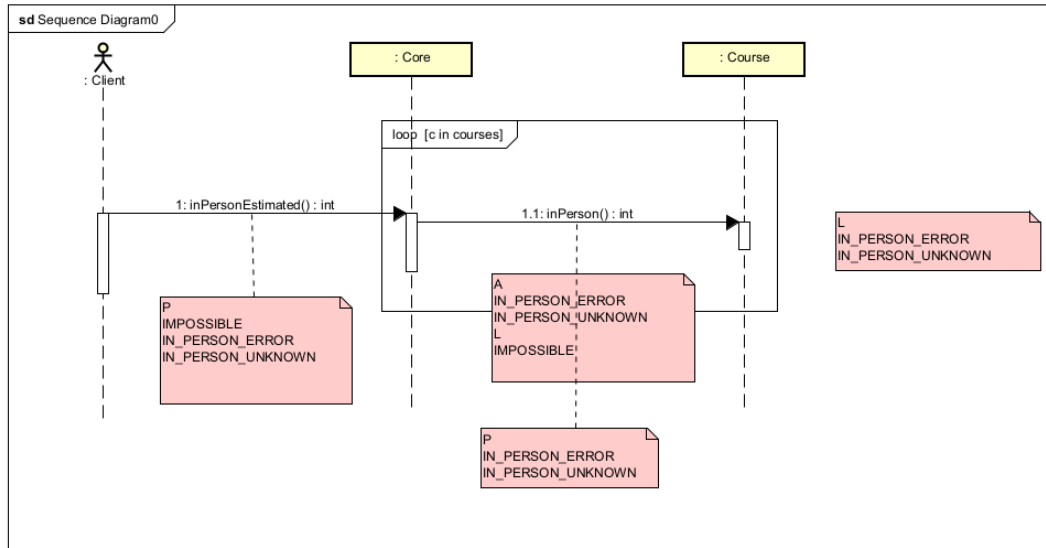


4. Dada la documentación y el diseño, codifiquen y prueben el método `creditsEstimated()`.

```
/**
 * Calculate the actual or estimated credits
 * If necessary (unknown or error), assumes the number of credits is 3
 * is equal to the in-person hours.
 *
 * @return
 * @throws Plan15Exception IMPOSSIBLE, If there are no credits
 */
public int creditsEstimated() throws Plan15Exception {
    int suma = 0;
    if (courses.isEmpty()) {
        throw new Plan15Exception(Plan15Exception.IMPOSSIBLE);
    }
    for (Course c : courses) {
        try {
            suma += c.credits();
        } catch (Plan15Exception e) {
            suma += 3;
        }
    }
    return suma;
}
```



5. Documenten, diseñen, codifiquen y prueben el método `inPersonEstimated()`.



```

/**
 * Calculate the estimated in-person hours, considering courses that do not have
 * credit issues.
 * If the hours of a course are not known, calculate the course in-person hours
 * using the percentage suggested in the unit core.
 * If the hours of a course are incorrect, it is assumed that all the time is in
 * person.
 *
 * @return int horas
 * @throws Plan15Exception IMPOSSIBLE, If there are no courses or all courses
 *       has credit issues
 */
public int inPersonEstimated() throws Plan15Exception {
    int horas = 0;
    boolean cursoValido = false;

    if (courses.isEmpty()) {
        throw new Plan15Exception(Plan15Exception.IMPOSSIBLE);
    }

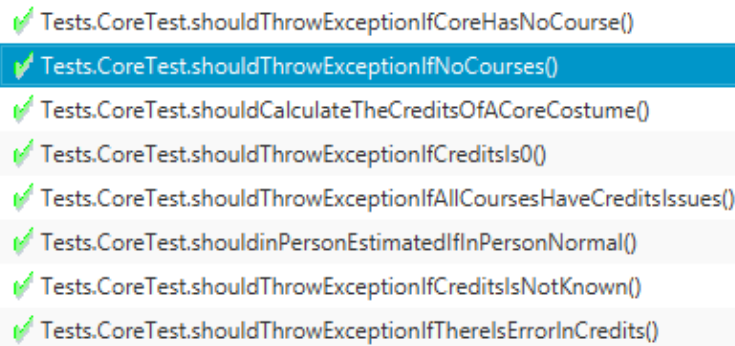
    for (Course c : courses) {
        try {
            horas += c.inPerson();
            cursoValido = true;
        } catch (Plan15Exception e) {
            if (e.getMessage().equals(Plan15Exception.IN_PERSON_UNKNOWN)) {
                horas += c.credits() * inPersonPercentage;
                cursoValido = true;
            }

            if (e.getMessage().equals(Plan15Exception.IN_PERSON_ERROR)) {
                horas += c.credits();
                cursoValido = true;
            }
        }
    }

    if (!cursoValido) {
        throw new Plan15Exception(Plan15Exception.IMPOSSIBLE);
    }

    return horas;
}

```



```
✓ Tests.CoreTest.shouldThrowExceptionIfCoreHasNoCourse()
✓ Tests.CoreTest.shouldThrowExceptionIfNoCourses()
✓ Tests.CoreTest.shouldCalculateTheCreditsOfACoreCostume()
✓ Tests.CoreTest.shouldThrowExceptionIfCreditsIs0()
✓ Tests.CoreTest.shouldThrowExceptionIfAllCoursesHaveCreditsIssues()
✓ Tests.CoreTest.shouldinPersonEstimatedIfInPersonNormal()
✓ Tests.CoreTest.shouldThrowExceptionIfCreditsIsNotKnown()
✓ Tests.CoreTest.shouldThrowExceptionIfThereIsErrorInCredits()
```

CONCIENDO EL PROYECTO PLAN 15

1. En su directorio descarguen los archivos contenidos en plan15.zip, revisen el contenido. ¿Cuántos archivos se tienen? ¿Cómo están organizados? ¿Cómo deberían estar organizados?

En la carpeta se encuentran 3 archivos, Log.java, Plan15.java y Plan15GUI.java, los tres están organizados de forma alfabética dentro de la carpeta Plan15J pero deberían estar organizados en dos paquetes: dominio y presentación

2. Estudien el diseño del programa: diagramas de paquetes y de clases. ¿cuántos paquetes tenemos? ¿cuántas clases tiene el sistema? ¿cómo están organizadas? ¿cuál es la clase ejecutiva?

Se tienen 2 paquetes : domain y presentation (a su vez, se tienen los paquetes estándar de java). A su vez , se tienen 5 clases, de las cuales Core , Course, Plan15 y Unit son parte del paquete de dominio .Su clase Ejecutiva es Plan15GUI.

3. Prepare los directorios necesarios para ejecutar el proyecto. ¿qué estructura debe tener? ¿qué clases deben tener? ¿dónde están esas clases? ¿qué instrucciones debe dar para ejecutarlo?

Debe tener una estructura siguiendo los directorios src, bin y docs:

En cada uno de ellos se debe tener los dos paquetes y las clases mencionadas anteriormente, de las cuales Course, Core y Unit se encuentran en la carpeta Units usada en la sección anterior.

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> New-Item -ItemType Directory -Path ".\src", ".\bin"
```

Directorio: C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units

Mode	LastWriteTime	Length	Name
d-----	31/03/2025 22:40		src
d-----	31/03/2025 22:40		bin

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> New-Item -ItemType Directory -Path ".\src\domain", ".\src\presentation"
```

Directorio: C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units\src

Mode	LastWriteTime	Length	Name
d-----	31/03/2025 22:43		domain
d-----	31/03/2025 22:43		presentation

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> New-Item -ItemType Directory -Path ".\bin\domain", ".\bin\presentation"
```

Directorio: C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units\bin

Mode	LastWriteTime	Length	Name
d-----	31/03/2025 22:43		domain
d-----	31/03/2025 22:43		presentation

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> Move-Item -Path ".\domain\*.java" -Destination ".\src\domain\"
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> Move-Item -Path ".\presentation\*.java" -Destination ".\src\presentation\"
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> Move-Item -Path ".\domain\*.class" -Destination ".\bin\domain\"
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> Move-Item -Path ".\presentation\*.class" -Destination ".\bin\presentation\"
```

```
Windows PowerShell
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\Units> Remove-Item -Path ".\domain", ".\presentation" -Recurse -Force
```

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4> remove-Item -Path ".\plan15J" -Recurse -Force
```

```
PS C:\Users\Karol\OneDrive\Documentos\GitHub\EntregasPoob\Lab4\plan15> javadoc -d docs src\domain\*.java src\presentation\*.java
Loading source file src\domain\Course.java...
Loading source file src\domain\Log.java...
Loading source file src\domain\Plan15.java...
Loading source file src\domain\Plan15Exception.java...
Loading source file src\domain\Unit.java...
Loading source file src\presentation\Plan15GUI.java...
Constructing Javadoc information...
Creating destination directory: "docs\"
Building index for all the packages and classes...
Standard Doclet version 24+36-3646
Building tree for all the packages and classes...
src\domain\Log.java:11: warning: empty comment

```

4. Ejecute el proyecto, ¿qué funcionalidades ofrece? ¿cuáles funcionan?

El proyecto tiene 3 funcionalidades: listar, adicionar y buscar. De las cuales, listar y adicionar tienen dos botones: uno para hacer la función de añadidura y el otro para limpiar. En estas funcionalidades, se permite limpiar lo escrito en los campos dados y se permite adicionar o listar. En cambio, en la función de buscar no se permite hacer la búsqueda.

5. Revisen el código y la documentación del proyecto. ¿De dónde salen las unidades iniciales? ¿Qué clase pide que se adicionen? ¿Qué clase los adiciona?

Las unidades iniciales son parte de la variable plan que es de la clase Plan15, la clase que pide que se adicionen es Plan15GUI y la clase que las adiciona es Plan 15.

```
private Plan15GUI(){
    plan=new Plan15();
    prepareElements();
    prepareActions();
}
```

```
public class Plan15{
    private ArrayList<Unit> units;
    private TreeMap<String, Course> courses;

    /**
     * Create a Plan15
     */
    public Plan15(){
        units = new ArrayList<Unit>();
        courses = new TreeMap<String, Course>();
        addSome();
    }

    private void addSome(){
        String [][] courses = {{ "PRI1", "Proyecto Integrador", "9", "3"},
                                {"DDYA", "Diseño de Datos y Algoritmos", "4", "4"},
                                {"MPIN", "Matematicas para Informatica", "3", "4"};
        for (String [] c: courses){
            addCourse(c[0],c[1],c[2],c[3]);
        }
        String [][] Core = {{ "FCC", "Nucleo formacion por comun por campo", "50", "PRI1\nDDYA\nMPIN"};
        for (String [] c: Core){
            addCore(c[0],c[1],c[2],c[3]);
        }
    }
}
```

```
/**
 * Add a new course
 */
public void addCourse(String code, String name, String credits, String inPerson){
    Course nc=new Course(code,name,Integer.parseInt(credits),Integer.parseInt(inPerson));
    units.add(nc);
    courses.put(code.toUpperCase(),nc);
}

/**
 * Add a new core
 */
public void addCore(String code, String name, String percentage, String theCourses){
    Core c = new Core(code,name,Integer.parseInt(percentage));
    String [] aCourses= theCourses.split(regex:"\\n");
    for (String b : aCourses){
        c.addCourse(courses.get(b.toUpperCase()));
    }
    units.add(c);
}
```

ADICIONAR Y LISTAR. TODO OK.

1. Adicionen un nuevo curso y un nuevo núcleo

Curso

DOSW Desarrollo y Operaciones Software 4 y 4

Core

NFPE Nucleo de formación específica 100

DOSW

¿Qué ocurre? ¿Cómo lo comprueban? Capturen la pantalla. ¿Es adecuado este comportamiento?

El curso y el núcleo se añaden a la lista, esto se comprueba en listar .

```
6 unidades
>PRI1: Proyecto Integrador. Creditos:9[3+24]
>DDYA: Diseño de Datos y Algoritmos. Creditos:4[4+8]
>MPIN: Matematicas para Informatica. Creditos:3[4+5]
>FCC: Nucleo formacion por comun por campo. [50%]
    PRI1: Proyecto Integrador. Creditos:9[3+24]
    DDYA: Diseño de Datos y Algoritmos. Creditos:4[4+8]
    MPIN: Matematicas para Informatica. Creditos:3[4+5]
>DOSW: Desarrollo y Operaciones Software. Creditos:4[4+8]
>NFPE: Nucleo de Formación Específica. [100%]
    DOSW: Desarrollo y Operaciones Software. Creditos:4[4+8]
```

A pesar de que si se hace la añadidura del curso y núcleo de forma exitosa podría ser mas conveniente que los cursos se añadieran únicamente en sus núcleos.

2. *Revisen el código asociado a adicionar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?*

```
private void actionAdd(){
    if (basics.getText().trim().equals(anObject:"")){
        plan.addCourse(code.getText(),name.getText(),credits.getText(),inPerson.getText());
    }else{
        plan.addCore(code.getText(),name.getText(),credits.getText(),basics.getText());
    }
}
```



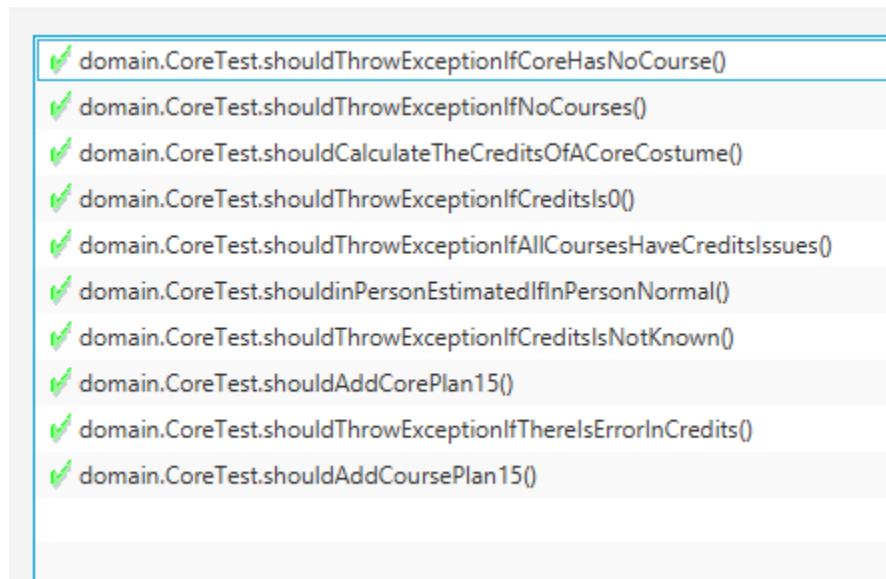
```

/**
 * Add a new course
 */
public void addCourse(String code, String name, String credits, String inPerson){
    Course nc=new Course(code,name,Integer.parseInt(credits),Integer.parseInt(inPerson));
    units.add(nc);
    courses.put(code.toUpperCase(),nc);
}

/**
 * Add a new core
 */
public void addCore(String code, String name, String percentage, String theCourses){
    Core c = new Core(code,name,Integer.parseInt(percentage));
    String [] aCourses= theCourses.split(regex:"\\n");
    for (String b : aCourses){
        c.addCourse(courses.get(b.toUpperCase()));
    }
    units.add(c);
}

```

3. Realicen ingeniería reversa para la capa de dominio para adicionar. Capturen los resultados de las pruebas de unidad.



4. Revisen el código asociado a listar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método en la capa de dominio?

```
private void actionList(){
    textDetails.setText(plan.toString());
}
```

```
/**
 * Return the data of all units
 * @return
 */
public String toString(){
    return data(units);
}
```

El método responsable en la capa presentación es actionList() y en la capa dominio es toString().

5. Realicen ingeniería reversa para la capa de dominio para listar.

Capturen los resultados de las pruebas de unidad.

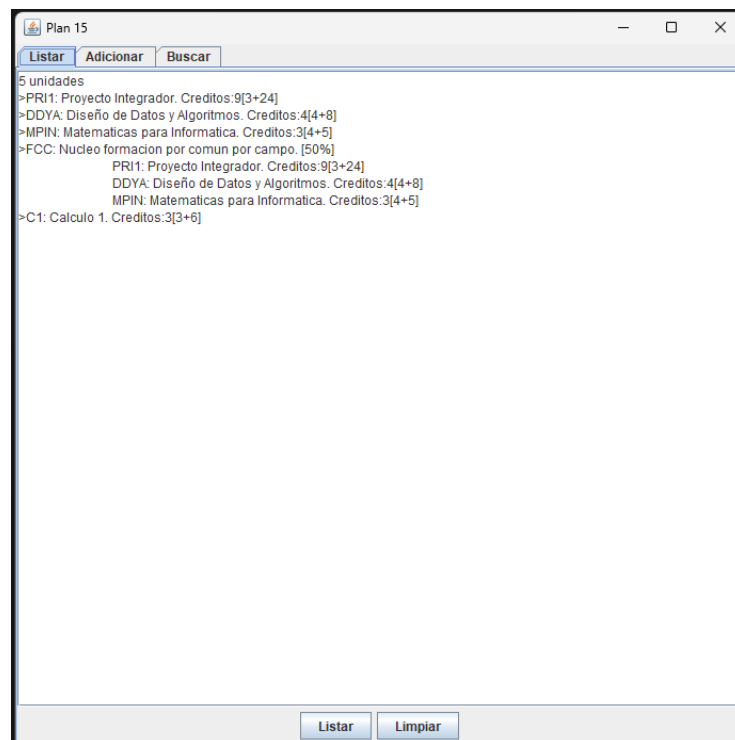
```
● PS C:\Users\Karol\OneDrive\Desktop> java -jar junit.jar
● PS C:\Users\Karol\OneDrive\Desktop> java -jar junit.jar
JUnit version 4.13.2
.....
Time: 0,042

OK (20 tests)
```

6. Propongan y ejecuten una prueba de aceptación.

Run | Debug | Run main | Debug main

```
public static void main(String args[]){  
    Plan15GUI gui=new Plan15GUI();  
    gui.setVisible(b:true);  
  
    gui.code.setText(t:"C1");  
    gui.name.setText(t:"Calculo 1");  
    gui.credits.setText(t:"3");  
    gui.inPerson.setText(t:"3");  
    gui.basics.setText(t:"");  
  
    gui.buttonAdd.doClick();  
  
    gui.buttonList.doClick();  
  
}  
}
```



Adicionar una unidad. Funcionalidad robusta

Caso A: ¿Y si el nombre de la unidad no existe?

1. Propongan una prueba de aceptación que genere el fallo.

```
Run | Debug | Run main | Debug main
public static void main(String args[]){
    Plan15GUI gui=new Plan15GUI();
    gui.setVisible(b:true);

    gui.code.setText(t:"C1");
    gui.name.setText(t:"Calculo 1");
    gui.credits.setText(t:"3");
    gui.inPerson.setText(t:"3");
    gui.basics.setText(t:"");
    gui.buttonAdd.doClick();

    gui.code.setText(t:"CAPA");
    gui.name.setText(t:"");
    gui.credits.setText(t:"10");
    gui.inPerson.setText(t:"2");
    gui.basics.setText(t:"");
    gui.buttonAdd.doClick();
    gui.buttonList.doClick();
}
```

Si el nombre del curso se coloca como vacío no pone ningún nombre, si no se define el curso lo guarda con el nombre de la anterior unidad guardada. Al listar se ve de la siguiente manera :

```
mi iv. matemáticas pt
>C1: Calculo 1. Creditos:3[3+6]
>CAPA: . Creditos:10[2+28]
```

2. Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.
Propaga y Lanza : addCourse, addCore
Atiende actionAdd
3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

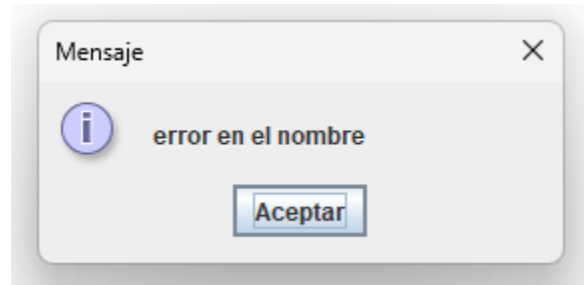
```

domain.CoreTest.shouldThrowExceptionCoreNameError()
domain.CoreTest.shouldThrowExceptionCourseNameError()

```

4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1.
¿Qué sucede ahora? Capture la pantalla

Se muestra un mensaje diciendo que el nombre es erroneo.



Caso B: ¿Y si los valores enteros no son enteros?

1. Propongan una prueba de aceptación que genere el fallo.

```

public static void main(String args[]){
    Plan15GUI gui=new Plan15GUI();
    gui.setVisible(true);
    gui.code.setText("CAPA2");
    gui.name.setText("caso prueba 2");
    gui.credits.setText("10.3");
    gui.inPerson.setText("2.3");
    gui.basics.setText("");
    gui.buttonAdd.doClick();
    gui.buttonList.doClick();
}

```

```

Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "10.3"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:662)

```

2. Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

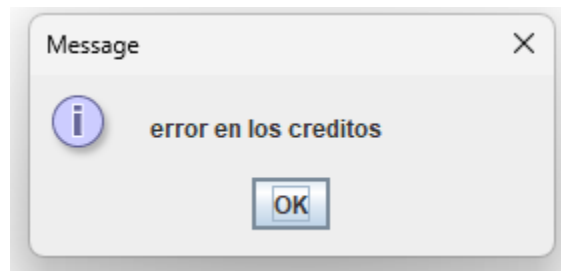
Los métodos que debería lanzar y propagar la excepción son `addCourse` y `AddCore`, y el método que debe atenderla es `actionAdd()`.

3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

```
PS C:\Users\Karol\OneDrive\Documents> java -jar JUnit.jar
PS C:\Users\Karol\OneDrive\Documents> java -jar JUnit.jar
JUnit version 4.13.2
.....
Time: 0,042

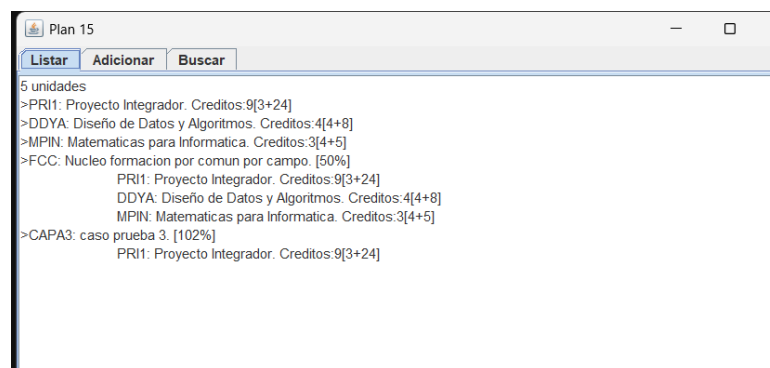
OK (20 tests)
```

4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Capture la pantalla.



Caso C: ¿Y si el porcentaje no está entre 0 y 100?

1. Propongan una prueba de aceptación que genere el fallo.



2. Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.

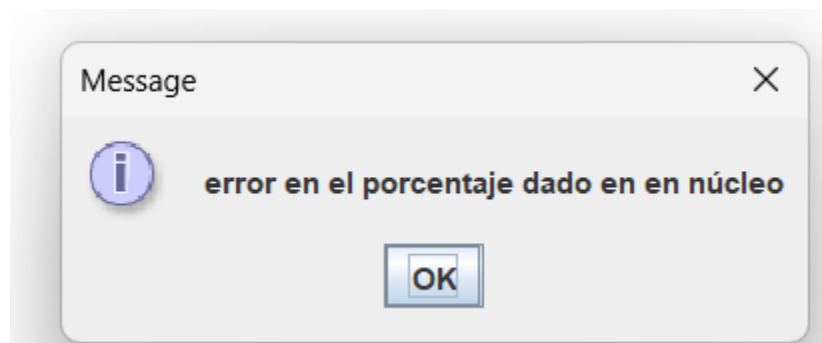
Los métodos que debería lanzar y propagar la excepción son `addCourse` y `AddCore`, y el método que debe atenderla es `actionAdd()`.

3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

```
PS C:\Users\Karo1\OneDrive
PS C:\Users\Karo1\OneDrive
JUnit version 4.13.2
.....
Time: 0,042

OK (20 tests)
```

4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1. ¿Qué sucede ahora? Capture la pantalla.



Caso D: ¿Y si el nombre de la unidad ya existe?

1. Propongan una prueba de aceptación que genere el fallo.

```
Run | Debug
public static void main(String args[]){
    Plan15GUI gui=new Plan15GUI();
    gui.setVisible(b:true);
    gui.code.setText(t:"CAPA4");
    gui.name.setText(t:"caso prueba 4");
    gui.credits.setText(t:"10");
    gui.inPerson.setText(t:"2");
    gui.basics.setText(t:"");
    gui.buttonAdd.doClick();

    gui.code.setText(t:"CAPA4");
    gui.name.setText(t:"caso prueba 4");
    gui.credits.setText(t:"10");
    gui.inPerson.setText(t:"2");
    gui.basics.setText(t:"");
    gui.buttonAdd.doClick();
    gui.buttonList.doClick();
}

Plan 15
Listar Adicionar Buscar
6 unidades
>PRI1: Proyecto Integrador. Creditos:9[3+24]
>DDYA: Diseño de Datos y Algoritmos. Creditos:4[4+8]
>MPIN: Matematicas para Informatica. Creditos:3[4+5]
>FCC: Nucleo formacion por comun por campo. [50%]
    PRI1: Proyecto Integrador. Creditos:9[3+24]
    DDYA: Diseño de Datos y Algoritmos. Creditos:4[4+8]
    MPIN: Matematicas para Informatica. Creditos:3[4+5]
>CAPA4: caso prueba 4. Creditos:10[2+28]
>CAPA4: caso prueba 4. Creditos:10[2+28]
```

2. *Analicen el diseño realizado. Para hacer el software robusto: ¿Qué método debería lanzar la excepción? ¿Qué métodos deberían propagarla? ¿Qué método debería atenderla? Explique claramente.*

Los métodos addCourse y addCore deberían lanzar y propagar la excepción. El método que debe atenderla es actionAdd.

3. *Construya la solución propuesta. Capture los resultados de las pruebas de unidad.*

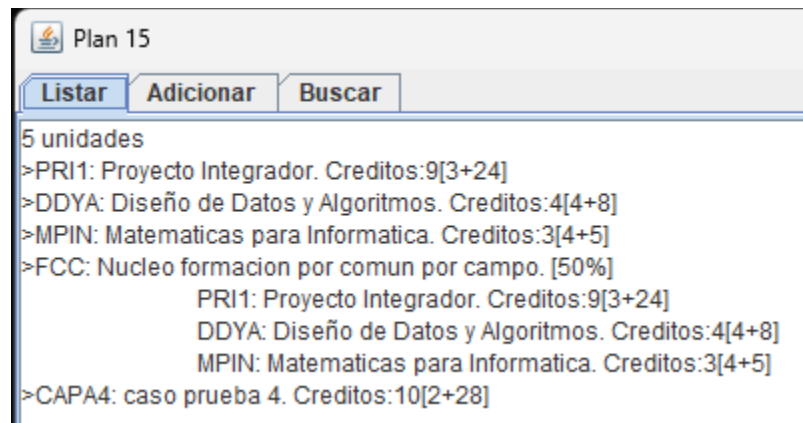
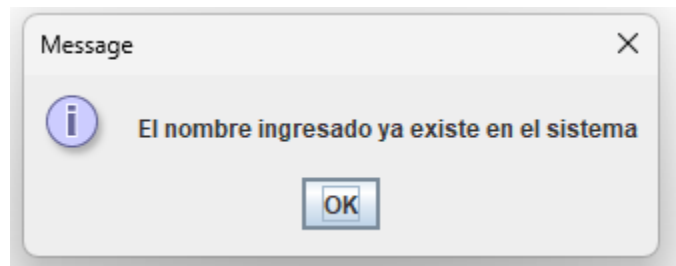

```

● PS C:\Users\Karol\OneDrive
● PS C:\Users\Karol\OneDrive
JUnit version 4.13.2
.....
Time: 0,042

OK (20 tests)

```

4. Ejecuten nuevamente la aplicación con el caso de aceptación propuesto en 1.
¿Qué sucede ahora? Capture la pantalla.



```

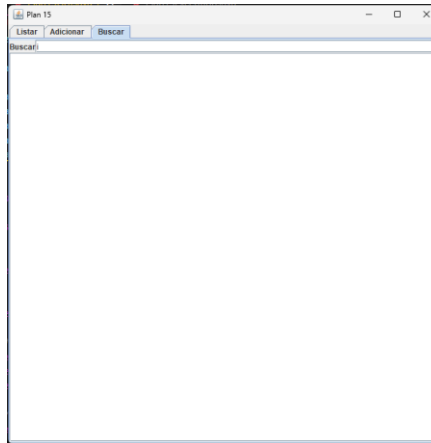
● PS C:\Users\Karol\OneDrive
● PS C:\Users\Karol\OneDrive
JUnit version 4.13.2
.....
Time: 0,042

OK (20 tests)

```

Consultando por patrones. ¡ No funciona y queda sin funcionar!

1. Consulten una unidad que inicie con I. ¿Qué sucede? ¿Qué creen que pasó? Capturen el resultado. ¿Quién debe conocer y quien NO debe conocer esta información?



```
at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4996)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4828)
at java.desktop/java.awt.KeyboardFocusManager.redispatchEvent(KeyboardFocusManager.java:1952)
at java.desktop/java.awt.DefaultKeyboardFocusManager.dispatchEvent(DefaultKeyboardFocusManager.java:883)
at java.desktop/java.awt.DefaultKeyboardFocusManager.preDispatchKeyEvent(DefaultKeyboardFocusManager.java:1146)
at java.desktop/java.awt.DefaultKeyboardFocusManager.typeAheadAssertions(DefaultKeyboardFocusManager.java:1028)
at java.desktop/java.awt.DefaultKeyboardFocusManager.dispatchEvent(DefaultKeyboardFocusManager.java:848)
at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4877)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2780)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4828)
at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:775)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:720)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:714)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:400)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:87)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:747)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:745)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:400)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:87)
at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:744)
at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:109)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
```

Al consultar no muestra ninguna información al usuario de forma directa en la GUI , pero si se muestra los eventos sucedidos en terminal, esta información la debe conocer solamente los programadores y desarrolladores del software, los usuarios no deberían de tener acceso a esta información.

2. Exploren el método record de la clase Log ¿Qué servicio presta?

El servicio que presta es el registro de errores o excepciones, por lo que guarda un archivo de errores que ocurren durante la ejecución del programa.

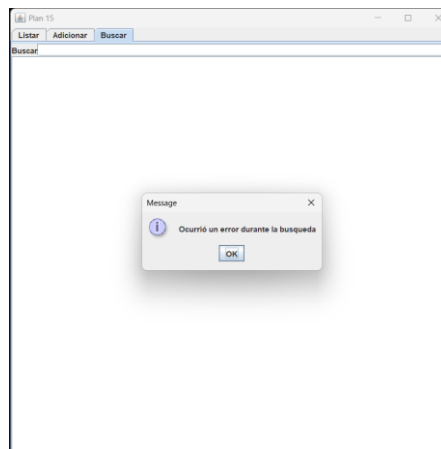
2. Analicen el punto adecuado para que EN ESTE CASO se presente un mensaje especial de alerta al usuario, se guarde la información del error en el registro y continúe la ejecución. Expliquen y construyan la solución.

Se atiende la excepción en el método actionSearch

```
private void actionSearch(){
    String patronBusqueda=textSearch.getText();
    String answer = "";

    try{
        if(patronBusqueda.length() > 0) {
            answer = plan.search(patronBusqueda);
        }
        textResults.setText(answer);
    }
    catch(Exception ex){
        JOptionPane.showMessageDialog(this,message:"Ocurrió un error durante la busqueda");
        textResults.setText("");
        Log.record(ex);
    }
}
```

3. Ejecuten nuevamente la aplicación con el caso propuesto en 1. ¿Qué mensaje salió en pantalla? ¿La aplicación termina? ¿Qué información tiene el archivo de errores?



La aplicación no termina


```

private void actionList() {
    try {
        textDetails.setText(plan.toString());
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(this, Plan15Exception.LIST_ERROR);
        Log.record(e);
    }
}

private void actionAdd() {
    try {
        if (basics.getText().trim().equals(anObject: "")) {
            plan.addCourse(code.getText(), name.getText(), credits.getText(), inPerson.getText());
        } else {
            plan.addCore(code.getText(), name.getText(), credits.getText(), basics.getText());
        }
    } catch (Plan15Exception e) {
        JOptionPane.showMessageDialog(this, e.getMessage());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, Plan15Exception.ADD_ERROR);
        Log.record(e);
    }
}

```

```

private void actionSearch(){
    String patronBusqueda=textSearch.getText();
    String answer = "";

    try{
        if(patronBusqueda.length() > 0) {
            answer = plan.search(patronBusqueda);
        }
        textResults.setText(answer);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(this, Plan15Exception.SEARCH_ERROR);
        Log.record(e);
    }
}

```

Se encapsularon los métodos de acciones de la GUI en bloques try-catch para garantizar que cualquier excepción sea capturada y tratada de forma segura, siendo guardada en el log en caso de ser una excepción que no sea parte de las esperadas en Plan15Exception.

Consultando por patrones. ¡Ahora sí funciona!

1. Revisen el código asociado a buscar en la capa de presentación y la capa de dominio. ¿Qué método es responsable en la capa de presentación? ¿Qué método es responsable en la capa de dominio?

```
private void actionSearch(){
    String patronBusqueda=textSearch.getText();
    String answer = "";

    try{
        if(patronBusqueda.length() > 0) {
            answer = plan.search(patronBusqueda);
        }
        textResults.setText(answer);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(this, Plan15Exception.SEARCH_ERROR);
        Log.record(e);
    }
}
```

```
/**
 * Return the data of units with a prefix
 * @param prefix
 * @return
 */
public String search(String prefix){
    return data(select(prefix));
}
```

2. Realicen ingeniería reversa para la capa de dominio para buscar. Capturen los resultados de las pruebas. Deben fallar.

```

JUnit version 4.13.2
.....E..E.....
Time: 0,047
There were 2 failures:
1) shouldSearchCourseByName(domain.CoreTest)
java.lang.IndexOutOfBoundsException: Index 6 out of bounds for length 6
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:100)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
    at java.base/java.util.Objects.checkIndex(Objects.java:385)
    at java.base/java.util.ArrayList.get(ArrayList.java:427)
    at domain.Plan15.select(Plan15.java:141)
    at domain.Plan15.search(Plan15.java:176)
    at domain.CoreTest.shouldSearchCourseByName(CoreTest.java:281)
2) shouldSearchCoreByName(domain.CoreTest)
java.lang.IndexOutOfBoundsException: Index 6 out of bounds for length 6
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:100)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
    at java.base/java.util.Objects.checkIndex(Objects.java:385)
    at java.base/java.util.ArrayList.get(ArrayList.java:427)
    at domain.Plan15.select(Plan15.java:141)
    at domain.Plan15.search(Plan15.java:176)
    at domain.CoreTest.shouldSearchCoreByName(CoreTest.java:295)

FAILURES!!!
Tests run: 22, Failures: 2

```

4. ¿Cuál es el error? Soluciónenlo. Capturen los resultados de las pruebas.

El problema es que en la definición del for se colocó un <= en la definición del máximo del índice cuando se comienza a contar desde cero por lo que se trata de llegar a un índice que no existe en la lista.

```

/**
 * Consults the units that start with a prefix
 * @param
 * @return
 */
public ArrayList<Unit> select(String prefix){
    ArrayList<Unit> answers=new ArrayList<Unit>();
    prefix=prefix.toUpperCase();
    for(int i=0;i <=units.size();i++){
        if(units.get(i).code().toUpperCase().startsWith(prefix)){
            answers.add(units.get(i));
        }
    }
    return answers;
}

```

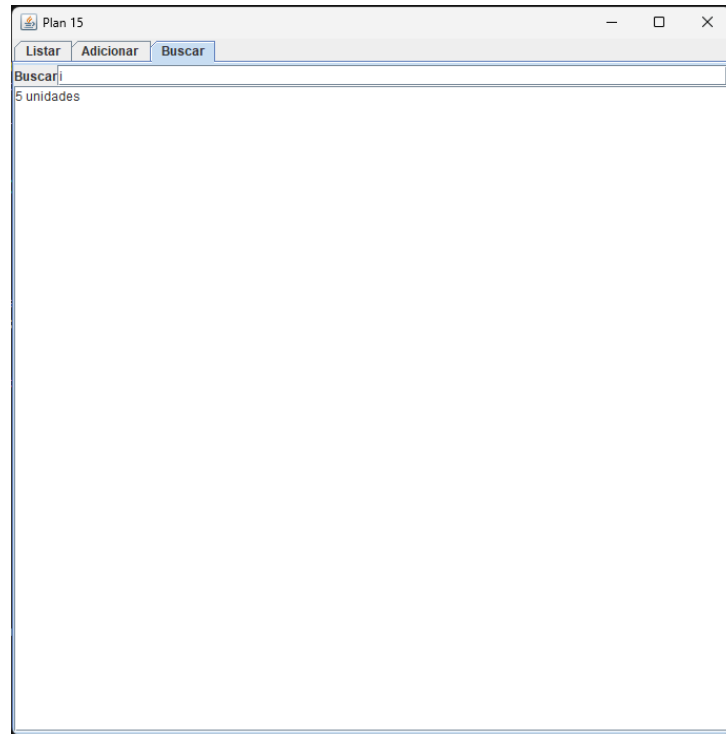
```

JUnit version 4.13.2
.....
Time: 0,048

OK (22 tests)

```

5. Ejecuten la aplicación nuevamente con el caso propuesto. ¿Qué tenemos en pantalla? ¿Qué información tiene el archivo de errores?



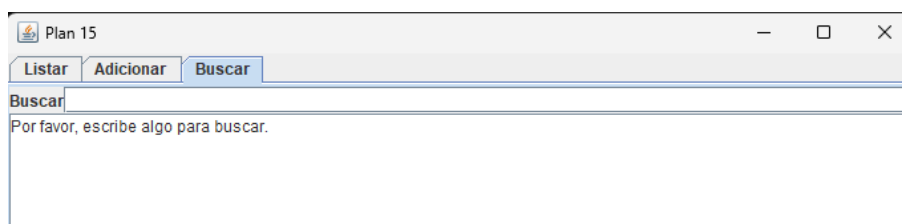
```

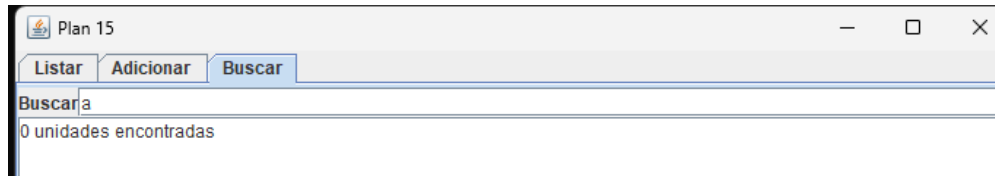
abr 05, 2025 11:57:00 A.M. domain.log record
SEVERE: java.lang.IndexOutOfBoundsException: Index 5 out of bounds for length 5
java.lang.IndexOutOfBoundsException: Index 5 out of bounds for length 5
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:100)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
    at java.base/java.util.Objects.checkIndex(Objects.java:385)
    at java.base/java.util.ArrayList.get(ArrayList.java:427)
    at domain.Plan15.select(Plan15.java:141)
    at domain.Plan15.search(Plan15.java:176)
    at presentation.Plan15GUI.actionSearch(Plan15GUI.java:246)
    at presentation.Plan15GUI.insertUpdate(Plan15GUI.java:211)
    at java.desktop/javafx.scene.control.TextInputControl.updateText(TextInputControl.java:227)
    at java.desktop/javafx.scene.control.TextInputControl.handleInsertString(AbstractDocument.java:781)
    at java.desktop/javafx.scene.control.TextInputControl.insertString(AbstractDocument.java:740)
    at java.desktop/javafx.scene.control.TextInputControl.insertString(PlainDocument.java:111)
    at java.desktop/javafx.scene.control.TextInputControl.replace(AbstractDocument.java:699)
    at java.desktop/javafx.scene.control.TextInputControl.replaceSelection(TextComponent.java:1335)
    at java.desktop/javafx.scene.control.TextInputControl.DefaultKeyTypedAction.actionPerformed(DefaultEditorKit.java:90)
    at java.desktop/javafx.scene.control.TextInputControl.notifyAction(SwingUtilities.java:1810)
    at java.desktop/javafx.scene.control.TextInputControl.processKeyEvent(Component.java:266)
    at java.desktop/javafx.scene.control.TextInputControl.processKeyEvent(Component.java:1084)
    at java.desktop/javafx.scene.control.TextInputControl.processKeyEvent(Component.java:2018)
    at java.desktop/javafx.scene.control.TextInputControl.processEvent(Component.java:6198)
    at java.desktop/javafx.scene.control.TextInputControl.processEvent(Container.java:2265)
    at java.desktop/javafx.scene.control.TextInputControl.dispatchEventImpl(Component.java:4996)
    at java.desktop/javafx.scene.control.TextInputControl.dispatchEventImpl(Container.java:2324)
    at java.desktop/javafx.scene.control.TextInputControl.dispatchEvent(Component.java:4820)
    at java.desktop/javafx.scene.control.TextInputControl.dispatchEvent(KeyboardFocusManager.java:1952)

```

5. Refactorice la funcionalidad para que sea más amable con el usuario. ¿Cuál es la propuesta? ¿Cómo la implementa?

Se ha hecho un cambio para que muestre la cantidad de unidades que coinciden con el criterio de búsqueda en vez de las totales y que diga que se debe escribir algo cuando no haya nada escrito en la barra.





```
private void actionSearch() {
    String patron = textSearch.getText().trim();

    try {
        if (patron.isEmpty()) {
            textResults.setText(t:"Por favor, escribe algo para buscar.");
            return;
        }

        String resultado = plan.search(patron);

        textResults.setText(resultado);
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(this, Plan15Exception.SEARCH_ERROR);
        Log.record(e);
    }
}
```

```
/**
 * Consult selected units
 * @param selected
 * @return
 */
public String data(ArrayList<Unit> selected){
    StringBuffer answer=new StringBuffer();
    answer.append(selected.size() + " unidades encontradas\n");
    for(Unit p : selected) {
        try{
            answer.append('>' + p.data());
            answer.append(str:"\n");
        }catch(Plan15Exception e){
            answer.append("**** " + e.getMessage());
        }
    }
    return answer.toString();
}
```

RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?

18 horas y media

2. *¿Cuál es el estado actual del laboratorio? ¿Por qué?*

Completo

3. *Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?*

Considero que la práctica más útil fue diseñar de forma simple ya que permitió entender mejor los métodos y como llevar a cabo los cambios en ellos.

4. *¿Cuál consideran fue el mayor logro? ¿Por qué?*

El mayor logro fue el entendimiento de

5. *¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?*

El mayor problema técnico fue el uso y manejo de las excepciones, para resolverlo se hizo una distinción que diferencie una excepción de la clase Plan15Exception de una general en añadir.

6. *¿Qué hicieron bien como actividades? ¿Qué se comprometen a hacer para mejorar los resultados?*

Se hizo un mejor manejo del tiempo.

7. *¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.*

8. GeeksforGeeks. (2021, 7 diciembre). *Difference Between throw and throws in Java*.

GeeksforGeeks. <https://www.geeksforgeeks.org/difference-between-throw-and-throws-in-java/>