

PROGRAMACIÓN ORIENTADA A OBJETOS

Excepciones

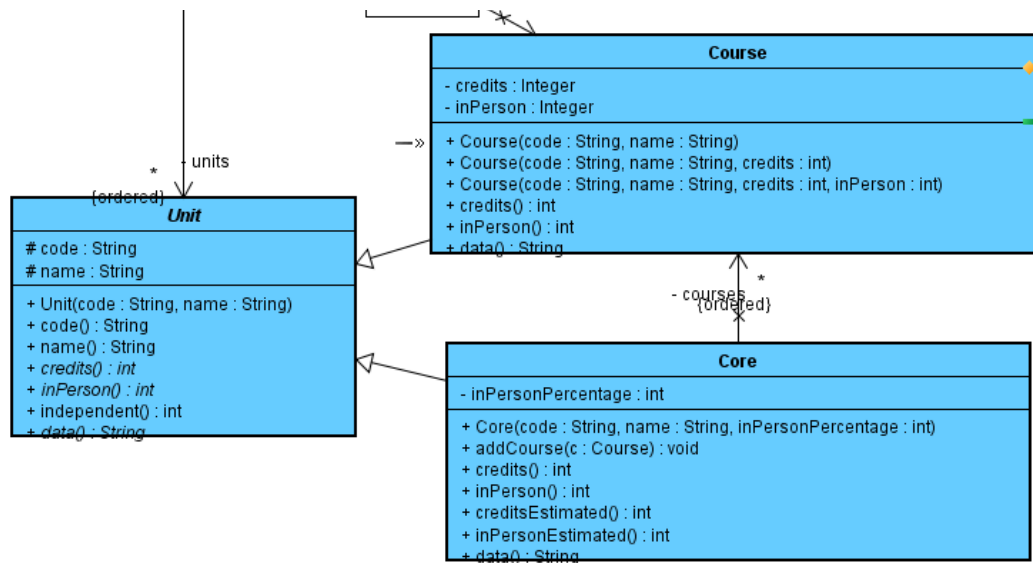
Laboratorio 4/6

Integrantes:

Daniel Useche

Daniel Patiño

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).



la seccion a revisar es la indicada en azul, para la cual se observca que:

- tenemos una clase abstracta.
- dos clases que heredan de Unit las cuales son Core, Course.
- tiene una relacion de atributo pues Core tiene un ArrayList de Course.

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

La razon por la cual el codigo no compila es debido a que no existe la clase de Plan15Exception, como se ve a continuación lanza un error de tipo ya que en ningun momento se declara la excepción.

Undeclared variable: Plan15Exception

Para solucionarlo se debe crear una clase que se llame Plan15Exception que aborde la excepción, en ella colocamos nuestras excepciones:

```
public class Plan15Exception extends Exception{

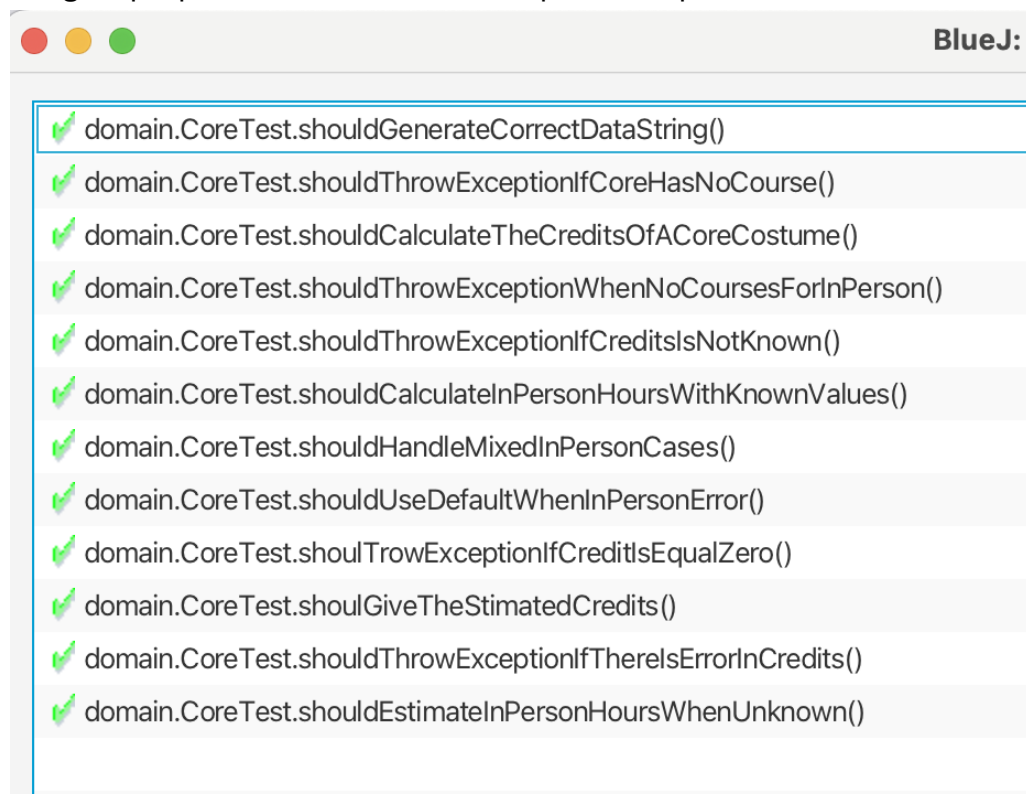
    public static final String CREDITS_UNKNOWN = "CREDITS_UNKNOWN";
    public static final String CREDITS_ERROR = "CREDITS_ERROR";
    public static final String IN_PERSON_UNKNOWN = "IN_PERSON_UNKNOWN";
    public static final String IN_PERSON_ERROR = "IN_PERSON_ERROR";
    public static final String IMPOSSIBLE = "IMPOSSIBLE ";

    public Plan15Exception(String message){
        super(message);
    }

}
```

Puntos 3,4 y 5

en Astah se encuentra el diseño, en Blue J la implementación y a continuación una imagen que prueba cada uno de los aspectos requeridos.



Plan15

EN CONSOLA

Conociendo el proyecto Plan15 [En lab04.doc]

1. En su directorio descarguen los archivos contenidos en plan15.zip, revisen el contenido.

- a. ¿Cuántos archivos se tienen?

plan15J contiene 3 archivos .java

```
Directorio de C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 04\Last Version\plan15J
05/04/2025 10:35 p. m. <DIR>      .
05/04/2025 10:35 p. m. <DIR>      ..
05/04/2025 10:34 p. m.           753 Log.java
05/04/2025 10:34 p. m.       3.392 Plan15.java
05/04/2025 10:34 p. m.       7.278 Plan15GUI.java
                3 archivos      11.423 bytes
                2 dirs  70.793.400.320 bytes libres
```

- b. ¿Cómo están organizados?

Actualmente solo están dentro del directorio plan15J, no hay ningún subdirectorio.

- c. ¿Cómo deberían estar organizados?

Deberían estar organizados bajo la siguiente estructura:

```
-plan15J
    bin
        archivos.class.
    docs
        documentacion del programa.
    src
        archivos.java (si no se encuentra empaquetado).
```

2. Estudien el diseño del programa: diagramas de paquetes y de clases.

a. ¿cuántos paquetes tenemos?

Hay dos paquetes:

-domain.

-presentation.

b. ¿cuántas clases tiene el sistema?

Hay 3 clases en total:

- Log

- Plan15

- Plan15GUI

c. ¿cómo están organizadas?

Actualmente no se encuentran bajo ningún orden, solo se encuentran dentro del directorio principal plan15J.

d. ¿cuál es la clase ejecutiva?

La clase ejecutiva es la clase plan15GUI.

3. Prepare los directorios necesarios para ejecutar el proyecto.

a. ¿qué estructura debe tener?

Deberían estar organizados bajo la siguiente estructura:

plan15J

bin

domain

Archivos .class de domain.

presentation

Archivos .class de presentation.

docs

documentacion del programa.

src

domain

-archivos.java de domain.

presentation

-archivos.java de presentation.

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 04\Last Version\plan15J>tree
Listado de rutas de carpetas para el volumen Windows
El número de serie del volumen es A2DB-F999
C:..
|-- bin
|   |-- domain
|   |-- presentation
|-- docs
|-- lib
|-- src
|   |-- domain
|   |-- presentation
```

b. ¿qué clases deben tener?

Hacen falta las siguientes clases:

- Unit
- Course
- Core
- Plan15Exception
- CoreTest

c. ¿dónde están esas clases?

Esas clases están en el directorio Units hecho anteriormente.

d. ¿qué instrucciones debe dar para ejecutarlo?

Para ejecutarlo primero se tienen que copiar los archivos .java de Units faltantes para que no genere error y los colocamos en el paquete de domain, luego, nos dirigimos al directorio plan15J y se usa el siguiente comando:

```
javac -d bin -sourcepath src src/presentation/*.java
```

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 04\Last Version\plan15J>javac -d bin -sourcepath src src/presentation/*.java
```

Ejecutado este comando, se compilará el código generando los archivos .class y la documentación, seguido a esto usamos el siguiente comando para ejecutarlo (Estando ubicados desde el directorio raíz):

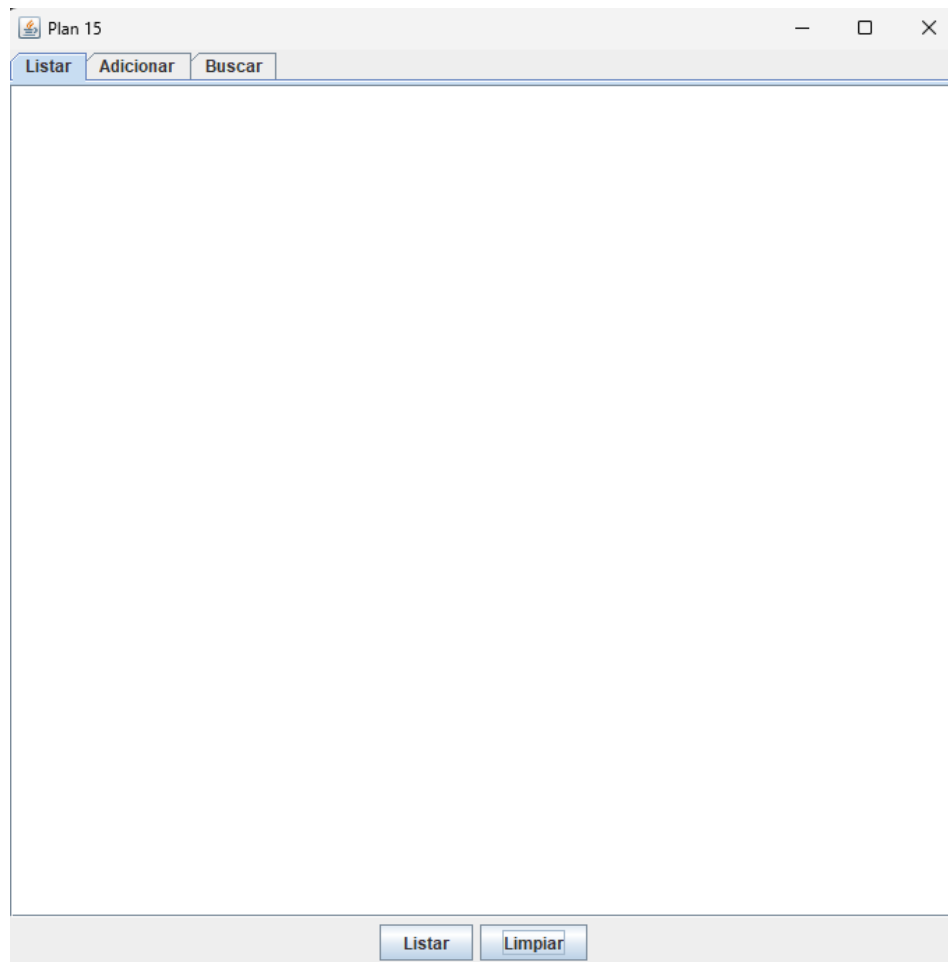
```
java -cp bin presentation.Plan15GUI
```

4. Ejecute el proyecto:

a. ¿qué funcionalidades ofrece?

Al ejecutarlo sale una ventana emergente donde en la parte superior se encuentra una barra de búsqueda donde podemos elegir entre listar, adicionar y buscar.

Si nos posicionamos en Listar, encontraremos dos botones en la parte inferior los cuales son listar y limpiar.



Si nos posicionamos en Adicionar, encontramos un listado donde se pueden adicionar materias, colocando sigla, nombre, creditos o porcentaje, horas

presenciales y cursos. De igual manera encontramos en la parte inferior los botones de listar y limpiar.



Plan 15

Listar Adicionar Buscar

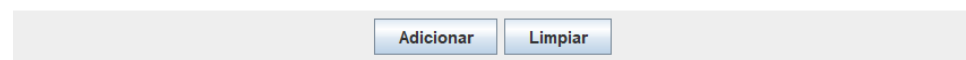
Sigla

Nombre

Creditos o porcentaje

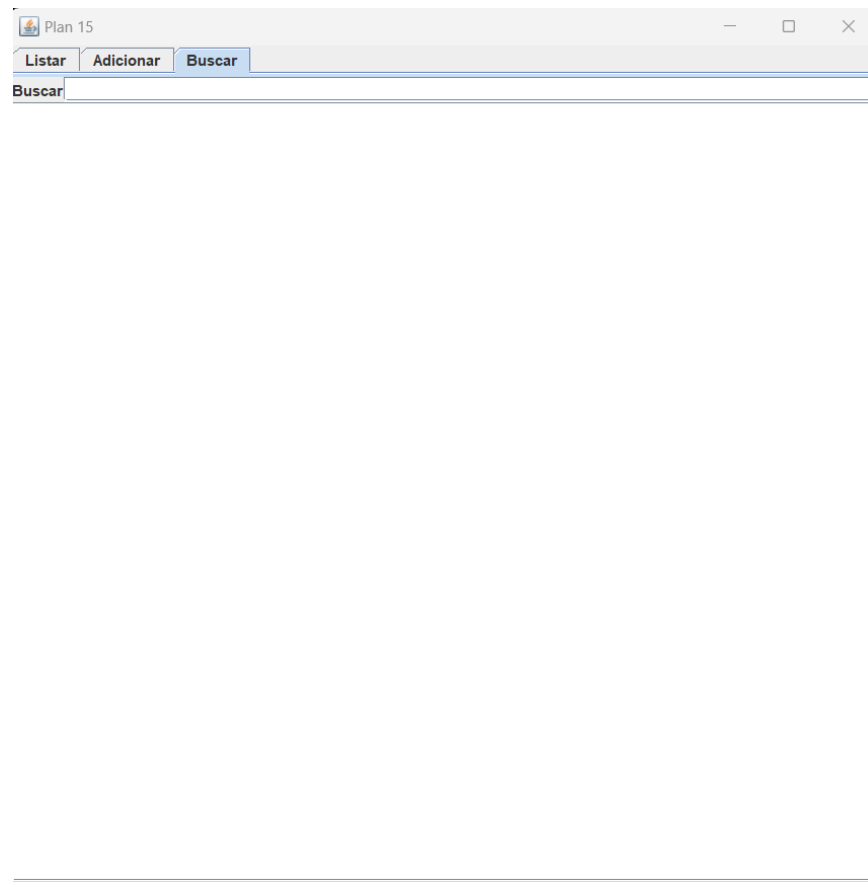
Horas presenciales (solo para cursos)

Cursos (solo para nucleos)



Adicionar Limpiar

Por último si nos posicionamos en Buscar, encontraremos una barra de búsqueda donde podremos introducir texto.



b. ¿cuáles funcionan?

La barra superior los tres botones que hay funcionan para navegar en programa. Si nos posicionamos en Listar, Los dos botones inferiores funcionan para mostrar una lista, o ocultarla.

Plan 15

Listar

Adicionar

Buscar

4 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]

Listar

Limpiar

Plan 15

Listar

Adicionar

Buscar

Listar

Limpiar

Si nos desplazamos a Adicionar, podemos asignar un nuevo curso o nucleo donde se pueden llenar los múltiples espacios, tanto Adicionar como limpiar funcionan correctamente.

Plan 15

Listar Adicionar Buscar

Sigla
FCFI

Nombre
fundamentos

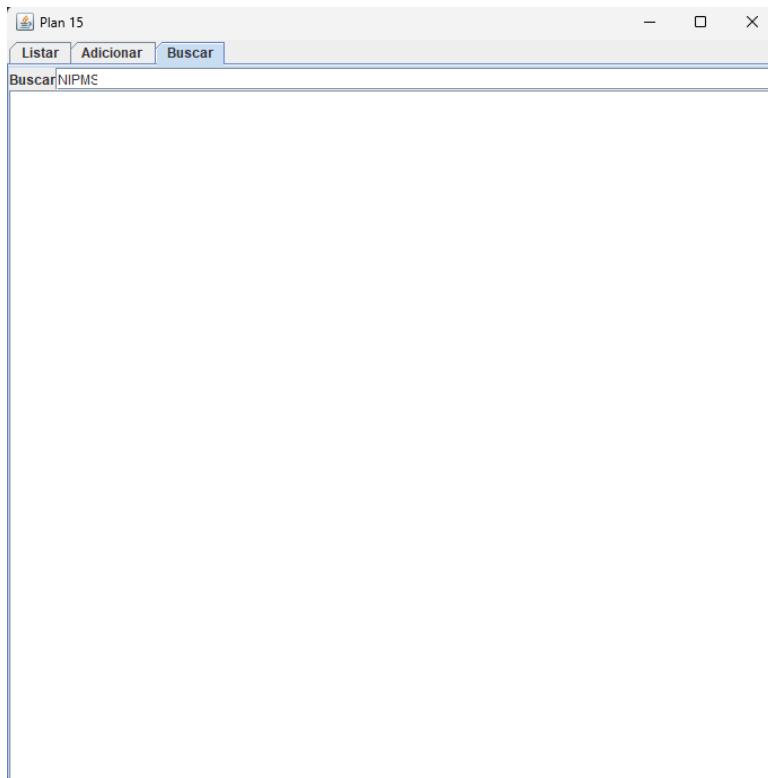
Creditos o porcentaje
60

Horas presenciales (solo para cursos)

Cursos (solo para nucleos)
MABA

Adicionar Limpiar

Sin embargo, si nos vamos a Buscar, no deja hacer la búsqueda correctamente por lo que consideramos que no funciona.



5. Revisen el código y la documentación del proyecto.

a. ¿De dónde salen las unidades iniciales?

Las unidades iniciales salen del siguiente método:

```
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]);
    }
}
```

Donde se adicionan los cursos iniciales y el núcleo al que pertenecen.

b. ¿Qué clase pide que se adicionen?

La misma clase al crear el constructor, llama al método de addSome donde se adicionan las unidades iniciales.

c. ¿Qué clase los adiciona?

Plan 15 los adiciona en los contenedores units y courses.

Adicionar y listar. Todo OK.

El objetivo es realizar ingeniería reversa a las funciones de adicionar y listar.

1. Adicionen un nuevo curso y un nuevo núcleo

Curso

DOSW Desarrollo y Operaciones Software 4 y 4

Core

NFPE Núcleo de formación específica 100

DOSW

a. ¿Qué ocurre?

Para adicionar los cursos primero ejecutamos el programa y nos vamos a la sección de adicionar:

Plan 15

Listar Adicionar Buscar

Sigla

Nombre

Creditos o porcentaje

Horas presenciales (solo para cursos)

Cursos (solo para nucleos)

Adicionar Limpiar

A continuación, añadimos el nuevo curso y núcleo solicitado.

Plan 15

Listar Adicionar Buscar

Sigla
DOSW

Nombre
Desarrollo y Operaciones Software

Creditos o porcentaje
4

Horas presenciales (solo para cursos)
4

Cursos (solo para nucleos)

Adicionar Limpiar

Le damos al botón de adicionar y luego le damos en limpiar para ingresar los datos del núcleo.

Plan 15

Listar Adicionar Buscar

Sigla
INFPE

Nombre
Núcleo de formación específica

Creditos o porcentaje
100

Horas presenciales (solo para cursos)

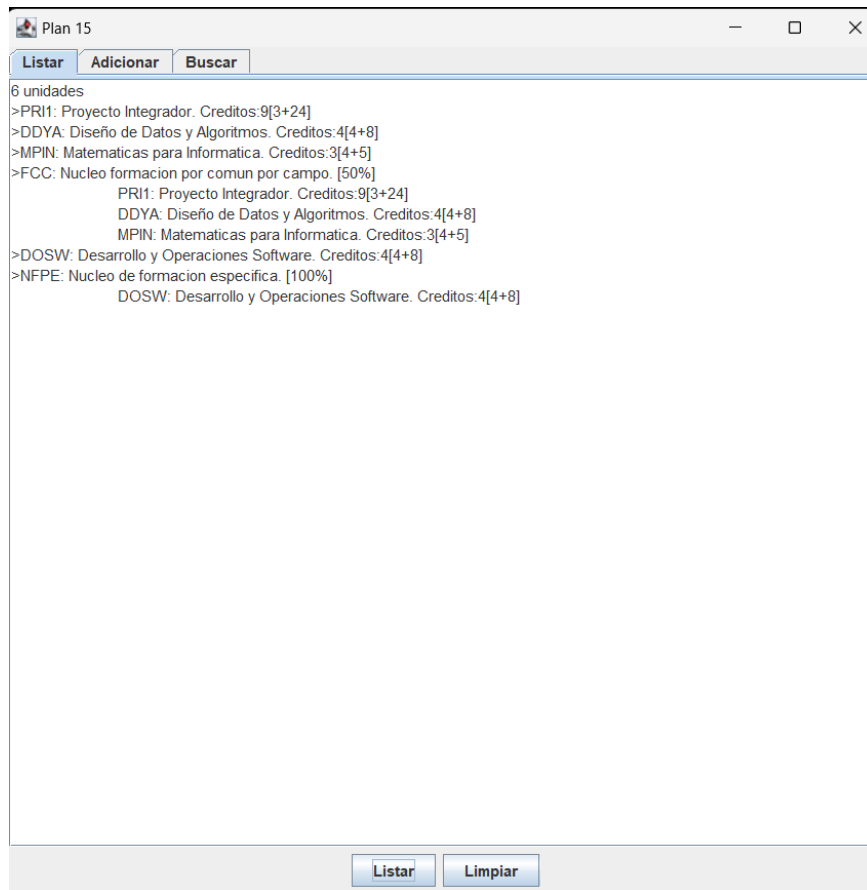
Cursos (solo para nucleos)
DOSW

Adicionar Limpiar

Finalmente le damos al boton de Adicionar.

b. ¿Cómo lo comprueban? Capturen la pantalla.

Para comprobarlo, nos vamos a la sección de Listar y presionamos el botón de listar:



c. ¿Es adecuado este comportamiento?

Es adecuado, sin embargo, se podría mejorar con algún mensaje de confirmación de que el núcleo o el curso ha sido adicionado de manera exitosa o si ya fue añadido que de un mensaje de que el curso ya fue añadido y no deje volverlo a agregar a la lista o también si hay algún error o falta algún dato que envíe un mensaje y no un error en la consola.

2. Revisen el código asociado a adicionar en la capa de presentación y la capa de dominio.

a. ¿Qué método es responsable en la capa de presentación?

El método responsable en la capa de presentación es el `actionAdd()`, ya que este es el encargado de verificar si el campo de basics este vacío entonces añade el curso, de no estarlo, procedería a añadir un núcleo.

b. ¿Qué método en la capa de dominio?

Los métodos asociados a adicionar son `addCore` y `addCourse` de la clase `Plan15` en `domain`, ya que estos métodos son los que se utilizan para adicionar si se desea un curso o un núcleo.

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

```
✓ Tests.CoreTest.shouldNotAddCourseWithInvalidValues()
✓ Tests.CoreTest.shouldThrowExceptionIfThereIsErrorInCredits()
✓ Tests.CoreTest.shouldNotAddCoreWithNonExistingCourse()
✓ Tests.CoreTest.shouldNotAddCourseWithInvalidName()
✓ Tests.CoreTest.shouldAddCourseWithValidValues()
```

4. Revisen el código asociado a listar en la capa de presentación y la capa de dominio.

a. ¿Qué método es responsable en la capa de presentación?

El método responsable en la capa de presentación es el `actionList()`.

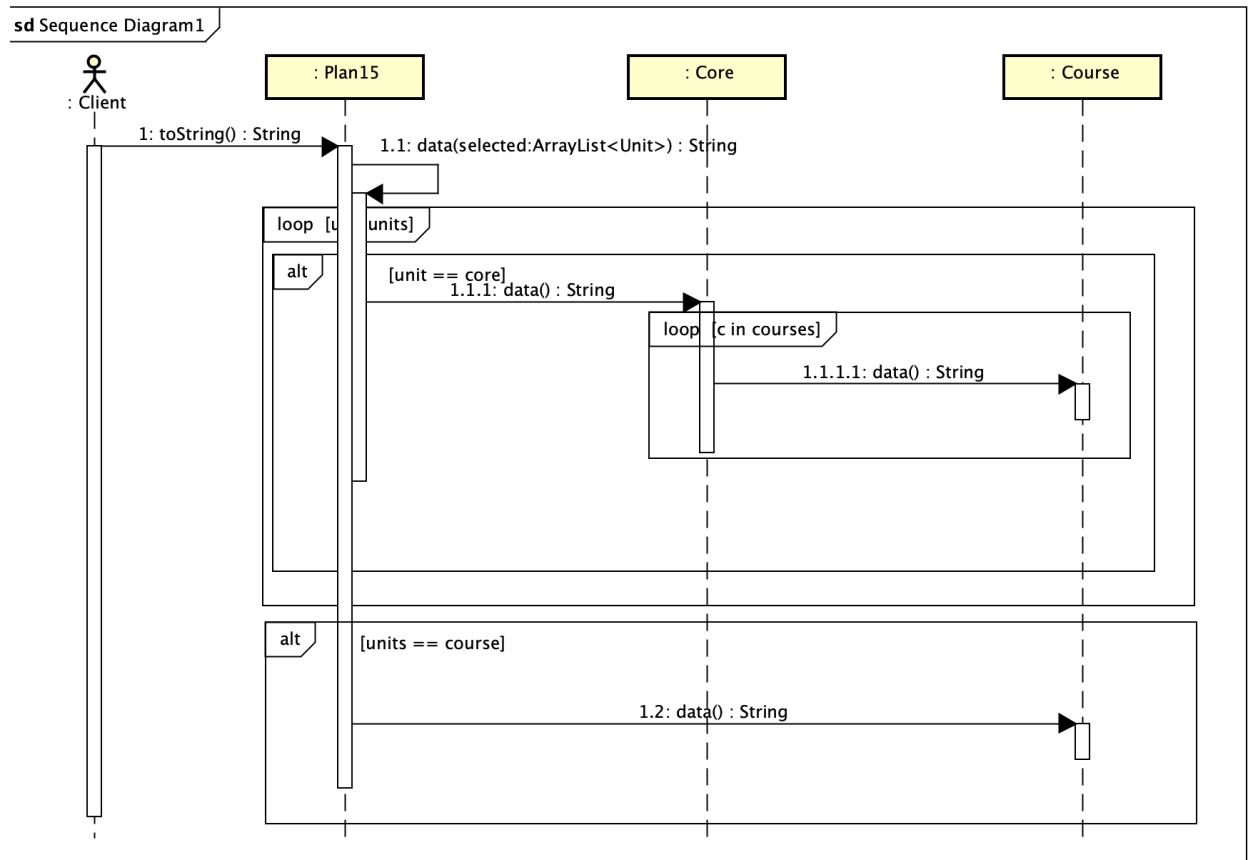
El cual llama a `toString`

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

b. ¿Qué método en la capa de dominio?

El método responsable en la capa de dominio es `data`.

5. Realicen ingeniería reversa para la capa de dominio para listar. Capturen los resultados de las pruebas de unidad.



6. Propongan y ejecuten una prueba de aceptación.

✓ Tests.Plan15Test.shouldFailNameDoesNotExist()

✓ Tests.Plan15Test.testToStringWithCourses()

**Adicionar una unidad. Funcionalidad robusto [En lab04.doc, Plan15.astay *.java]
(NO OLVIDEN BDD – MDD)**

El objetivo es perfeccionar la funcionalidad de adicionar un curso para hacerla más robusta.

Para cada uno de los siguientes casos realice los pasos del 1 al 4.

a. ¿Y si el nombre de la unidad no existe?

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

```
@Test
public void addCourseNameDoesNotExist(){
    try{
        Plan15 plan = new Plan15();
        plan.addCourse("MBDA", null, "4", "4");
    } catch(Plan15Exception e){
        assertEquals(Plan15Exception.INVALID_NAME, e.getMessage());
    }
}
```

2. Analicen el diseño realizado. Para hacer el software robusto:

¿Qué método debería lanzar la excepción?

El método que debería lanzar la excepción es addCourses ya que es el que se encarga de adicionar los cursos.

¿Qué métodos deberían propagarla?

Los métodos que deben propagarla son aquellos métodos que llamen a addCourses hasta llegar a la que la atiende.

¿Qué método debería atenderla?

El método que debería atenderla es addAction() ya que es el que finalmente mostrara los cursos.

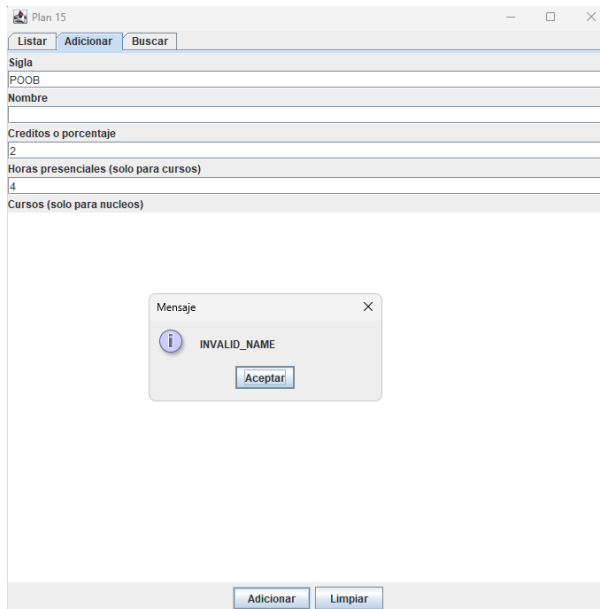
3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

```
/**
 * Add a new course
 */
public void addCourse(String code, String name, String credits, String inPerson) throws Plan15Exception{
    if(name == null || name.trim().isEmpty()){
        throw new Plan15Exception(Plan15Exception.INVALID_NAME);
    }

    Course nc=new Course(code,name,Integer.parseInt(credits),Integer.parseInt(inPerson));
    units.add(nc);
    courses.put(code.toUpperCase(),nc);
}

Tests.Plan15Test.shouldFailNameDoesNotExist()
```

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



Se ingresa un nuevo curso sin nombre y luego se da en adicionar, como se ve en la pantalla la excepción se lanzó correctamente.

b. ¿Y si los valores enteros no son enteros?

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

```
@Test
public void shouldFailNumberNotInteger(){
    try{
        Plan15 plan = new Plan15();
        plan.addCourse("POOB", "Programacion Orientada a Objetos", "4.5", "4");
    }catch(Plan15Exception e){
        assertEquals(Plan15Exception.CREDITS_ERROR, e.getMessage());
    }
}
```

2. Analicen el diseño realizado. Para hacer el software robusto:

¿Qué método debería lanzar la excepción?

El método que debe lanzar la excepción es el método addCourse(), ya que es el método encargado en agregar los cursos

¿Qué métodos deberían propagarla?

Los metodos que deben propagarla son todos aquellos que llaman al metodo addCourse hasta llegar al método que la atiende

¿Qué método debería atenderla?

El método que atiende la excepción es actionAdd() ya que es el que finalmente agregaría los cursos al GUI.

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

```
/**
 * Add a new course
 */
public void addCourse(String code, String name, String credits, String inPerson) throws Plan15Exception{
    int parsedCredits;
    int parsedInPerson;
    if(name == null || name.trim().isEmpty()){
        throw new Plan15Exception(Plan15Exception.INVALID_NAME);
    }

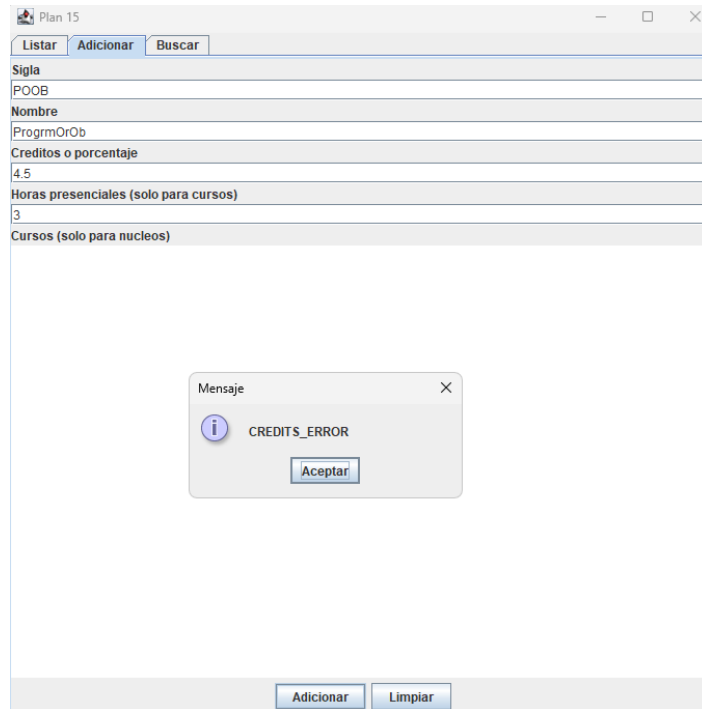
    try{
        parsedCredits = Integer.parseInt(credits);
        parsedInPerson = Integer.parseInt(inPerson);
    } catch(NumberFormatException e){
        throw new Plan15Exception(Plan15Exception.CREDITS_ERROR);
    }

    Course nc=new Course(code,name,Integer.parseInt(credits),Integer.parseInt(inPerson));
    units.add(nc);
    courses.put(code.toUpperCase(),nc);
}
```

Tests.Plan15Test.shouldFailNumberNotInteger()

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

Cuando se ingresa algun valor que no es entero, se lanza la excepcion y lo que ve el usuario es un mensaje avisando que la cantidad ingresada es invalida.



c. ¿Y si el portage no está entre 0 y 100?

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

```
@Test
public void shouldFailIfOutOfRange(){
    try{
        Plan15 plan = new Plan15();
        plan.addCore("CBI", "Ciclo Basico Inicial", "-50", "POOB");
    }catch(Plan15Exception e){
        assertEquals(Plan15Exception.PORTAGE_ERROR, e.getMessage());
    }
}
```

2. Analicen el diseño realizado. Para hacer el software robusto:

¿Qué método debería lanzar la excepción?

El método que debe lanzar la excepción es el método `addCore()`, ya que es el método encargado en agregar los nucleos.

¿Qué métodos deberían propagarla?

Los métodos que deben propagarla son todos aquellos que llaman al metodo `addCore` hasta llegar al método que la atiende

¿Qué método debería atenderla?

El método que atiende la excepción es `addActionAdd()` ya que es el que finalmente agregaría los cursos al GUI.

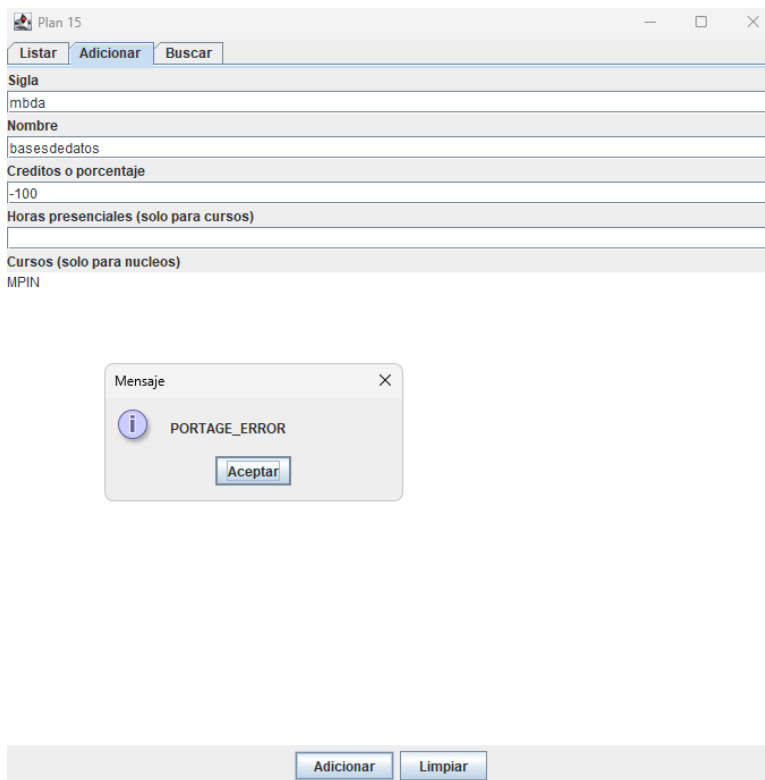
3. Construya la solución propuesta. Capture los resultados de las pruebas de unidad.

```
/**
 * Add a new core
 */
public void addActionAdd(String code, String name, String percentage, String theCourses) throws Plan15Exception{
    int parsedPercentage;
    try{
        parsedPercentage = Integer.parseInt(percentaje);
        if(parsedPercentage < 0 || parsedPercentage > 100){
            throw new Plan15Exception(Plan15Exception.PORTAGE_ERROR);
        }
        Core c = new Core(code,name,Integer.parseInt(percentaje));
        String [] aCourses= theCourses.split("\n");
        for (String b : aCourses){
            c.addActionAdd(courses.get(b.toUpperCase()));
        }
        units.addActionAdd(c);
    }catch(NumberFormatException e){
        throw new Plan15Exception(Plan15Exception.PORTAGE_ERROR);
    }
}
```

Tests.Plan15Test.shouldFailIfOutOfRange()

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

Se intenta ingresar un porcentaje menor a cero y lanza el siguiente error



d. Y si el código es nulo?

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

```
@Test
public void shouldFailCodeDoesNotExist(){
    try{
        Plan15 plan = new Plan15();
        plan.addCourse(null, "Algoritmos y programacion", "2", "14");
    } catch (Plan15Exception e){
        assertEquals(Plan15Exception.INVALID_CODE, e.getMessage());
    }
}
```

2. Analicen el diseño realizado. Para hacer el software robusto:

¿Qué método debería lanzar la excepción?

El método que debe lanzar la excepción es el método `addCourse()`, ya que es el método encargado en agregar los cursos

¿Qué métodos deberían propagarla?

Los métodos que deben propagarla son aquellos métodos que llamen a addCourses hasta llegar a la que la atiende.

¿Qué método debería atenderla?

El método que atiende la excepción es addAction() ya que es el que finalmente agregaría los cursos al GUI.

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

```
/**
 * Add a new course
 */
public void addCourse(String code, String name, String credits, String inPerson) throws Plan15Exception{
    int parsedCredits;
    int parsedInPerson;
    if(code == null || code.trim().isEmpty()){
        throw new Plan15Exception(Plan15Exception.INVALID_CODE);
    }

    if(name == null || name.trim().isEmpty()){
        throw new Plan15Exception(Plan15Exception.INVALID_NAME);
    }

    try{
        parsedCredits = Integer.parseInt(credits);
        parsedInPerson = Integer.parseInt(inPerson);
    } catch(NumberFormatException e){
        throw new Plan15Exception(Plan15Exception.CREDITS_ERROR);
    }

    Course nc=new Course(code,name,Integer.parseInt(credits),Integer.parseInt(inPerson));
    units.add(nc);
    courses.put(code.toUpperCase(),nc);
}
```

✓ Tests.Plan15Test.shouldFailCodeDoesNotExist()

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

Lo que se hace en la aplicación, es que no se pone el código del curso por lo que al querer Adicionarlo lanza un error de código inválido.

Plan 15

Listar Adicionar Buscar

Sigla

Nombre

Teoria de Conjuntos

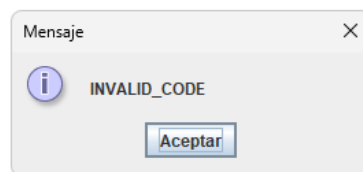
Creditos o porcentaje

4

Horas presenciales (solo para cursos)

12

Cursos (solo para nucleos)

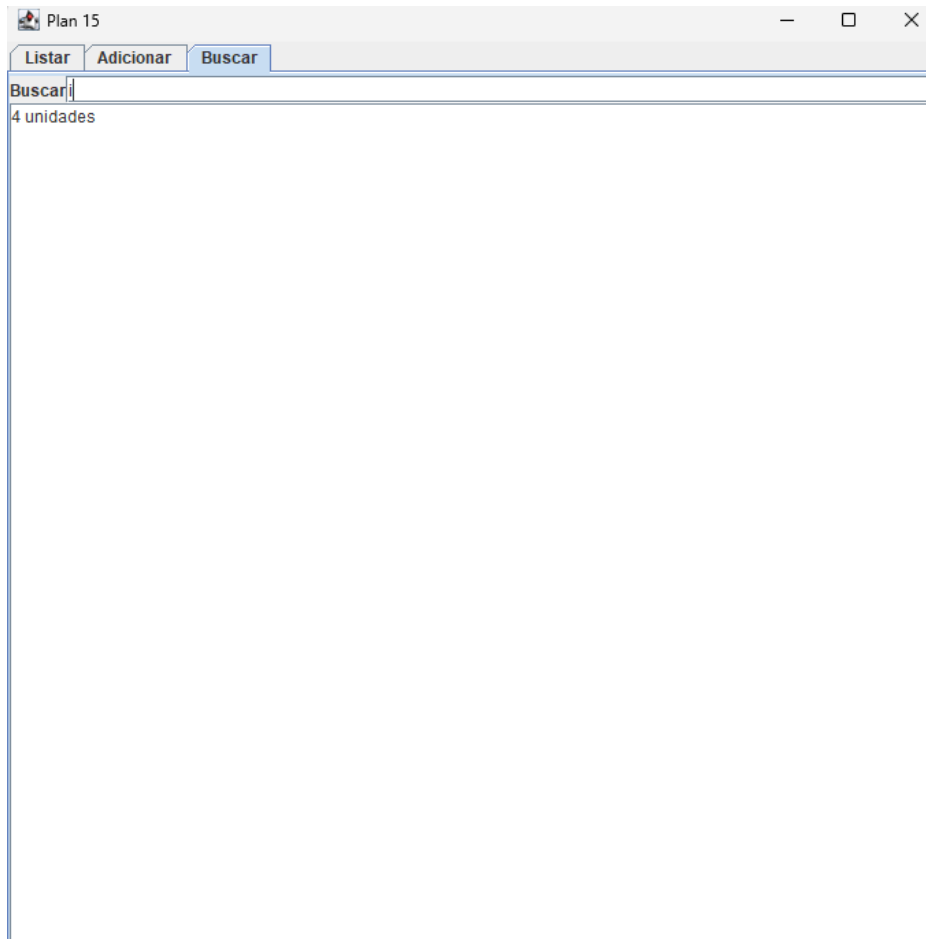


Adicionar Limpiar

Consultando por patrones. ¡ No funciona y queda sin funcionar!
[En Plan15.asta, Plan15.log, lab04.java y *.java]
(NO OLVIDEN BDD - MDD)

1. Consulten una unidad que inicie con I.
¿Qué sucede?

En este caso ya no aparece nada, el unico mensaje que salio en pantalla fue un no result que dejamos, la aplicación se sigue ejecutando normalmente



5. ¿Es adecuado que la aplicación continúe su ejecución después de sufrir un incidente como este? ¿de qué dependería continuar o parar?

Todo depende de la gravedad del error, si es un error el cual no afecte en alto grado la funcionalidad del programa, se puede continuar, de lo contrario si es un error fatal, el programa se tiene que detener ya que podria afectar la experiencia del usuario. En este caso si es adecuado que continúe ya que no afecta en alto grado la funcionalidad del del programa.

6. Modifiquen la aplicación para garantizar que SIEMPRE que haya un error se maneje de forma adecuada. ¿Cuál fue la solución implementada?

Con los try-catch se manejan los problemas de manera adecuada, cada que haya un error de sintaxis o que el usuario no este colocando bien algo, se lanza una advertencia de error, sin embargo, tambien se utilizan algunas funciones como el Log.record(mensaje) el cual cada que se ejecuta el programa nos ayuda a guardar los errores y a solucionarlos lo antes posible.

Consultando por patrones. ¡Ahora si funciona!

[En Plan15.asta, Plan15.log, lab04.java y *.java]

(NO OLVIDEN BDD - MDD)

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?


El metodo responsable de buscar en la capa de presentacion es actionSearch()

¿Qué método es responsable en la capa de dominio?

El metodo responsable de buscar en la capa de dominio es search


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

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

 Tests.Plan15Test.shouldFailSearch()

El error es un menor o igual que se encuentra en el metodo select de la clase plan15, al decir que es menor o igual se sale del rango, por lo que la solucion es solo manejarlo como menor que.

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

 Tests.Plan15Test.shouldFailSearch()

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

¿Cómo la implementa?

La propuesta es usar JOptionPane, la cual ya fue implementada anteriormente, es una advertencia amable con el usuario y no se muestra un error que puede llegar a ser confuso para el usuario. En las imagenes anteriores se puede evidenciar el uso del JOptionPane en el programa.

RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?
(Horas/Hombre)

El tiempo invertido fue de aproximadamente 20 horas por cada uno de nosotros.

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

El laboratorio se encuentra parcialmente completado, ya que hicieron falta algunas pocas cosas que no pudimos resolver.

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

Una de las más útiles fue la de bdd ya que nos ayudó a tener mayor claridad sobre el producto que se quería entregar, y a tener mayor claridad sobre los objetivos en cada aspecto del proyecto y pues tambien a reconocer y evitar algunos errores.

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

El mayor logro fue haber encontrado el error del indice ya que no sabíamos con claridad donde se encontraba el error, lo que nos tomo bastante tiempo encontrarlo.

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

El mayor problema técnico fue el manejo de otro editor que no fuera bluej ya que tuvimos bastante conflicto con la importacion de las pruebas entre otros problemas, nuestra solución fue abrir el programa con bluej.

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

Como equipo como siempre nos destacamos en trabajar siempre juntos con una carga de trabajo distribuida apoyándonos el uno al otro, nos comprometemos a mejorar la calidad de entrega de nuestros trabajos y a seguir mejorando.

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

Solo usamos una referencia la cual fue chat gpt, esta nos ayudó con algunos errores de sintaxis y a pensar diversas soluciones frente a algunos problemas.

OpenAI. (2024). *ChatGPT (versión GPT-4) [Modelo de lenguaje]*. <https://chat.openai.com/>