Object-Oriented Programming Project

Angie Nathaly Cisneros Guerrero	20242020084
Luis Sebastián Correa Barreto	20242020085

Project: This project aims to develop a functional replica of a digital news platform, taking as reference the interface and functionalities of the newspaper "El Espectador", including the presentation of headlines, summaries, access to full articles and navigation through different categories of information.

In addition, it seeks to simulate the user experience of a modern and intuitive news portal, providing access to updated journalistic content organized in various thematic sections.

1. Workshop

1.1 Conceptual Design Updates: (updated , more detailed descriptions)

- **Objetives:**

★ **Develop a system that manages news content efficiently**, implementing a design capable of storing, organizing and retrieving news, including essential attributes such as title, content, date, author, and special markers such as "Featured new" or international classification to ensure a structured design.

★ **Create an intuitive and navigable user interface**, designing and building components that allow users to access and explore the news easily. This includes the creation of classes such as Homepage, displaying news highlights and specialized sections, as well as detailed visualization of individual articles, focusing on accessibility for the end user.

★ **Generate concise summaries and access to detailed reports**, integrating the method to obtain brief summaries of the news, facilitating a quick overview of the content. In addition, ensure the complete presentation of articles, allowing users to delve deeper into the topics of their interest, providing clear information on the depth of content and content.

★ **Implement news classification and filtering mechanisms,** developing a logic that allows categorizing news by type such as international, or latest news and providing filtering sections so that users can find specific content efficiently, improving the search experience and content organization.

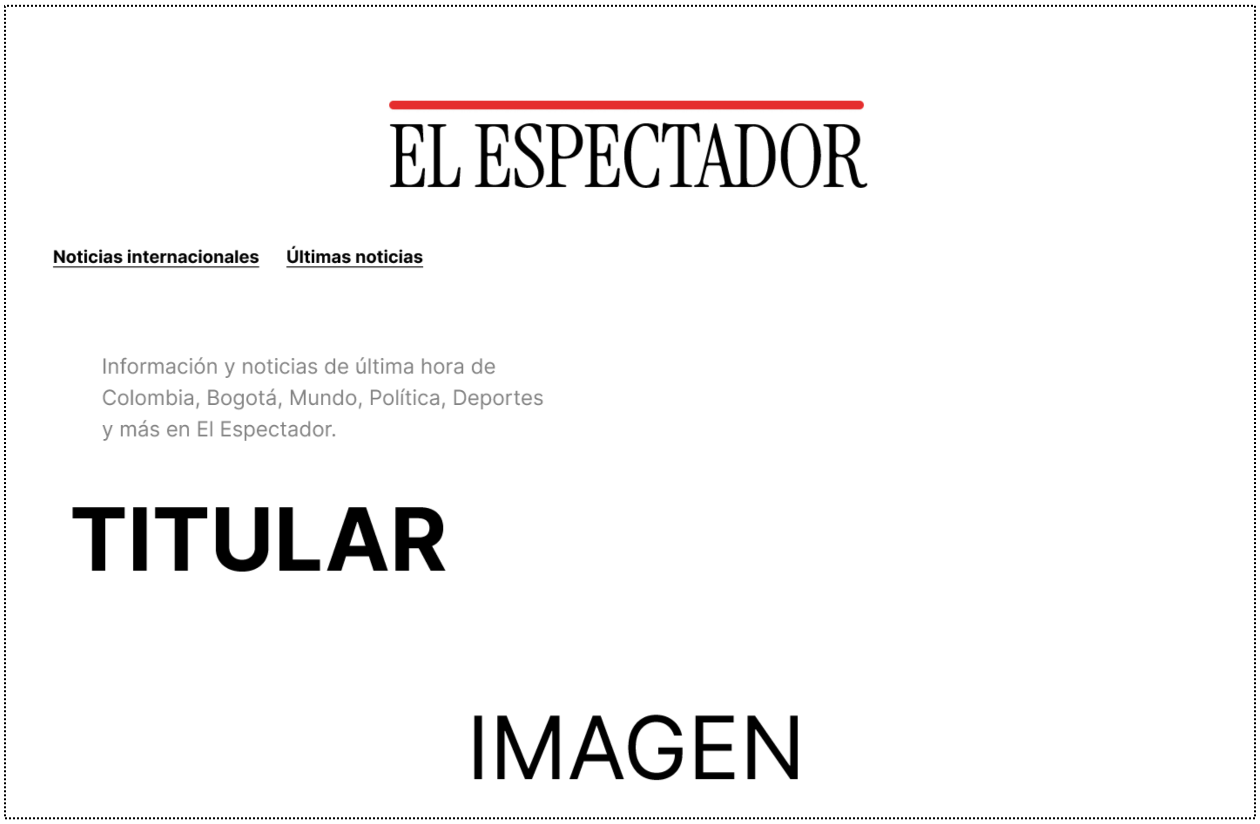
- **Functional requirements:**
  - Display the main news on the Home Page, showing a specific selection of featured news on the Homepage. This section will be constantly updated to show the most relevant and prioritized news.
  - Display the "Featured new" by accessing a specific section that presents the news marked as "featured new".
  - Navigate through the different news sections, accessing different news categories through a navigation menu.
  - Display the Latest Published News, showing a list of the 9 most recent news items in a dedicated "Latest News" section. The news will be sorted by descending publication date, displaying the most recent news first.
  - Access the Full Detail of a News Article by clicking on any news headline to access a detail page showing the full article.
- **Non-Functional requirements:**
  - Scalability for increased news content, the system must be able to handle a large increase in the volume of news stored and accessed without significant performance degradation. the architecture must allow for the addition of concurrent users without increasing the average response time.
  - Intuitive and accessible navigation (for demographics), The user interface should be intuitive, allowing users with different levels of digital skills to find and access the news efficiently, with the least number of clicks for the main functionalities (view full story, navigate to section).

- Modular and maintainable codebase, the code must be modularized, following SOLID design principles, to facilitate the addition of new functionality and modification of existing functionality with minimal impact on other parts of the system.

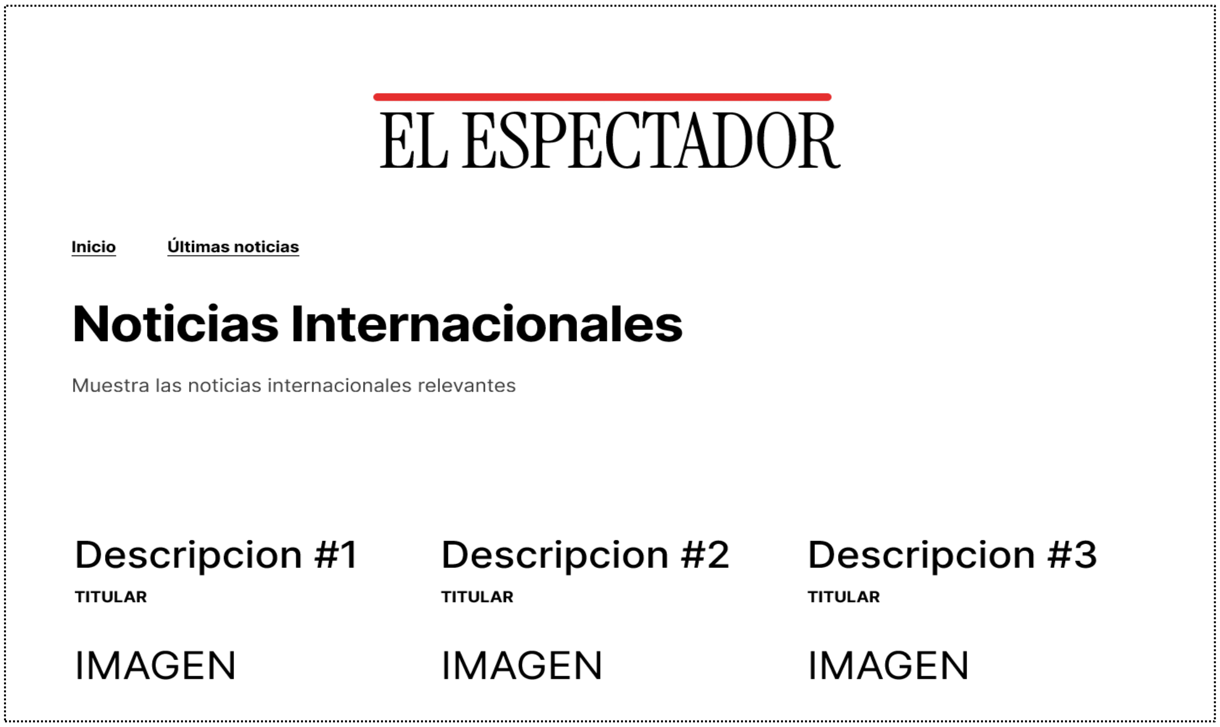
- Adequate documentation of code and architecture, generating complete and updated technical documentation of the source code using comments for all classes and methods, creating a UML class diagram representing the main architecture of the system, and descriptions of design decisions.

- **Mockup: (updated , images)**

Homepage:


EL ESPECTADOR
Noticias internacionales
Últimas noticias
Información y noticias de última hora de Colombia, Bogotá, Mundo, Política, Deportes y más en El Espectador.
TITULAR
IMAGEN

International News:



EL ESPECTADOR

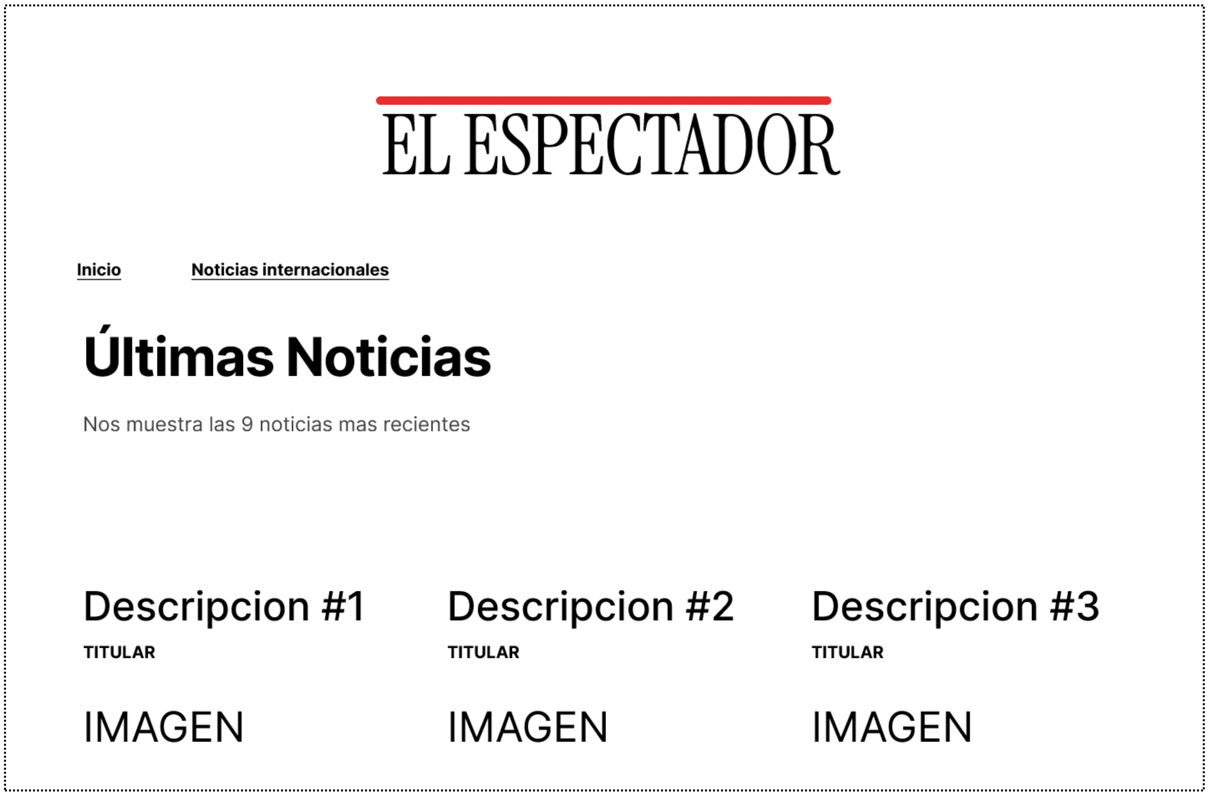
Inicio Últimas noticias

Noticias Internacionales

Muestra las noticias internacionales relevantes

Descripcion #1	Descripcion #2	Descripcion #3
TITULAR	TITULAR	TITULAR
IMAGEN	IMAGEN	IMAGEN

- **Latest News:**



EL ESPECTADOR

Inicio Noticias internacionales

Últimas Noticias

Nos muestra las 9 noticias mas recientes

Descripcion #1	Descripcion #2	Descripcion #3
TITULAR	TITULAR	TITULAR
IMAGEN	IMAGEN	IMAGEN

- **User stories:**

Title: Home Page Reader	**Priority:** High	**Estimate:** 6 weeks
User Story: ***<u>As a visitor</u>***, I want to see the main and top news clearly on the homepage so I can quickly get informed about the most relevant events.		
Acceptance Criteria: -Main news article displayed prominently. - Section for "Top News" visible under the main article.		
Title: Latest News Reader	**Priority:** Medium	**Estimate:** 4 weeks
User Story: ***<u>-As a</u>*** frequent visitor, I want to view the latest news quickly to stay updated.		
Acceptance Criteria: - “Latest News” section that dislays the 9 most recently published articles. - Articles ordered form newest to oldest.		

Title: International News	**Priority:** Low	**Estimate:** 4 weeks
User Story: ***As an*** international affairs enthusiast, I want a dedicated section for international news so I can stay informed globally.		
Acceptance Criteria: - International" section easily accessible. - Displays at least 9 articles from different countries.		

Title: News	**Priority:** High	**Estimate:** 1 week
User Story: ***As an*** app developer I want to create news whit title, content and date so the readers can see the info sorted and updated.		
Acceptance Criteria: - "News" this is the most important thing in the project because it's the base for all sections.		

Title: Different types of sections	**Priority:** Medium	**Estimate:** 4 weeks
User Story: ***-As a*** frequent visitor, I want to view a section of latest news and another I don't mind whatever it is.		
Acceptance Criteria: -This is acceptable and good because we can create a single class called "Section" to spare different kind of news		

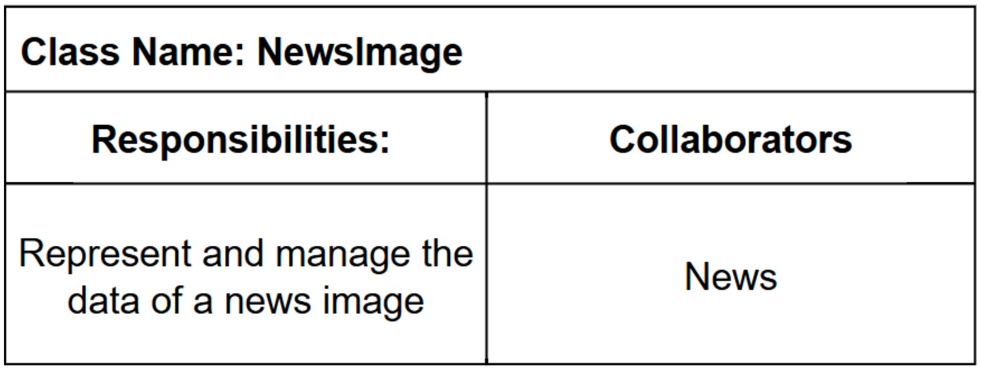
- **CRC Cards: (updated , format)**

Class Name: Homepage	
Responsibilities:	**Collaborators**
Display last news	-News -Section

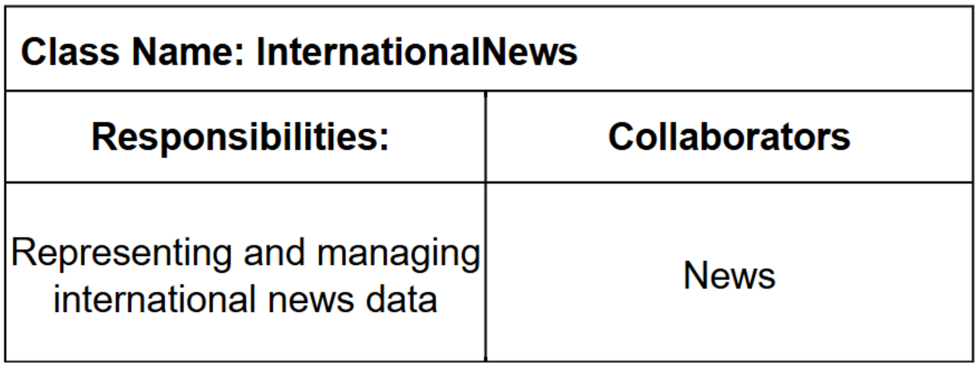
Class Name: Section	
Responsibilities:	**Collaborators**
Group news by category (Latest news, international news)	News

Class Name: InternationalNewsSection	
Responsibilities:	**Collaborators**
Manage and filter specifically international news	News

Class Name: New	
Responsibilities:	**Collaborators**
Manage article attributes (Title, content, date, etc.)	Section



Class Name: NewsImage	
Responsibilities:	**Collaborators**
Represent and manage the data of a news image	News



Class Name: InternationalNews	
Responsibilities:	**Collaborators**
Representing and managing international news data	News

1.2 Documentation of changes respect to workshop-1:

- Decisions emerged from further planning and discussions.

Workshop 1 → Workshop 2: Evolution of UML & OOP Implementation:

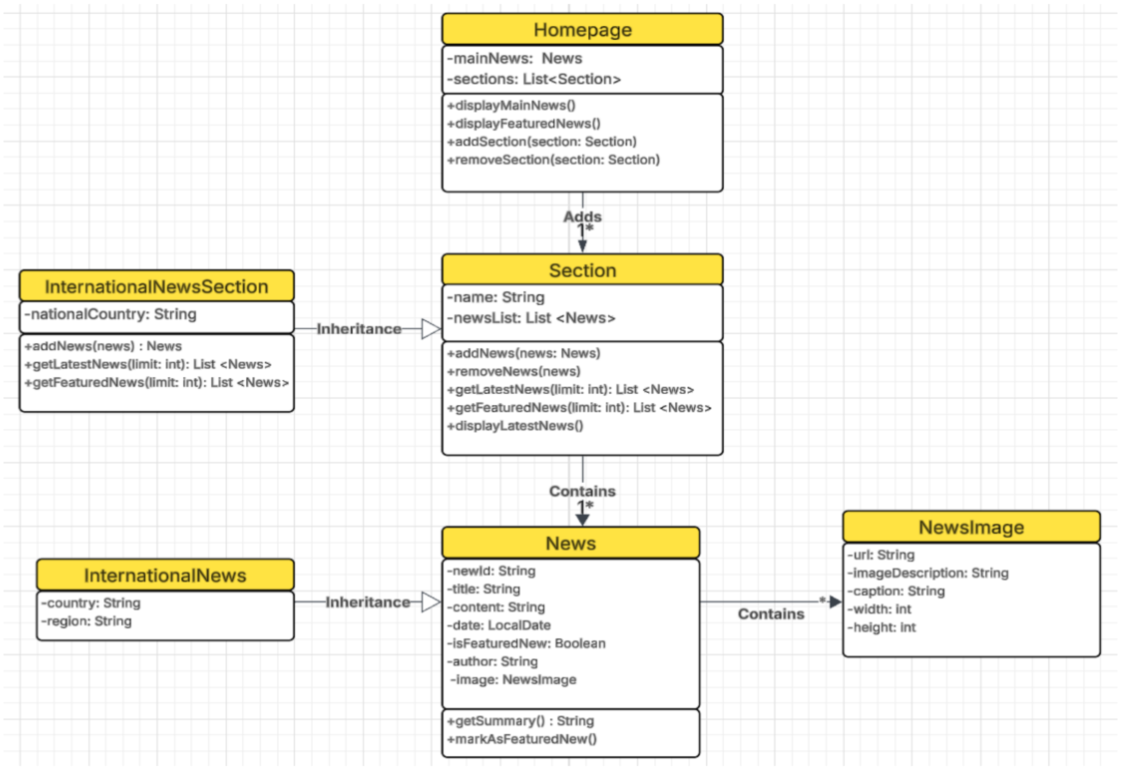
-Objective: Translate the initial conceptual design into a technical blueprint using OOP principles.

- Refined Class Diagram
- Added OOP Pillars
- New Components
- User Story Alignment
- CRC card classes (Section, News, Homepage, International news).
- Enhanced responsibilities

2. Workshop

2.1 Technical design:

- **UML diagram; there's hierarchies, class members (attributes, methods), and relationships:**



2.2 How classes implement or override methods to fulfill user requirements and how they integrate the OOP principles: (updated , table format)

Class	**User Requirements**:	**OOP Integration**:
News (Base Class)	✓ Provides the core structure for news articles (title, content, date) to satisfy the need for displaying news. ✓ Generates summaries (combining title, author, and content snippets) for quick previews.	**Encapsulation**: Hides internal data (title, content) behind private fields and exposes them via controlled methods (getters/setters with validation). For example, it rejects empty titles. **Reusability**: Serves as the parent class for specialized news types (like international news), avoiding code duplication.

NewsImage (Base Class)	✓ Provides a structured way to represent images associated with news articles (url, caption, image description) to fulfill the need for displaying visual content and improving accessibility. ✓ Potentially includes metadata (caption, dimensions) for richer image representation and layout management.	**Encapsulation:** Conceals internal data (url, image description, caption) using private fields and offers controlled access through methods (getters/setters with validation). For example, it might reject empty url's or enforce maximum lengths for description text. **Reusability:** Serves as a component of the News class (through composition), allowing different types of news to easily include image information without repeating image-related attributes in each specialized news class. This promotes a cleaner and more maintainable News hierarchy.
InternationalNews **(Derived Class)**	✓ Extends news with geographic data (country/region) to support the "International News" section. ✓ Filters out national news (non-Colombian news) to meet the demand for global content.	**Inheritance:** Inherits all attributes/methods from News (getSummary) while adding location-specific features. **Polymorphism:** Designed to override methods like getSummary (though not yet implemented) to include country info, enabling dynamic behavior.

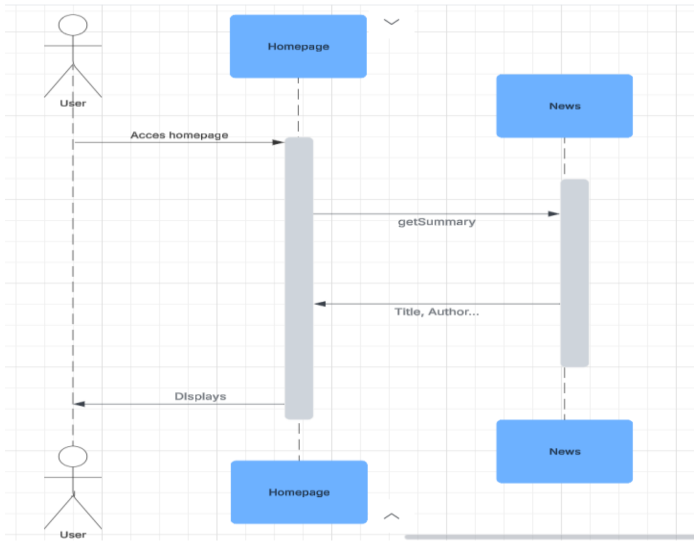
Section (Composition Class)	✓ Groups news by categories ("Latest News," "International") to organize content. ✓ Retrieves recent news (last 9 articles) for the "Latest News" section.	**Composition:** Manages a list of News objects instead of inheriting from News, adhering to "has-a" relationships. **Single Responsibility:** Focuses solely on news organization (sorting by date) without handling rendering or storage**.**
Homepage (Aggregator Class)	✓ Displays main and featured news on the homepage. ✓ Delegates news listing to Section classes ("Latest News" section).	**Delegation:** Relies on Section to fetch news (getLatestNews), promoting modularity. **Dependency Injection:** Accepts News and Section objects via constructor, decoupling dependencies.
InternationalNewsSection (Specialized Logic)	✓ Customizes the standard Section to exclude national news, ensuring only international content appears.	**Polymorphism:** Introduces unique filtering logic (isInternationalNews) without modifying the base Section class.

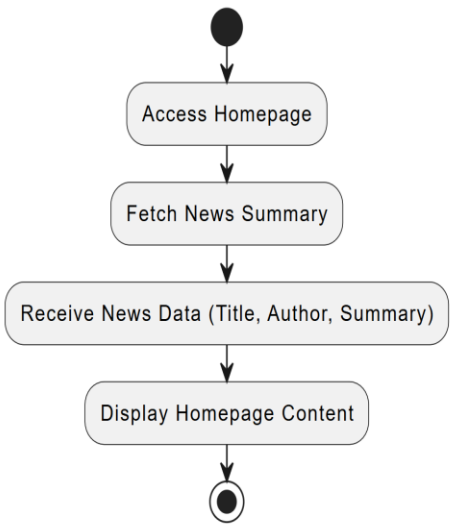
OOP Principles in Action

- **Encapsulation**: Safeguards data integrity (News validates dates).
- **Inheritance**: InternationalNews builds on News without rewriting common features.
- **Polymorphism**: Methods like getSummary can adapt to subclasses (e.g., appending country names).
- **Composition**: Homepage and Section collaborate through object relationships, not inheritance.

2.3 Sequences diagrams and activity diagrams (updated , adding description and diagram of activities); these illustrate interactions between classes and the flow of data:

  - **Diagram “Homepage”:**





- This sequence diagram describes the flow of interaction when a user first accesses the home page.

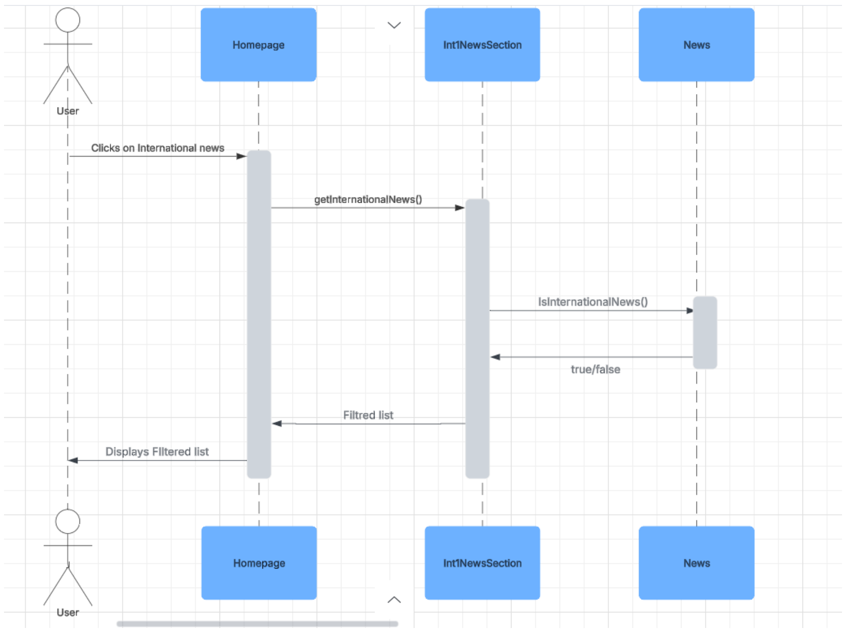
Start of Interaction: The User initiates the process by sending a “Homepage Access” message to the Homepage object.

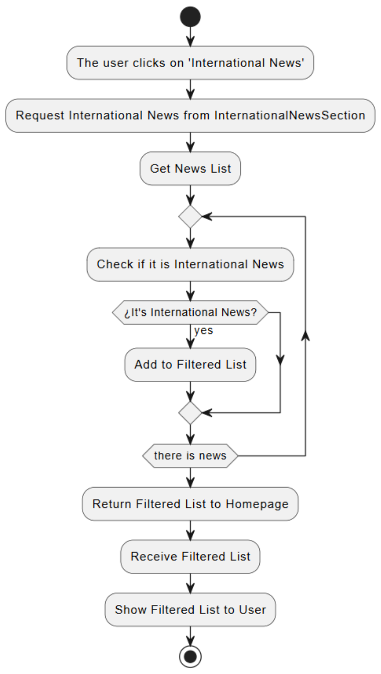
Getting Summary: The Homepage, upon receiving the access message, collaborates with a News object , presumably the featuredNews or one of the initial news items, by sending it a getSummary message.

Receiving Data: The News object responds to the Homepage by returning key data such as the “Title, Author...” and the summary of the news item.

Display: Finally, the Homepage uses the information received to perform a “Displays” action to the User, which implies rendering the visible content of the main page in the user's browser.

- **Diagram “Access to international news”:**




The user clicks on 'International News'
Request International News from InternationalNewsSection
Get News List
Check if it is International News
¿It's International News?
yes
Add to Filtered List
there is news
Return Filtered List to Homepage
Receive Filtered List
Show Filtered List to User

This sequence diagram illustrates the step-by-step interaction when a user requests view international news:

Start of Interaction: The User initiates the action by clicking on the “International News” option in the interface. This message is sent to the Homepage object.

Delegation to International Section: The Homepage receives this request and, to fulfill it, calls the getInternationalNews() method (presumably an implementation of getLatestNews for this section) on the IntNewsSection object (which is InternationalNewsSection).

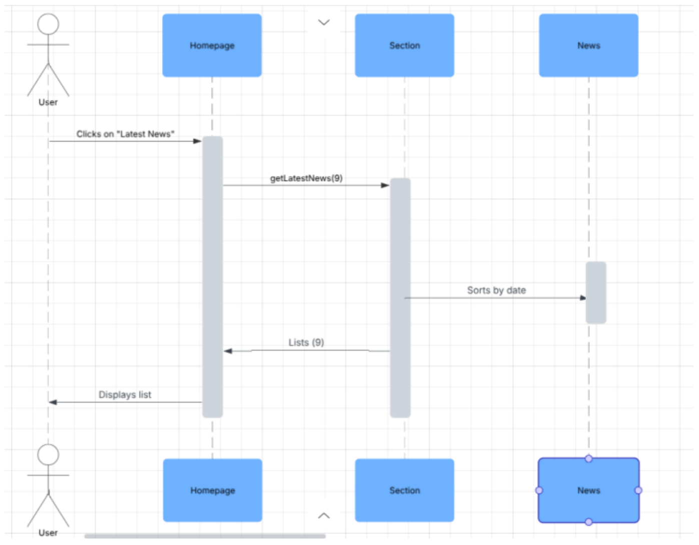
News Filtering: The IntNewsSection processes this request. For each individual News it is considering, it calls the IsInternationalNews() method on the corresponding News object.

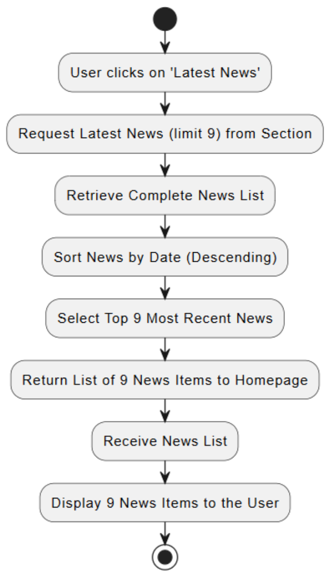
Filtering Response: The News object responds with a boolean value (true or false), indicating whether that particular News meets the criteria of being international (i.e., its country is not the national country configured in InternationalNewsSection).

Return Filtered List: Once IntNewsSection has processed and filtered all the news according to the international criteria, it returns a “Filtered list” of News to the Homepage.

Final Display: Finally, the Homepage receives this filtered list and proceeds to “Displays Filtered list” to the User, showing only the news considered international.

- **Diagram display Latest News:**





User clicks on 'Latest News'
Request Latest News (limit 9) from Section
Retrieve Complete News List
Sort News by Date (Descending)
Select Top 9 Most Recent News
Return List of 9 News Items to Homepage
Receive News List
Display 9 News Items to the User

- This sequence diagram illustrates the step-by-step interaction when a user requests to view the latest news:

Start of Interaction: The User initiates the action by clicking on the "Latest News" option. This message is sent to the Homepage object.

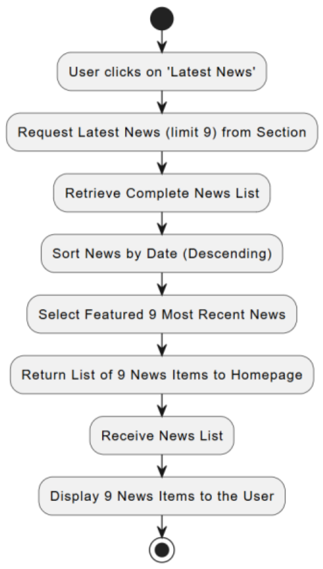
Delegation to the Section: The Homepage receives this request and, to fulfill it, calls the getLatestNews(9) method on a Section object (presumably an instance of LatestNewsSection or a generic Section configured for this purpose). The 9 indicates the limit of news to retrieve.

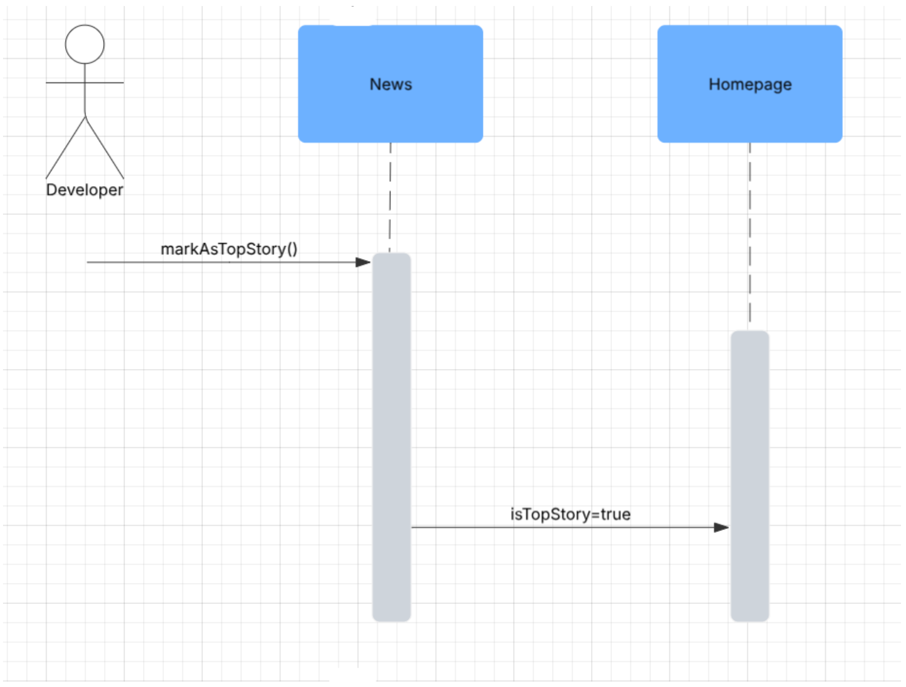
Processing and Sorting: The Section processes the getLatestNews(9) request. To get the most recent news, it invokes an internal action to "Sorts by date" the news it manages. This involves sorting the section's News list by its date attribute in descending order.

List Return: Once sorted, the Section selects the 9 most recent news items and returns a "Lists (9)" to the Homepage.

Final Display: Finally, the Homepage receives this list of 9 news items and proceeds to "Displays list" to the User, presenting the most recent articles in the interface.

- **Diagram Mark as Featured (Top) Story:**


User clicks on 'Latest News'
Request Latest News (limit 9) from Section
Retrieve Complete News List
Sort News by Date (Descending)
Select Featured 9 Most Recent News
Return List of 9 News Items to Homepage
Receive News List
Display 9 News Items to the User



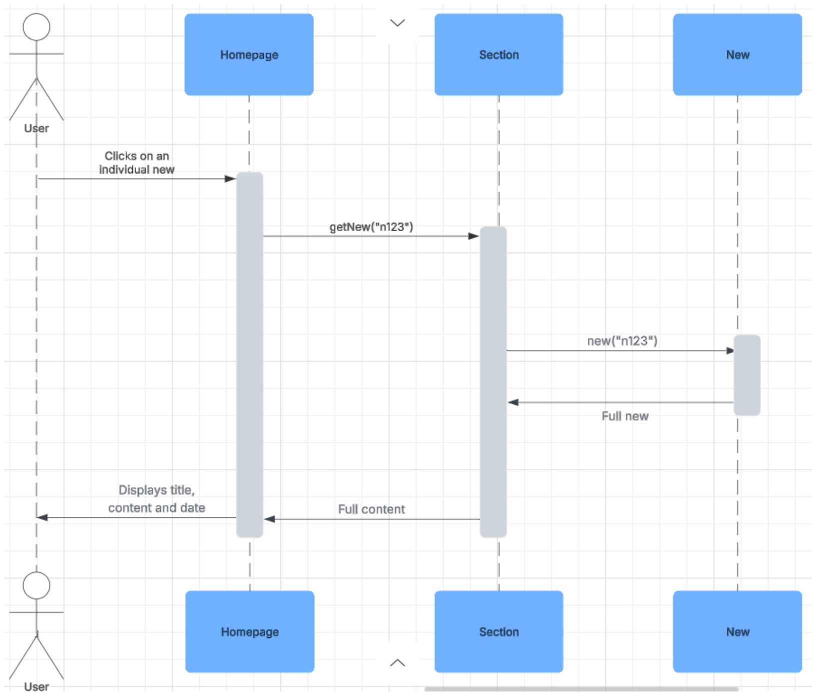
This sequence diagram illustrates the interaction to mark a news item as featured:

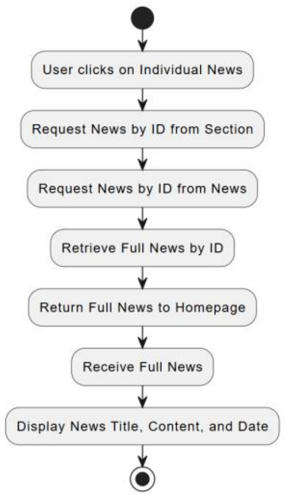
Initiation of the Interaction: The Developer (representing a user with administrative permissions or an automated process) initiates the action by calling the markAsFeaturedStory() method directly on a News object.

State Change: When markAsFeaturedStory() is executed, the News object internally changes its own isFeaturedStory attribute to true.

Notification to Homepage: Then, the diagram shows an isFeaturedStory=true message going from the News object to the Homepage object. This implies that, once a news item has been marked as featured, the Homepage (or some update mechanism) is informed of this change so that it can adjust its content and display the news item as featured,

- **Diagram display individual new:**




User clicks on Individual News
Request News by ID from Section
Request News by ID from News
Retrieve Full News by ID
Return Full News to Homepage
Receive Full News
Display News Title, Content, and Date

This sequence diagram illustrates the step-by-step interaction when a user wishes to view the full details of a specific news item:

Start of Interaction: The User initiates the action by clicking on an “Individual new” (an individual news item) in the interface. This implicit or explicit message is sent to the Homepage object.

Delegation to the Section: The Homepage receives this request and, to get the details of the news item, calls the getNew(“n123”) method (where “n123” would be the ID of the news item) on a Section object. This implies that the Homepage delegates the responsibility of finding the specific news item to the Section that contains it.

Obtaining the complete News: The Section processes this request. To obtain the complete News object, it sends a new(“n123”) message (which implies obtaining the actual News object by its ID) to the News object (or to a news repository, not directly visible here, but implicit).

Full News Return: The News object (or the component that manages it) returns the "Full new" to the Section.

Return Full Content to Homepage: The Section then returns the "Full content" to the Homepage.

Final Display: Finally, the Homepage receives the full content and proceeds to "Displays title, content and date" to the User, presenting the detailed view of the article.

2.4 Implementation Plan for OOP Concepts:

Summary of how our code will realize encapsulation, inheritance and polymorphism:

- **Encapsulation:** Protecting Data
  - Each class controls access to its internal attributes through private methods. For example, the News class keeps its title and content as private properties, only allowing modifications through methods that include validations (like ensuring the title isn't empty).
- **Inheritance:** Specializing Behavior
  - The InternationalNews class inherits from the News, leveraging all its base attributes and methods while adding specific properties like country of origin, demonstrating the principle of extending without modifying.
- **Polymorphism:** Flexible Behavior
  - When the system calls the "getSummary" method, the InternationalNews class responds differently than the base News class by automatically including country information. This allows processing different news types uniformly.
  - Also, when accessing sections from the Homepage and calling getLatestNews(), the InternationalNewsSection will execute its own logic to retrieve the latest news (excluding those from Colombia), while other sections, such as the general section, use different logic and simply list the most recent news by date without considering other aspects.
- **Composition:** Building Relationships

  - The Section class contains and manages collections of News objects, coordinating how news is grouped and retrieved. The Homepage in turn uses these sections to present information to users.

- **System Flow:**
  - Creation: News instances are created with their basic data.
  - Organization: News is assigned to thematic sections.
  - Presentation: The Homepage requests news from sections and displays it.
  - Interaction: Users can navigate through the app without any kind of log in.

- **Advantages of This Design:**
  - Simplicity: Working entirely in-memory avoids infrastructure complexity.
  - Flexibility: Easy to extend with new news types.
  - Maintainability: Each class has clear, well-defined responsibilities.
  - Scalability: The structure easily accommodates new features.

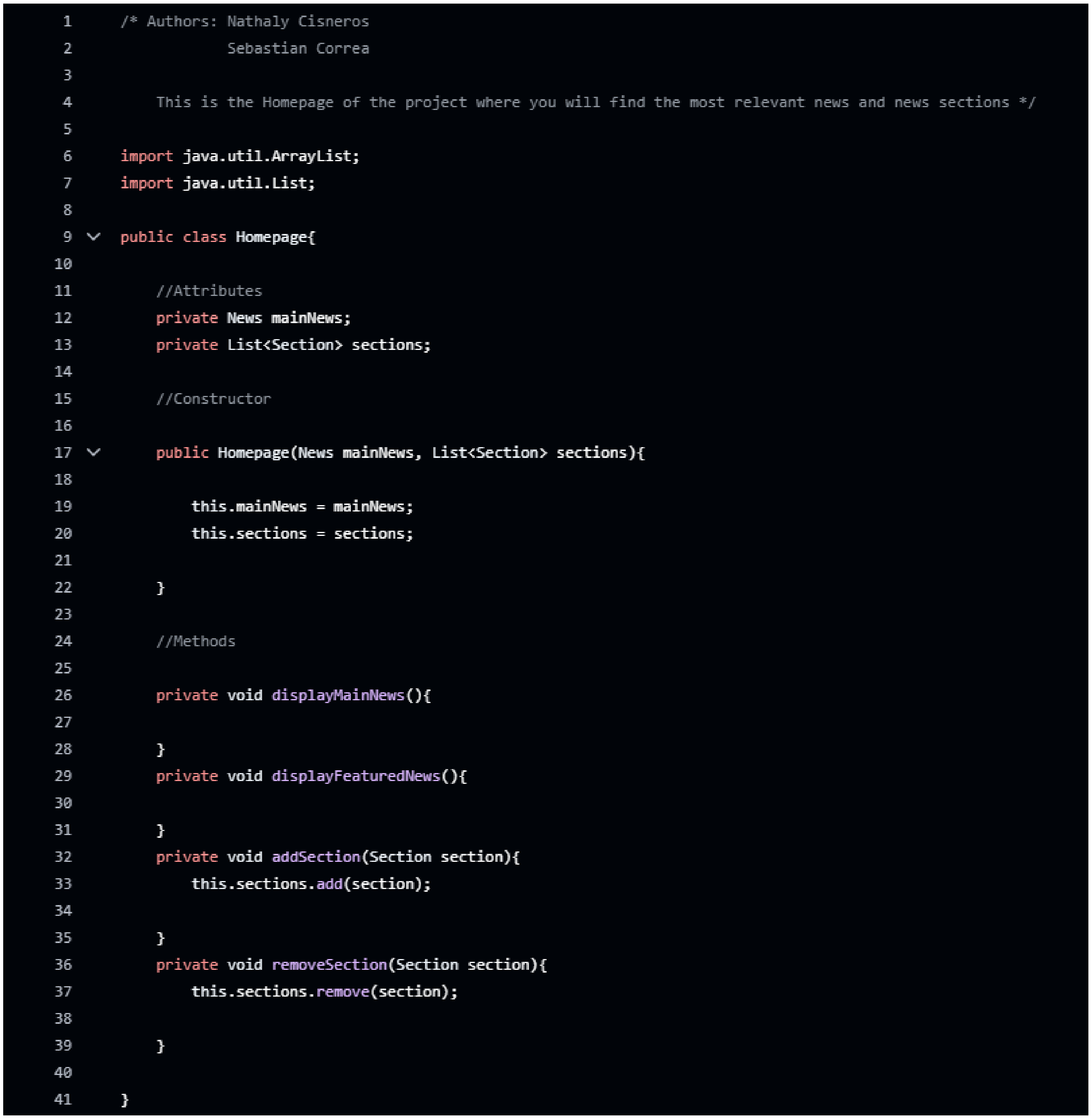
- **Logical Structure:**
  - Models: News (base), InternationalNews (specialized).
  - Managers: Section (organizes news).
  - Interface: Homepage (presents information).

This approach leverages OOP principles to create a cohesive system where components collaborate while maintaining independence, ideal for projects requiring clarity and evolution capability. The complete in-memory implementation ensures lightweight operation.

2.6. Initial code snippets illustrating class definitions, placeholder methods,

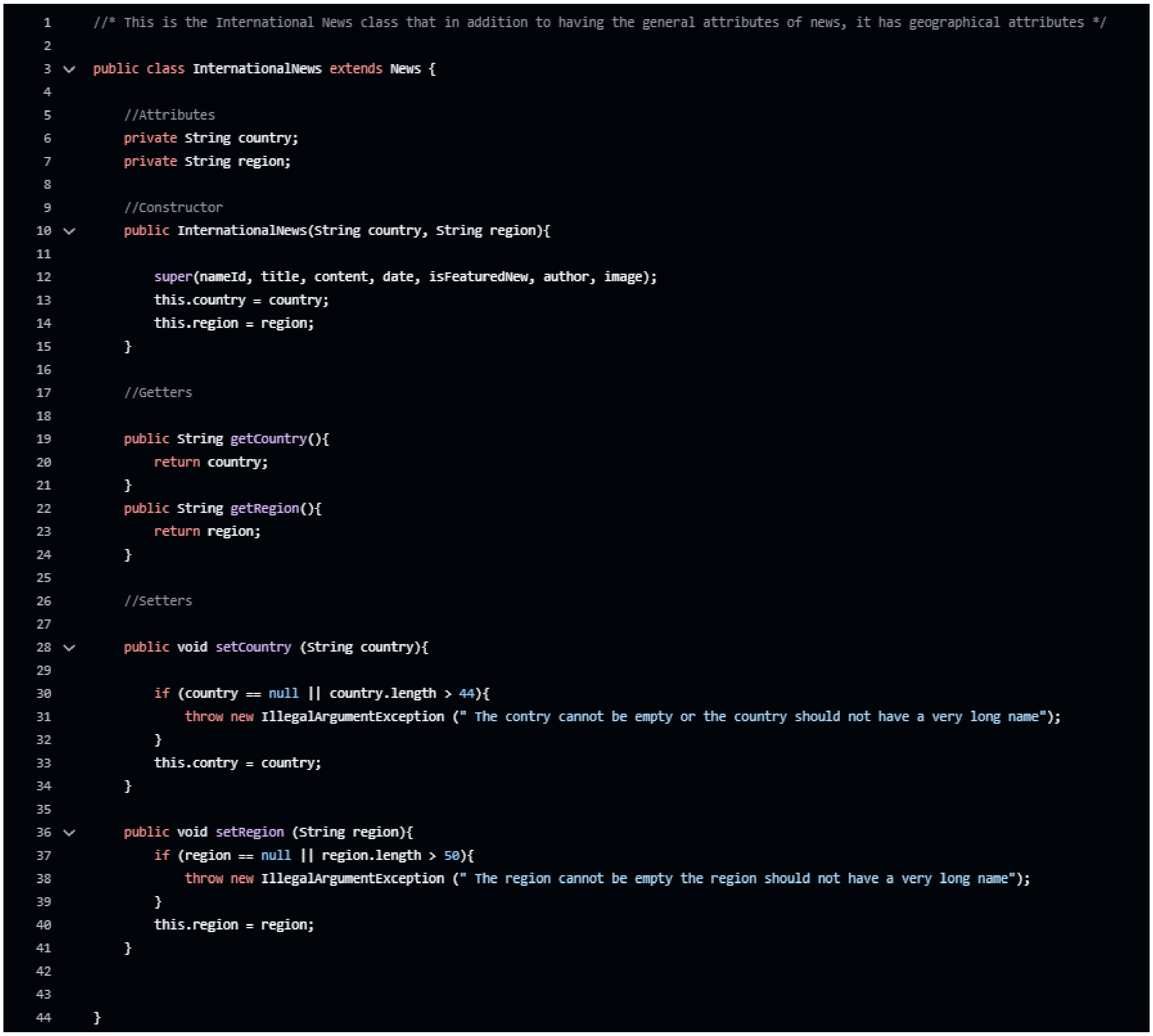
or abstract classes:

Home page:



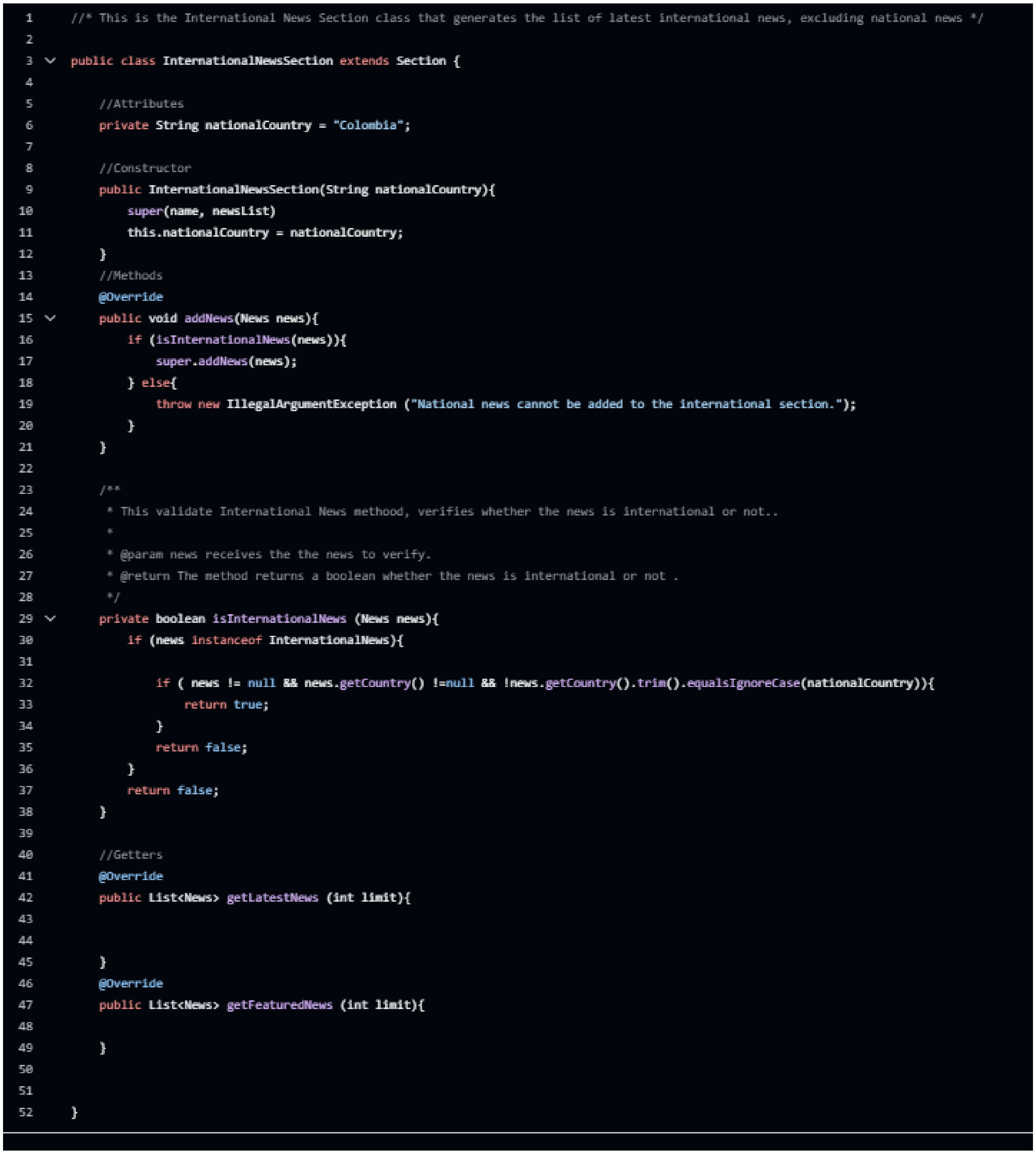
```java
/* Authors: Nathaly Cisneros
            Sebastian Correa

    This is the Homepage of the project where you will find the most relevant news and news sections */

import java.util.ArrayList;
import java.util.List;

public class Homepage{

    //Attributes
    private News mainNews;
    private List<Section> sections;

    //Constructor

    public Homepage(News mainNews, List<Section> sections){

        this.mainNews = mainNews;
        this.sections = sections;

    }

    //Methods

    private void displayMainNews(){

    }
    private void displayFeaturedNews(){

    }
    private void addSection(Section section){
        this.sections.add(section);

    }
    private void removeSection(Section section){
        this.sections.remove(section);

    }

}
```

InternationalNews:



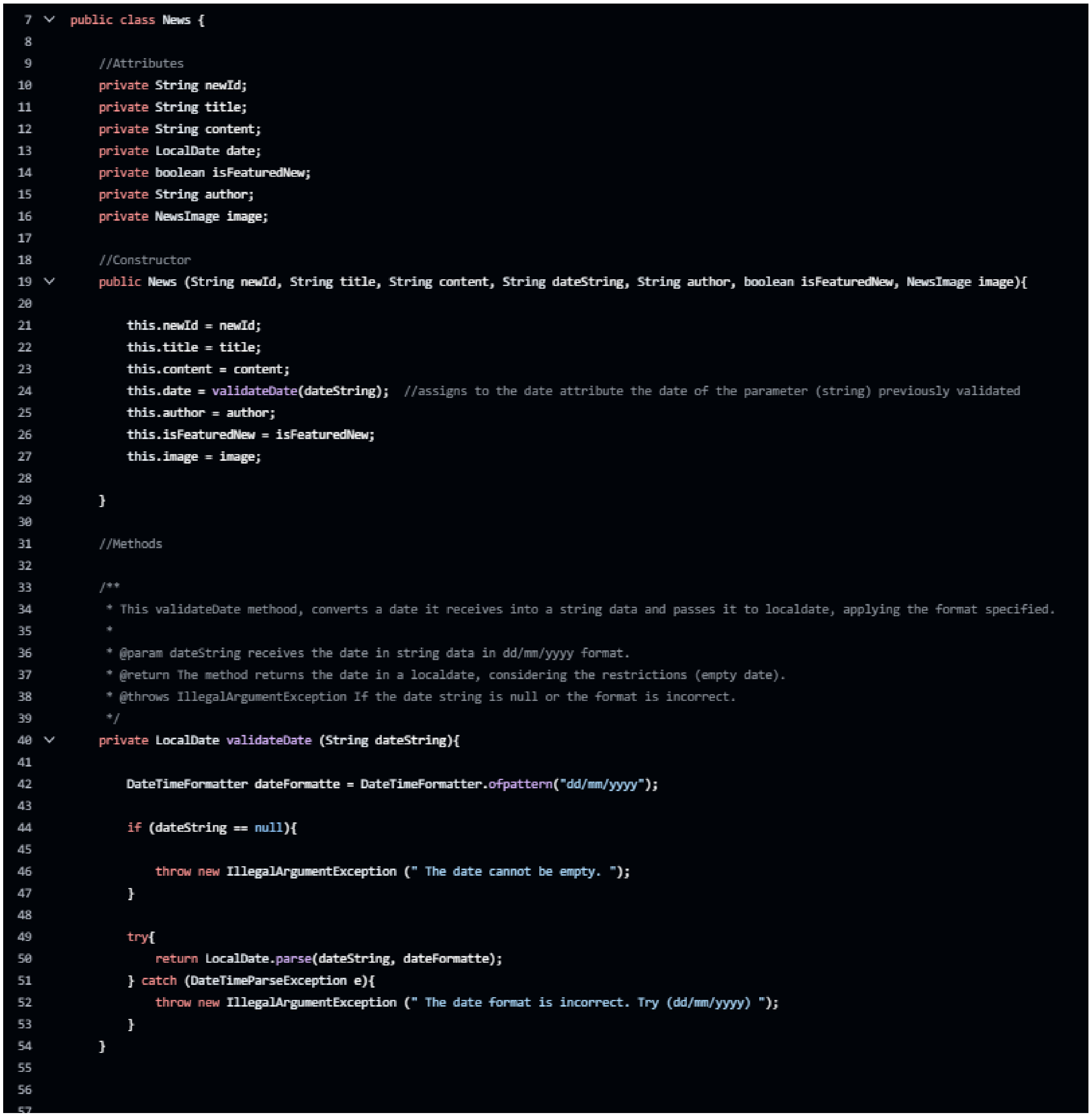
```java
//* This is the International News class that in addition to having the general attributes of news, it has geographical attributes */

public class InternationalNews extends News {

    //Attributes
    private String country;
    private String region;

    //Constructor
    public InternationalNews(String country, String region){

        super(nameId, title, content, date, isFeaturedNew, author, image);
        this.country = country;
        this.region = region;
    }

    //Getters

    public String getCountry(){
        return country;
    }
    public String getRegion(){
        return region;
    }

    //Setters

    public void setCountry (String country){

        if (country == null || country.length > 44){
            throw new IllegalArgumentException (" The contry cannot be empty or the country should not have a very long name");
        }
        this.contry = country;
    }

    public void setRegion (String region){
        if (region == null || region.length > 50){
            throw new IllegalArgumentException (" The region cannot be empty the region should not have a very long name");
        }
        this.region = region;
    }


}
```

InternationalNewsSection:



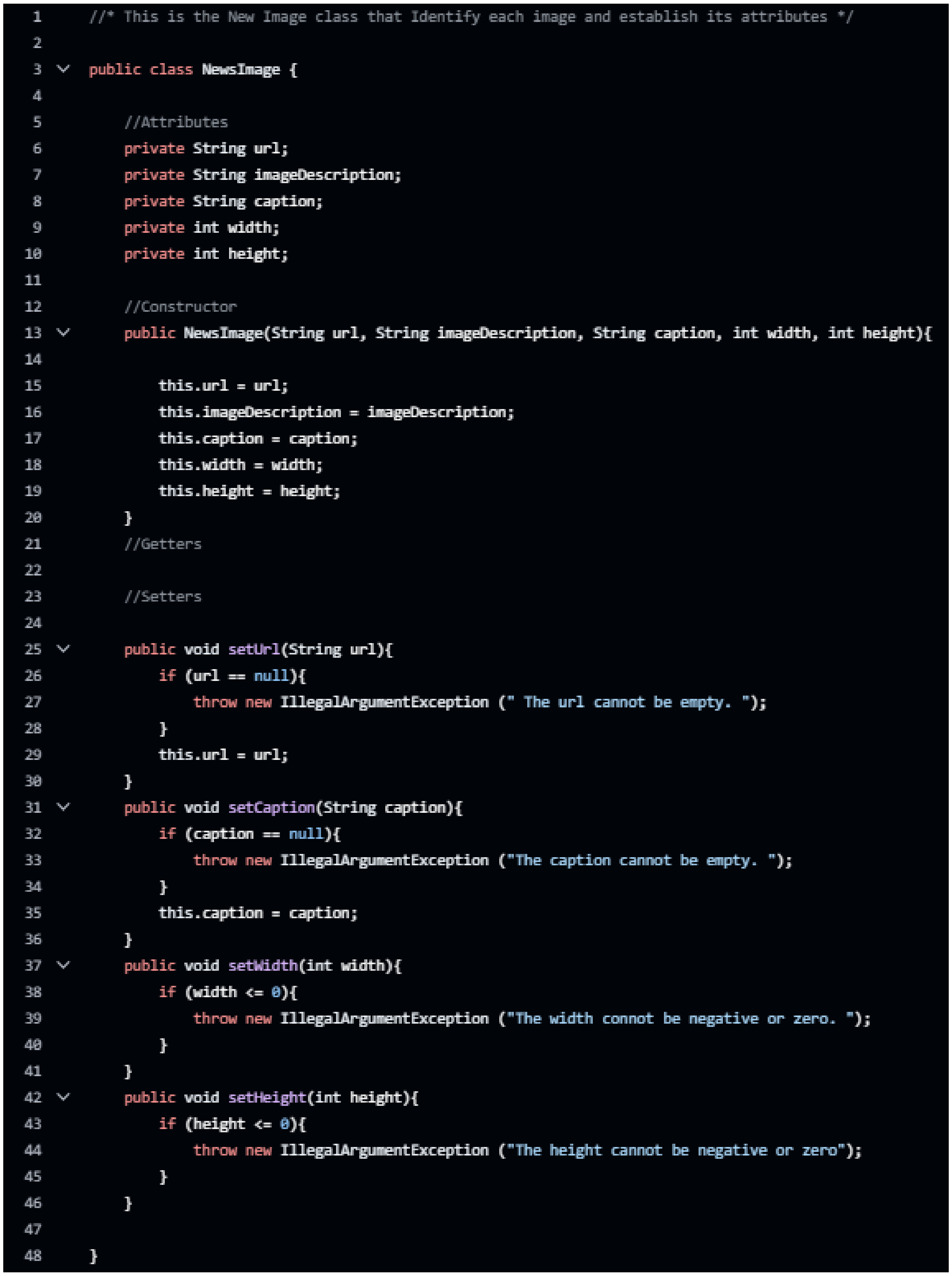
```java
//* This is the International News Section class that generates the list of latest international news, excluding national news */

public class InternationalNewsSection extends Section {

    //Attributes
    private String nationalCountry = "Colombia";

    //Constructor
    public InternationalNewsSection(String nationalCountry){
        super(name, newsList)
        this.nationalCountry = nationalCountry;
    }
    //Methods
    @Override
    public void addNews(News news){
        if (isInternationalNews(news)){
            super.addNews(news);
        } else{
            throw new IllegalArgumentException ("National news cannot be added to the international section.");
        }
    }

    /**
     * This validate International News methood, verifies whether the news is international or not..
     *
     * @param news receives the the news to verify.
     * @return The method returns a boolean whether the news is international or not .
     */
    private boolean isInternationalNews (News news){
        if (news instanceof InternationalNews){

            if ( news != null && news.getCountry() !=null && !news.getCountry().trim().equalsIgnoreCase(nationalCountry)){
                return true;
            }
            return false;
        }
        return false;
    }

    //Getters
    @Override
    public List<News> getLatestNews (int limit){


    }
    @Override
    public List<News> getFeaturedNews (int limit){

    }


}
```

News:



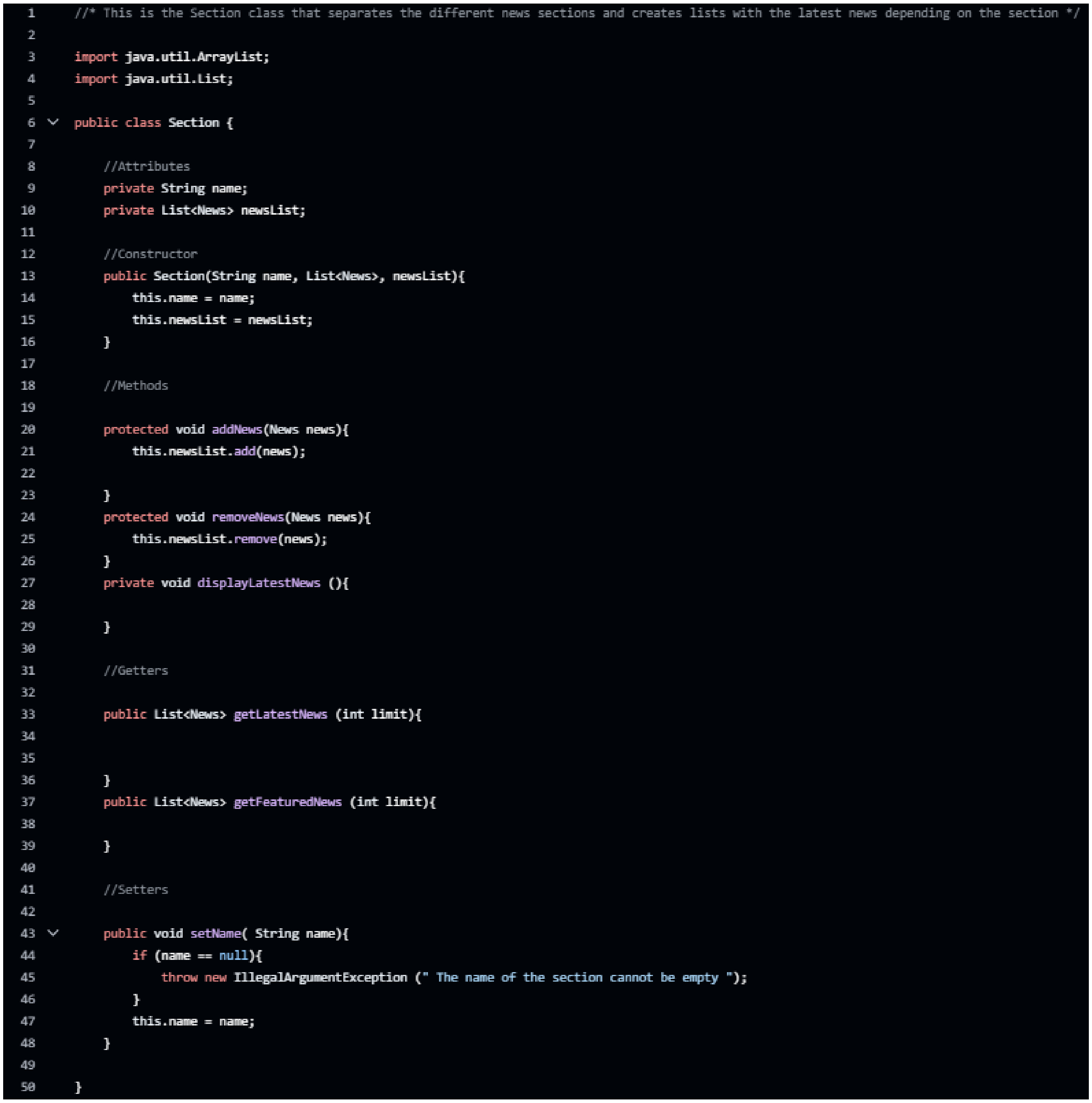
```java
public class News {

    //Attributes
    private String newId;
    private String title;
    private String content;
    private LocalDate date;
    private boolean isFeaturedNew;
    private String author;
    private NewsImage image;

    //Constructor
    public News (String newId, String title, String content, String dateString, String author, boolean isFeaturedNew, NewsImage image){

        this.newId = newId;
        this.title = title;
        this.content = content;
        this.date = validateDate(dateString);  //assigns to the date attribute the date of the parameter (string) previously validated
        this.author = author;
        this.isFeaturedNew = isFeaturedNew;
        this.image = image;

    }

    //Methods

    /**
     * This validateDate method, converts a date it receives into a string data and passes it to localdate, applying the format specified.
     *
     * @param dateString receives the date in string data in dd/mm/yyyy format.
     * @return The method returns the date in a localdate, considering the restrictions (empty date).
     * @throws IllegalArgumentException If the date string is null or the format is incorrect.
     */
    private LocalDate validateDate (String dateString){

        DateTimeFormatter dateFormatte = DateTimeFormatter.ofpattern("dd/mm/yyyy");

        if (dateString == null){

            throw new IllegalArgumentException (" The date cannot be empty. ");
        }

        try{
            return LocalDate.parse(dateString, dateFormatte);
        } catch (DateTimeParseException e){
            throw new IllegalArgumentException (" The date format is incorrect. Try (dd/mm/yyyy) ");
        }
    }


```

NewsImage:



```java
//* This is the New Image class that Identify each image and establish its attributes */

public class NewsImage {

    //Attributes
    private String url;
    private String imageDescription;
    private String caption;
    private int width;
    private int height;

    //Constructor
    public NewsImage(String url, String imageDescription, String caption, int width, int height){

        this.url = url;
        this.imageDescription = imageDescription;
        this.caption = caption;
        this.width = width;
        this.height = height;
    }
    //Getters

    //Setters

    public void setUrl(String url){
        if (url == null){
            throw new IllegalArgumentException (" The url cannot be empty. ");
        }
        this.url = url;
    }
    public void setCaption(String caption){
        if (caption == null){
            throw new IllegalArgumentException ("The caption cannot be empty. ");
        }
        this.caption = caption;
    }
    public void setWidth(int width){
        if (width <= 0){
            throw new IllegalArgumentException ("The width connot be negative or zero. ");
        }
    }
    public void setHeight(int height){
        if (height <= 0){
            throw new IllegalArgumentException ("The height cannot be negative or zero");
        }
    }

}
```

Section:



```java
//* This is the Section class that separates the different news sections and creates lists with the latest news depending on the section */

import java.util.ArrayList;
import java.util.List;

public class Section {

    //Attributes
    private String name;
    private List<News> newsList;

    //Constructor
    public Section(String name, List<News>, newsList){
        this.name = name;
        this.newsList = newsList;
    }

    //Methods

    protected void addNews(News news){
        this.newsList.add(news);

    }
    protected void removeNews(News news){
        this.newsList.remove(news);
    }
    private void displayLatestNews (){

    }

    //Getters

    public List<News> getLatestNews (int limit){


    }
    public List<News> getFeaturedNews (int limit){

    }

    //Setters

    public void setName( String name){
        if (name == null){
            throw new IllegalArgumentException (" The name of the section cannot be empty ");
        }
        this.name = name;
    }

}
```

2.7 UML-to-Code Correspondence:

Encapsulation (Data Protection)

- **UML Shows**: Private attributes (-country, -title) and public methods (+getSummary()).
- **Code Implementation**:
  - News class uses private String title with public setTitle() (validates non-null).
  - InternationalNews hides country/region but exposes them via validated setters (setCountry() checks length ≤ 44 chars).

Inheritance (Specialization)

- **UML Shows**: InternationalNews inherits from News (attributes/methods under "Inheritance" box).
- **Code Implementation**:
  - InternationalNews extends News to reuse title, content, etc. while adding geographic properties.
  - **Design Rationale**: Avoids duplicating shared attributes like date or author.

Polymorphism (Flexible Behavior)

- **UML Suggests**: Potential method overriding (getLatestNews() differs in InternationalNewsSection).
- **Code Implementation**:
  - InternationalNewsSection overrides getLatestNews() to exclude Colombian news via isInternationalNews().
  - **Design Rationale**: Enables sections to customize filtering while maintaining a consistent interface.

Composition (Relationships)

- **UML Shows**: Section contains List<News> (aggregation).
- **Code implementation:**
  - Section manages newsList to group news thematically.
  - **Design Rationale**: Decouples news storage (Section) from news logic (News).

Design Rationale Explained

System Flow Alignment

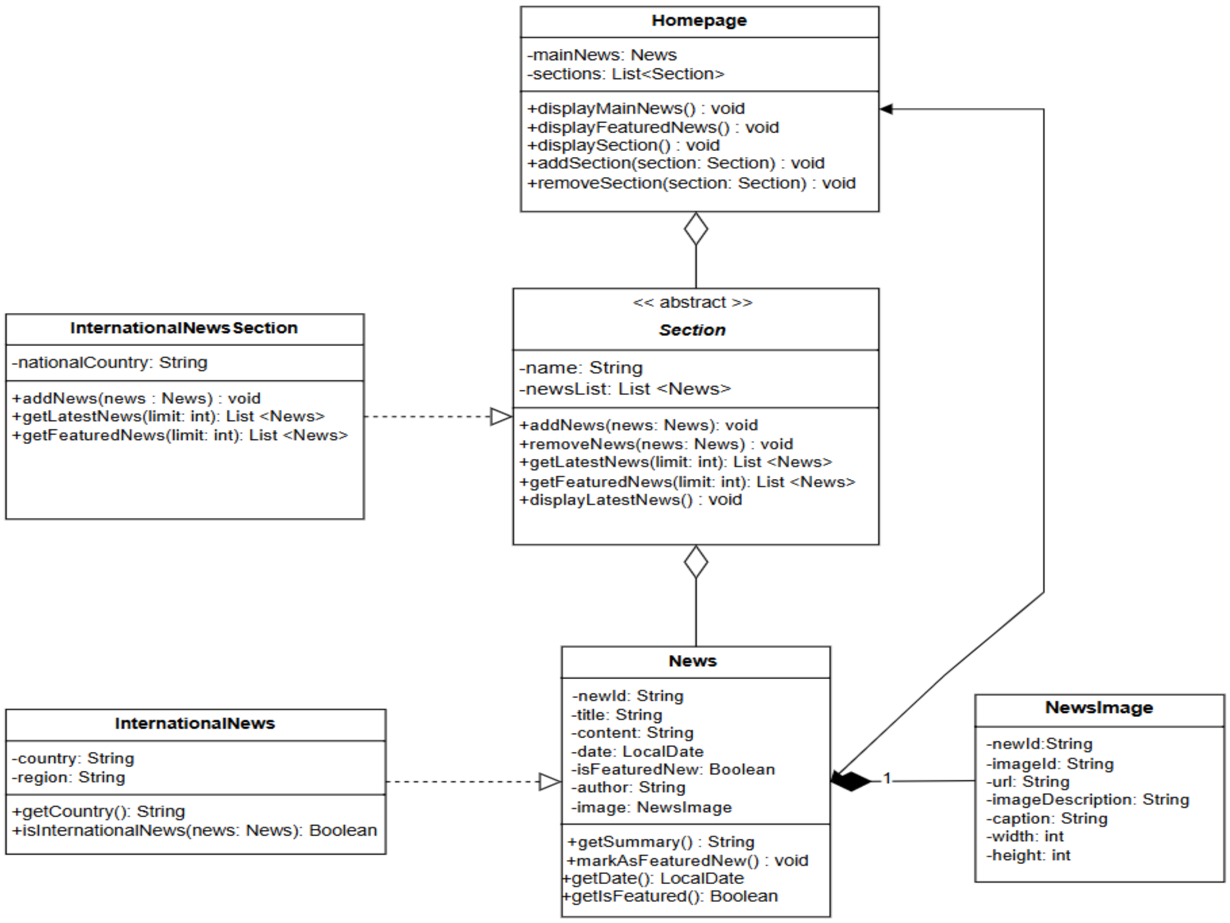
1. **Creation**: Matches UML's News/InternationalNews initialization.
2. **Organization**: Section subclasses (like InternationalNewsSection) implement UML's getLatestNews() for their specific logic.
3. **Presentation**: Homepage (implied in UML) queries sections to display results.

3 Workshop

3.1 Enhanced UML Diagrams:

3.1.1 new interfaces, abstract classes, or design patterns that ensure SOLID compliance.

UML diagram using Solid:


Homepage
-mainNews: News
-sections: List<Section>
+displayMainNews() : void
+displayFeaturedNews() : void
+displaySection() : void
+addSection(section: Section) : void
+removeSection(section: Section) : void
<< abstract >>
Section
-name: String
-newsList: List <News>
+addNews(news: News): void
+removeNews(news: News) : void
+getLatestNews(limit: int): List <News>
+getFeaturedNews(limit: int): List <News>
+displayLatestNews() : void
InternationalNewsSection
-nationalCountry: String
+addNews(news : News) : void
+getLatestNews(limit: int): List <News>
+getFeaturedNews(limit: int): List <News>
News
-newId: String
-title: String
-content: String
-date: LocalDate
-isFeaturedNew: Boolean
-author: String
-image: NewsImage
+getSummary() : String
+markAsFeaturedNew() : void
+getDate(): LocalDate
+getIsFeatured(): Boolean
InternationalNews
-country: String
-region: String
+getCountry(): String
+isInternationalNews(news: News): Boolean
NewsImage
-newId:String
-imageId: String
-url: String
-imageDescription: String
-caption: String
-width: int
-height: int
1

- By adjusting the composition-to-aggregation ratios where most appropriate. These changes strengthen adherence to SOLID principles, resulting in a more robust, flexible design that is easier to maintain and extend for future news system functionality.

Homepage:

Change: The relationship with Section has been corrected to Aggregation, indicating that Homepage contains Sections, but these can exist independently.

SOLID Implementation:

- ✓ Single Responsibility Principle (SRP): Homepage maintains its responsibility for orchestrating the display of the main news and sections. Aggregation reinforces that it does not existentially own the Sections, but uses them to fulfill its purpose.

Section:

Change: Section has been redefined as an abstract class. This allows it to have instance attributes (name, newsList) and concrete methods (addNews, removeNews, displayLatestNews() if generic) that can be shared, along with abstract methods (getLatestNews(), getFeaturedNews()) that subclasses must implement. The relationship with News to Aggregation) has been corrected, reflecting that a section groups news, but news can exist independently of a section.

SOLID Implementation:

- ✓ <u>Single Responsibility Principle (SRP):</u> Section now focuses on defining the contract and managing the generic news collection for a section. Responsibility for how "latest" or "featured" news is sourced is delegated to its subclasses.
- ✓ <u>Open/Closed Principle (OCP):</u> The abstract class Section is open for extension (new subclasses can implement their own getLatestNews and getFeaturedNews criteria) but closed for modification (the base logic of addNews, removeNews and displayLatestNews is stable in the superclass).
- ✓ <u>skov Substitution Principle (LSP):</u> Being an abstract class, the subclasses of Section (InternationalNewsSection) can be substituted by Section without altering the correctness of the program, since they will all implement the abstract methods and share the concrete ones in a coherent way.

InternationalNewsSection:

SOLID Implementation:

- ✓ Liskov Substitution Principle (LSP): InternationalNewsSection is a subtype of Section and is expected to implement the abstract methods of Section (getLatestNews, getFeaturedNews) in a way that is consistent with the definition of Section, but with the specialized logic of filtering international news.
- ✓ Dependency Inversion Principle (DIP) - subtly: Although not explicitly a dependency injection pattern, by relying on the Section abstraction, other parts of the system will depend on the interface or abstract class, making it easy to change or add new types of sections.

News:

Change: Added methods +getDate(): LocalDate and +getIsFeatured(): Boolean to facilitate data manipulation by other classes.

SOLID Implementation:

- ✓ Single Responsibility Principle (SRP): News now has the sole responsibility to represent and manage the data and behaviors inherent in an individual news item. The added methods are for accessing your own data, not for performing complex external operations.

NewsImage:

Change: To maintain the composition with News, the newid attribute was added.

SOLID Implementation:

- ✓ Single Responsibility Principle (SRP): NewsImage has the sole responsibility of representing and managing the data and properties of an image associated with a news item.
- ✓ It does not handle the business logic of the news or its general display, keeping its focus on its own attributes and access methods.

InternationalNews:

Change: Added method +getCountry(): String directly in InternationalNews.

SOLID Implementation:

- ✓ Single Responsibility Principle (SRP): Reinforces that InternationalNews is responsible for managing and providing information specific to an international news item, including its country of origin.

3.1.2 How do these principles improve the system's transaction flows?

The application of SOLID principles in the design of the news replication system transforms the operation flows from being monolithic and fragile to being modular, clear and adaptable. Greater development efficiency is achieved by reducing the need to modify existing code, greater reliability by minimizing errors caused by unexpected changes, and greater scalability by allowing easy integration of new functionality without significant reengineering.

Improved Clarity and Consistency of Data and Operation Flows, by ensuring that each class has a single reason to change, operation flows become more predictable and easier to understand, for example when the "information about a news item" needs to be obtained, the flow goes directly to the News class (news.getTitle(), news.getContent()). There is no ambiguity about which class has that responsibility.

Facilitation of Flow Extension and Adaptation (OCP, LSP), Operation flows are highly adaptable to new requirements without requiring modifications to the existing code base, If a new type of section is added the flow to get news for that section simply involves implementing the abstract methods getLatestNews() and getFeaturedNews() of Section, this improving Flexibility and Robustness of Flows

3.2 SOLID-Focused Implementation:

3.2.1 How Single Responsibility is achieved by splitting or reorganizing classes with multiple roles?

The class diagram design for the digital news system has been significantly optimized to adhere to SOLID principles, e.g. News and NewsImage have been corrected to manage data and their respective composition relationships from News to NewsImage or aggregation from Section to News and Homepage to Section, clearly defining their responsibilities (SRP).

Section has been transformed into an abstract class to provide a base contract and shared logic, while allowing polymorphic extension in its subclasses, thus ensuring OCP and LSP; furthermore, the relocation of methods such as getCountry() to InternationalNews and the dependency of abstractions, Homepage using Section, reinforce design cohesion, encapsulation and DIP, resulting in a more robust and extensible system.

3.2.2 Implement Open/Closed

This principle is mainly implemented through the abstract class Section

Open to extension: The Section class defines abstract methods such as getLatestNews() and getFeaturedNews(). This allows new functionality (such as the creation of a SportsNewsSection or LocalNewsSection) to extend the behavior of the system by simply creating new subclasses of Section. Each new Section can implement its own criteria for determining which news items are “latest” or “featured” without modifying the base Section class.

Close to Modification: Section's concrete methods, such as addNews(), removeNews() and displayLatestNews() (if its display implementation is universal), do not need to be modified when new subclasses are added. This ensures that existing code is stable and that new features do not introduce regressions on already implemented functionality.

3.2.3 adherence to Liskov Substitution when creating subclasses or interfaces.

The InternationalNews class (subtype of News) extends News. This implies that any code that expects a News object (for example, Section methods that operate on List<News>) can receive an instance of InternationalNews and operate on it without problems.

Similarly, InternationalNewsSection (a subtype of Section) can be used anywhere a Section object is expected (such as on Homepage). This allows Homepage to interact with different types of sections in a uniform way, calling methods such as getLatestNews() without knowing the underlying specific implementation

3.2.4 Reflect on Interface Segregation by proving that no class is forced to implement methods it does not need.

By ensuring that each class has a single responsibility (SRP), it automatically avoids the need for interfaces containing an excess of unrelated methods, e.g., News is not forced to implement display or collection management methods, since those responsibilities are segregated in other classes or layers.

3.2.5 Dependency Inversion by injecting

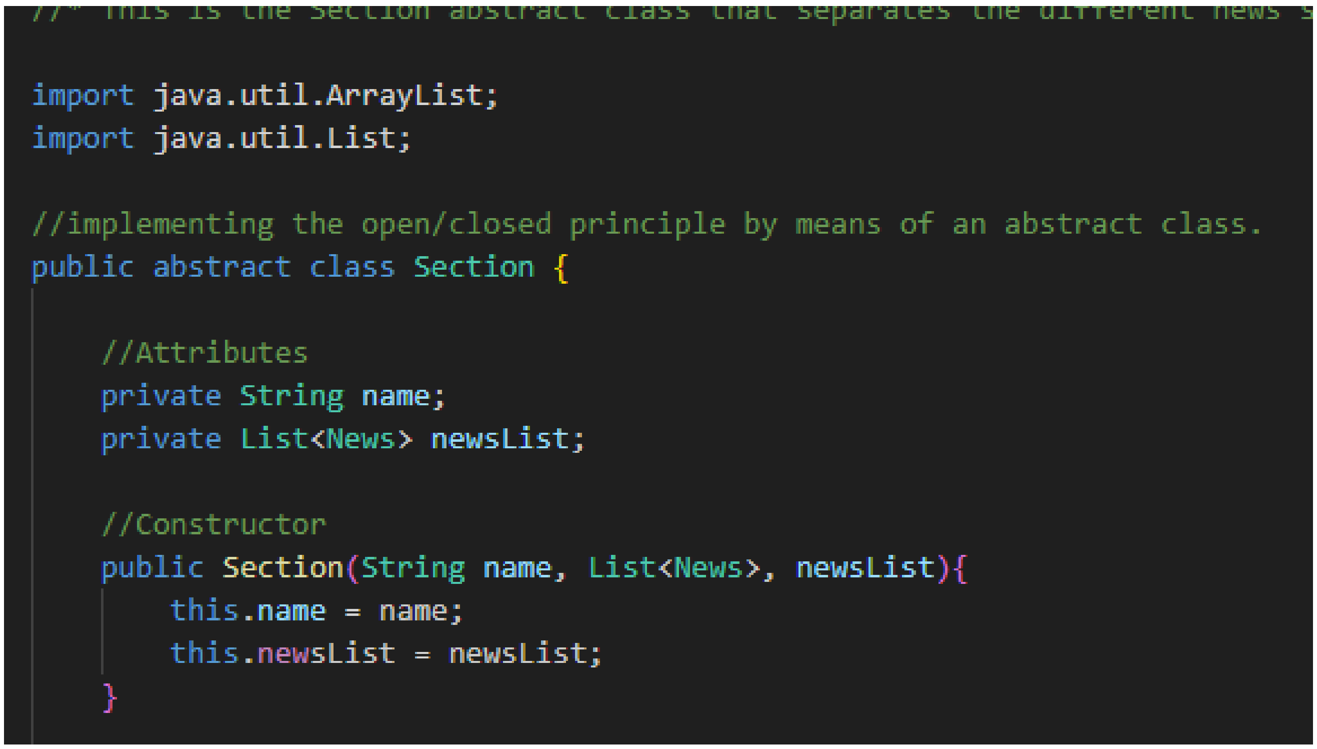
The Homepage (the high-level module that orchestrates the interface) does not depend directly on concrete implementations such as InternationalNewsSection. Instead, it depends on the Section abstraction (the abstract class). This is illustrated by the aggregation relationship of Homepage with Section. This design allows the Homepage to handle any type of Section without needing to be modified when new section specializations are introduced.

The conscious application of SOLID principles in this news replication system design has not only corrected initial inconsistencies, but has established a solid foundation for agile and scalable development. The clear separation of responsibilities, the ability to extend the system without modifying the code base, the substitutability of components, the specialization of interfaces, and the reversal of dependencies all contribute to a more robust, flexible, and maintainable software.

3.3 Work in Progress Code & Documentation:

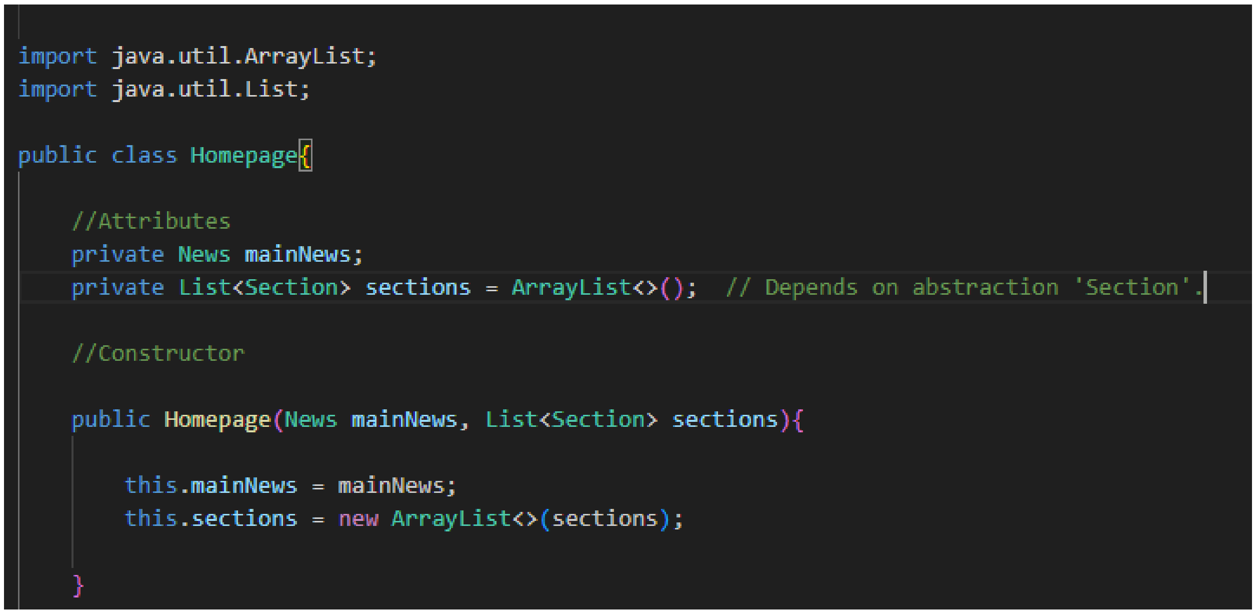
Applying solid, up to date designs in code:

- **Section:**



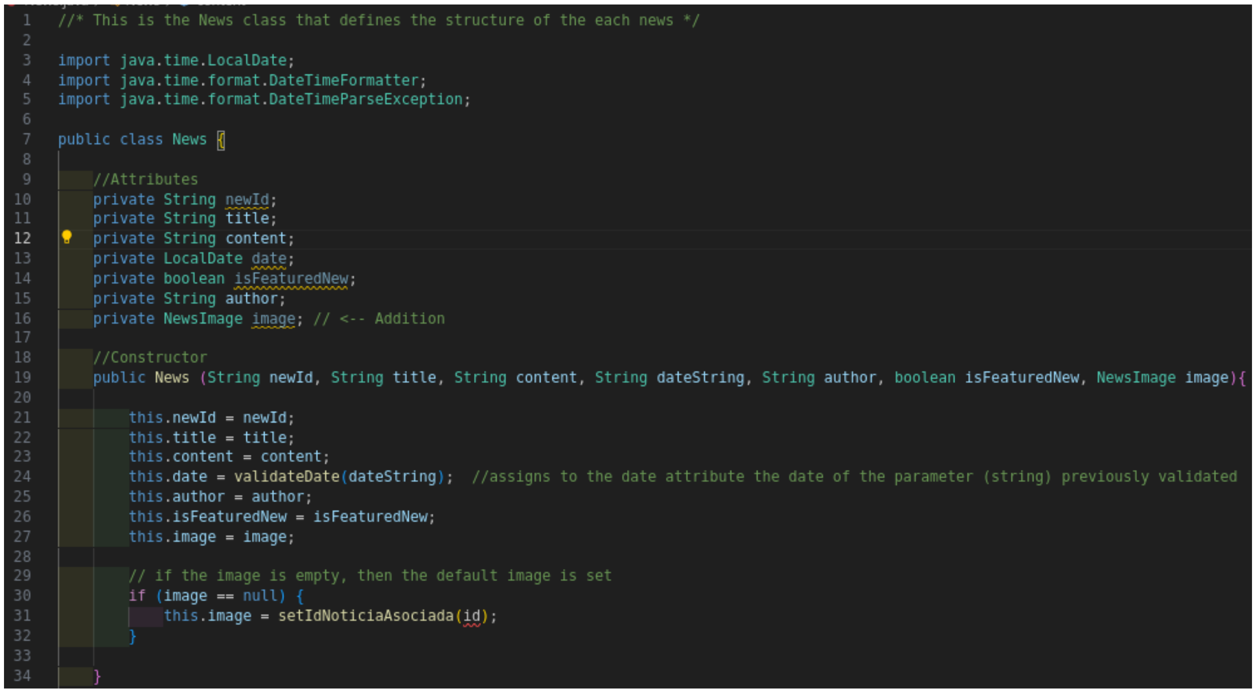
```java
//* This is the Section abstract class that separates the different news s

import java.util.ArrayList;
import java.util.List;

//implementing the open/closed principle by means of an abstract class.
public abstract class Section {

    //Attributes
    private String name;
    private List<News> newsList;

    //Constructor
    public Section(String name, List<News>, newsList){
        this.name = name;
        this.newsList = newsList;
    }
```

- **Homepage:**



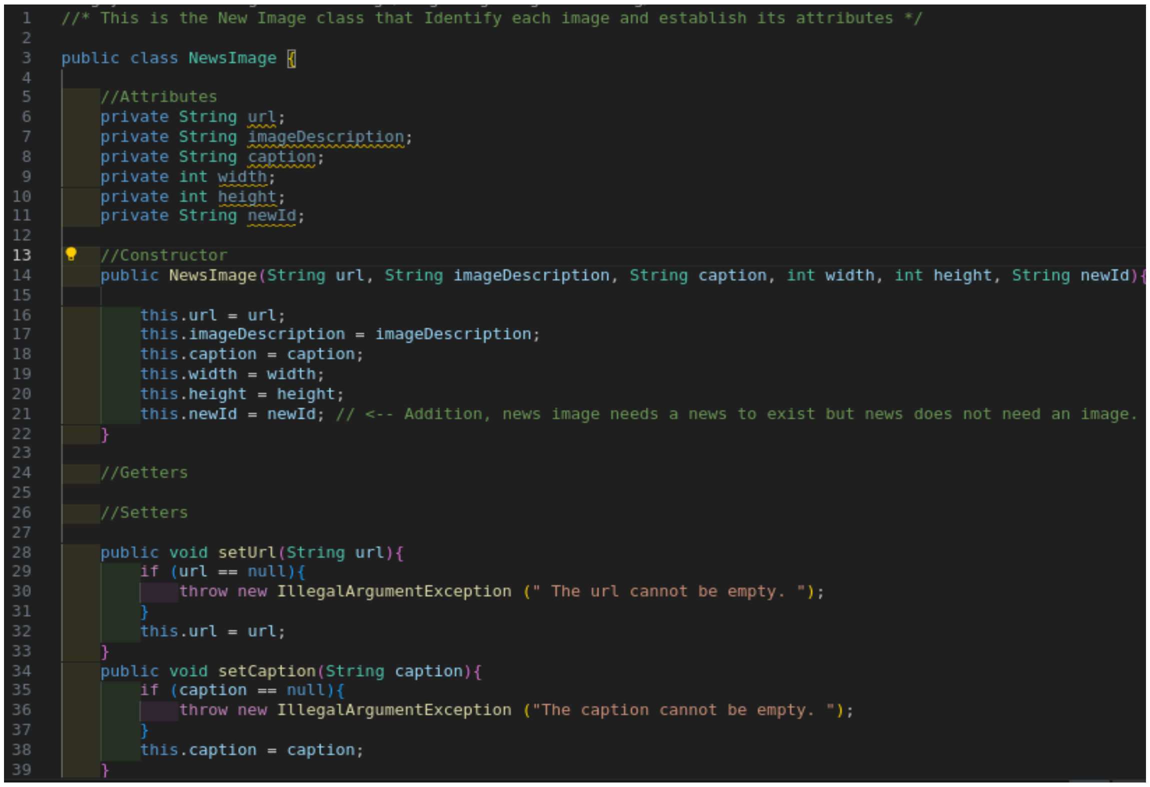
```java
import java.util.ArrayList;
import java.util.List;

public class Homepage{

    //Attributes
    private News mainNews;
    private List<Section> sections = ArrayList<>();  // Depends on abstraction 'Section'.

    //Constructor

    public Homepage(News mainNews, List<Section> sections){

        this.mainNews = mainNews;
        this.sections = new ArrayList<>(sections);

    }
```

- **News:**



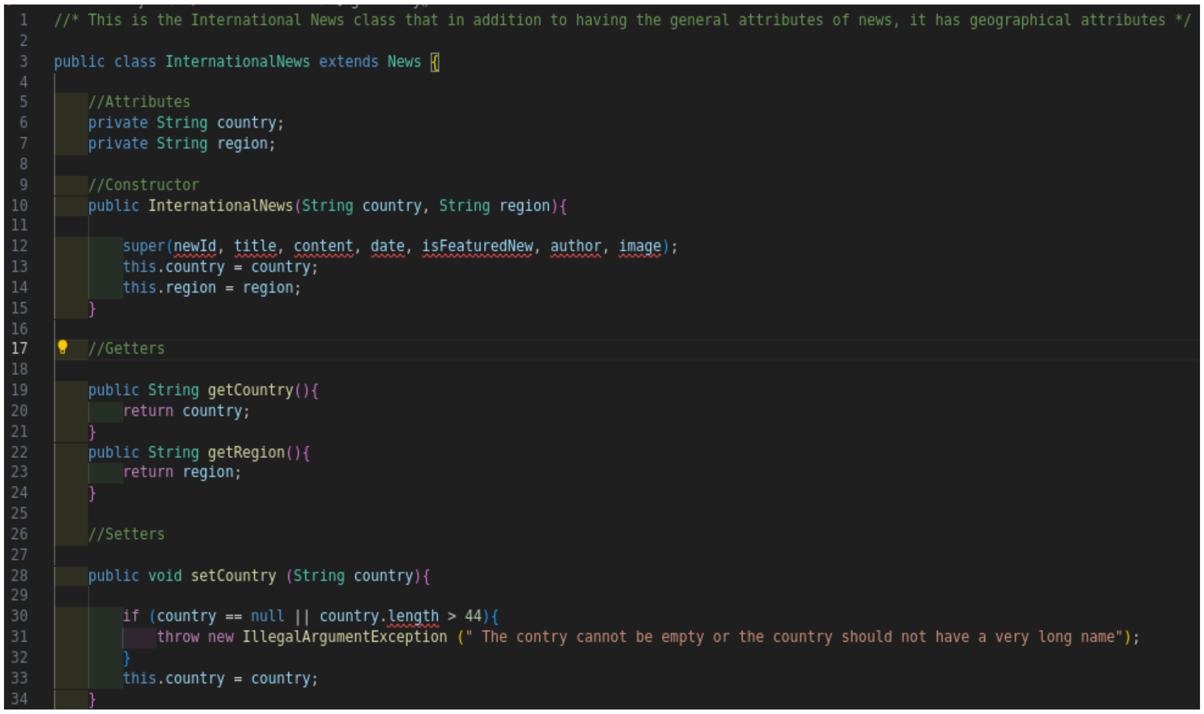
```java
//* This is the News class that defines the structure of the each news */

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class News {

    //Attributes
    private String newId;
    private String title;
    private String content;
    private LocalDate date;
    private boolean isFeaturedNew;
    private String author;
    private NewsImage image; // <-- Addition

    //Constructor
    public News (String newId, String title, String content, String dateString, String author, boolean isFeaturedNew, NewsImage image){

        this.newId = newId;
        this.title = title;
        this.content = content;
        this.date = validateDate(dateString);  //assigns to the date attribute the date of the parameter (string) previously validated
        this.author = author;
        this.isFeaturedNew = isFeaturedNew;
        this.image = image;

        // if the image is empty, then the default image is set
        if (image == null) {
            this.image = setIdNoticiaAsociada(id);
        }

    }
```

ImageNews:



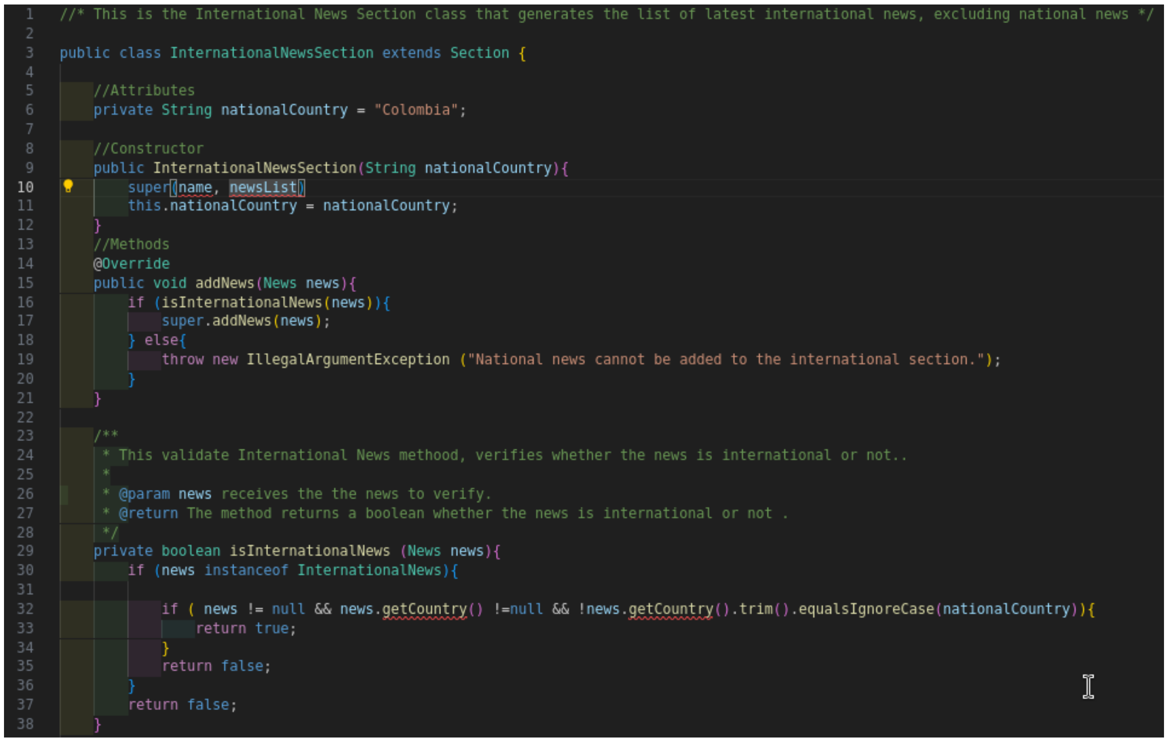
```java
//* This is the New Image class that Identify each image and establish its attributes */

public class NewsImage {

    //Attributes
    private String url;
    private String imageDescription;
    private String caption;
    private int width;
    private int height;
    private String newId;

    //Constructor
    public NewsImage(String url, String imageDescription, String caption, int width, int height, String newId){

        this.url = url;
        this.imageDescription = imageDescription;
        this.caption = caption;
        this.width = width;
        this.height = height;
        this.newId = newId; // <-- Addition, news image needs a news to exist but news does not need an image.
    }

    //Getters

    //Setters

    public void setUrl(String url){
        if (url == null){
            throw new IllegalArgumentException (" The url cannot be empty. ");
        }
        this.url = url;
    }
    public void setCaption(String caption){
        if (caption == null){
            throw new IllegalArgumentException ("The caption cannot be empty. ");
        }
        this.caption = caption;
    }
```

- **InternationalNews:**



```java
//* This is the International News class that in addition to having the general attributes of news, it has geographical attributes */

public class InternationalNews extends News {

    //Attributes
    private String country;
    private String region;

    //Constructor
    public InternationalNews(String country, String region){

        super(newId, title, content, date, isFeaturedNew, author, image);
        this.country = country;
        this.region = region;
    }

    //Getters

    public String getCountry(){
        return country;
    }
    public String getRegion(){
        return region;
    }

    //Setters

    public void setCountry (String country){

        if (country == null || country.length > 44){
            throw new IllegalArgumentException (" The contry cannot be empty or the country should not have a very long name");
        }
        this.country = country;
    }
```

- **InternationalNewsSection:**



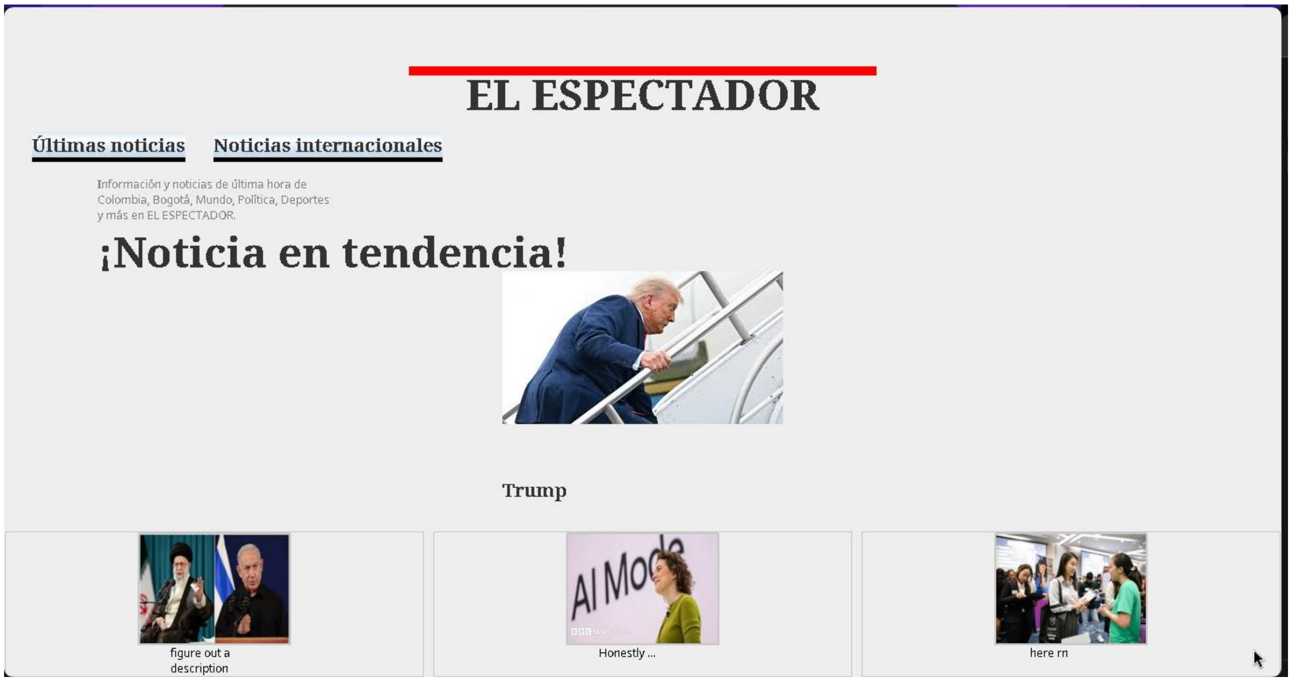
```java
//* This is the International News Section class that generates the list of latest international news, excluding national news */

public class InternationalNewsSection extends Section {

    //Attributes
    private String nationalCountry = "Colombia";

    //Constructor
    public InternationalNewsSection(String nationalCountry){
        super(name, newsList)
        this.nationalCountry = nationalCountry;
    }
    //Methods
    @Override
    public void addNews(News news){
        if (isInternationalNews(news)){
            super.addNews(news);
        } else{
            throw new IllegalArgumentException ("National news cannot be added to the international section.");
        }
    }

    /**
     * This validate International News method, verifies whether the news is international or not..
     *
     * @param news receives the the news to verify.
     * @return The method returns a boolean whether the news is international or not .
     */
    private boolean isInternationalNews (News news){
        if (news instanceof InternationalNews){

            if ( news != null && news.getCountry() !=null && !news.getCountry().trim().equalsIgnoreCase(nationalCountry)){
                return true;
            }
            return false;
        }
        return false;
    }
```

4 Workshop:

4.1 Revisiting Layers and Design:

Our class diagrams and documents still fit into this new final layer of our project.

4.2 Swing-based GUI Prototype:



The GUI was implemented using Java Swing to create a simple and functional interface for displaying news content. The main window is built using a `JFrame` as the top-level container, which holds the core UI components. A `JTextArea` as the primary display area for news content . Since no user input was required, the text area was set to read-only mode.

A basic navigation structure was added through a `JMenuBar` containing a "Sections" menu, though this could be expanded to include more categories. For user interaction, a "Swap section" button was implemented using `JButton` in the southern panel region. This button triggers a data reload by calling the controller's news-fetching method. The GUI follows a strict separation of concerns by delegating all business logic to the controller layer - when the refresh action occurs, the view simply requests updated data from the controller rather than accessing model classes directly.

The layout uses `BorderLayout` for clean organization, with the scrolling text area in the center and the refresh button at the bottom. All components were kept intentionally simple to focus on functionality over aesthetics, as specified in the requirements. The controller acts as an intermediary, receiving UI events and translating them into model operations, then returning formatted data for display. This design ensures the GUI remains decoupled from both the business logic and persistence layers, making future modifications or extensions easier to implement while maintaining the layered architecture.

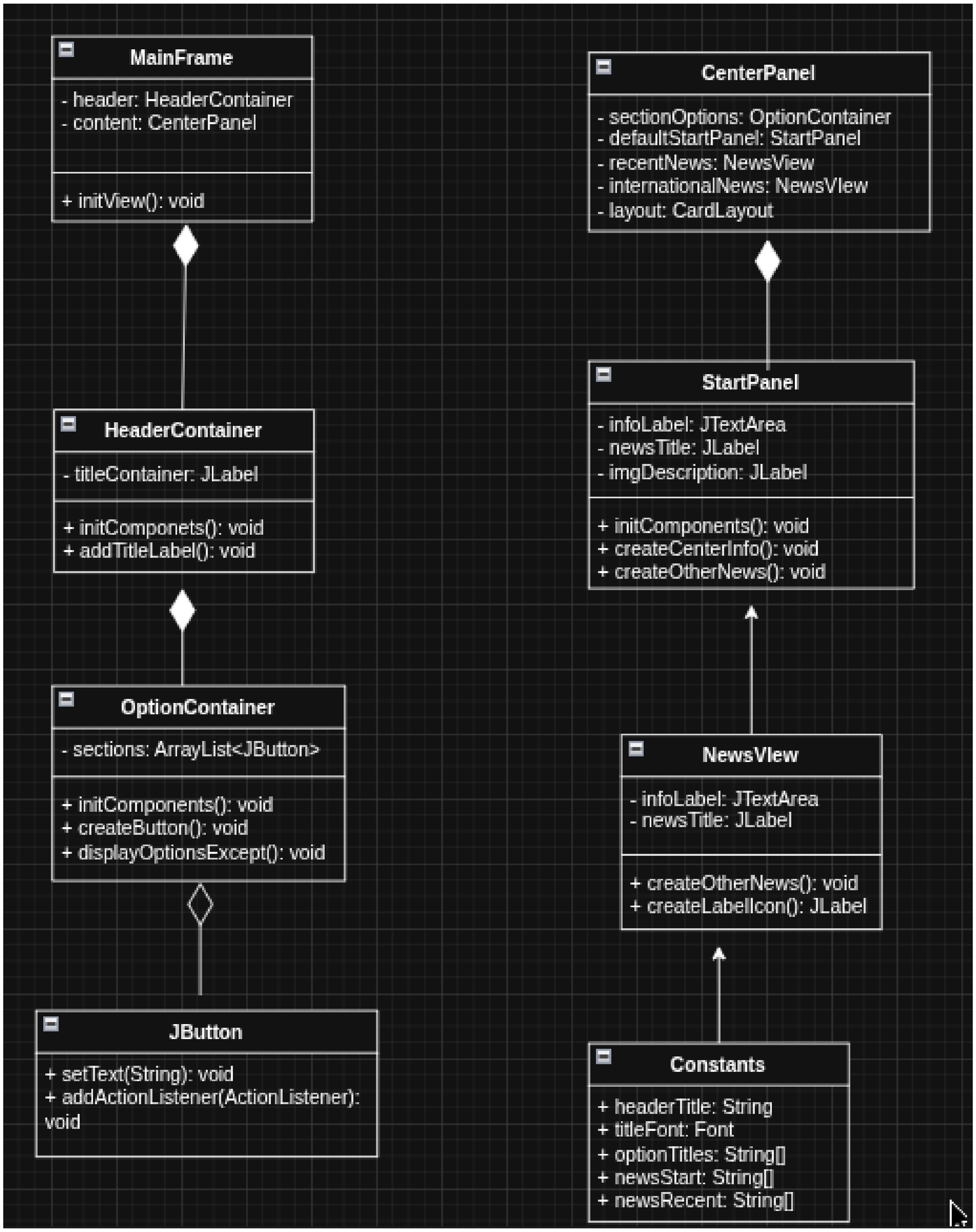
We wanted the user interface to be intuitive and easy to use and we have reached it.

4.3 File Storage:

Our project doesn't require validations or parameters to verify input, nor does it implement a database, as our main goal is to keep the user informed through an intuitive interface. Therefore, no implementation was implemented to process or save input data.

4.2 Documentation and Artifact Submission:

UML diagram demonstrating how each layer communicates in the final solution:



Presentation Layer (GUI)

MainFrame initializes and contains HeaderContainer and CenterPanel

CenterPanel manages different views (StartPanel, NewsView) using CardLayout

Example navigation flow:

```java
// In CenterPanel.java
sectionOptions.getOptions().get(i).addActionListener(e -> {
    layout.show(innerContainer, Constants.optionTitles[idx]);
});
```

Controller Layer

Mediates between GUI and business logic

Sample controller method:

```java
public String getNewsSummary(String section) {
    return model.getSection(section).getFormattedNews();
}
```

View Management

`CenterPanel.java`: Uses `CardLayout` to switch between:

```java
private NewsView recentNews = new NewsView(Constants.news_title[0],
Constants.news_info[0]);
private NewsView internationalNews = new NewsView(Constants.news_title[1],
Constants.news_info[1]);
```

News Display Components

`NewsView.java`: Creates news grids:

```java
JPanel auxContainer = new JPanel(new GridLayout(rows, cols, 10, 10));
for (int i = 0; i < n; i++) {
    // Creates image + description panels
}
```

Constants Configuration:

`Constants.java`: Centralizes UI settings:

```java
public static String[] optionTitles = { "Inicio", "Últimas noticias", "Noticias
internacionales" };
public static Font titleFont = new Font(Font.SERIF, Font.BOLD, 45);
```

Directory structure: (updated , adding user interface, and structured folders)

```
├── Workshop-4/
│   ├── bin/
│ │   ├── gui/
│ │ │     ├── CenterPanel.class
│ │ │     ├── Constants.class
│ │ │     ├── HeaderContainer.class
│ │ │     ├── MainFrame.class
│ │ │     ├── NewsView.class
│ │ │     ├── OptionContainer.class
│ │ │    └── StartPanel.class
│ │  ├── Images/
│ │ │     ├── test.jpeg
│ │ │     ├── testTwo.jpeg
│ │ │     ├── testThree.jpeg
│ │ │    └── ... (other image files)
│ │  ├── model/
│ │ │      ├── Homepage.class
│ │ │     ├── InternationalNews.class
│ │ │     ├── InternationalNewsSection.class
│ │ │     ├── News.class
│ │ │     ├── NewsImage.class
│ │ │    └── Section.class
│ │ └── presenter/
│ │         └── App.class
│ ├── lib/ ...
│ ├── src/
│ │  ├── gui/
│ │ │     ├── CenterPanel.java
│ │ │     ├── Constants.java
│ │ │     ├── HeaderContainer.java
│ │ │     ├── MainFrame.java
│ │ │     ├── NewsView.java
│ │ │     ├── OptionContainer.java
│ │ │    └── StartPanel.java
│ │  ├── Images/
│ │ │    (same image files as in bin/Images/)
│ │  ├── model/
│ │ │     ├── Homepage.java
│ │ │    ├── InternationalNews.java
│ │ │    ├── InternationalNewsSection.java
│ │ │    ├── News.java
│ │ │    ├── NewsImage.java
│ │ │    └── Section.java
│ │ └── presenter/
│ │        └── App.java
│ └── README.md
└── Workshop-4.pdf
```

- Mockup.

References:

-Viasure.

-Leanmind.