**NATIONAL UNIVERSITY OF SCIENCE AND**

**TECHNONLOGY**

**SMME (School of Mechanical & Manufacturing Engineering)**

**ME –15 B**

**FUNDAMENTAL OF PROGRAMMING**

**OBJECT ORIENTED PROGRAMMING**

**SUBMITTED TO:**

**SIR SAQIB NAZIR**

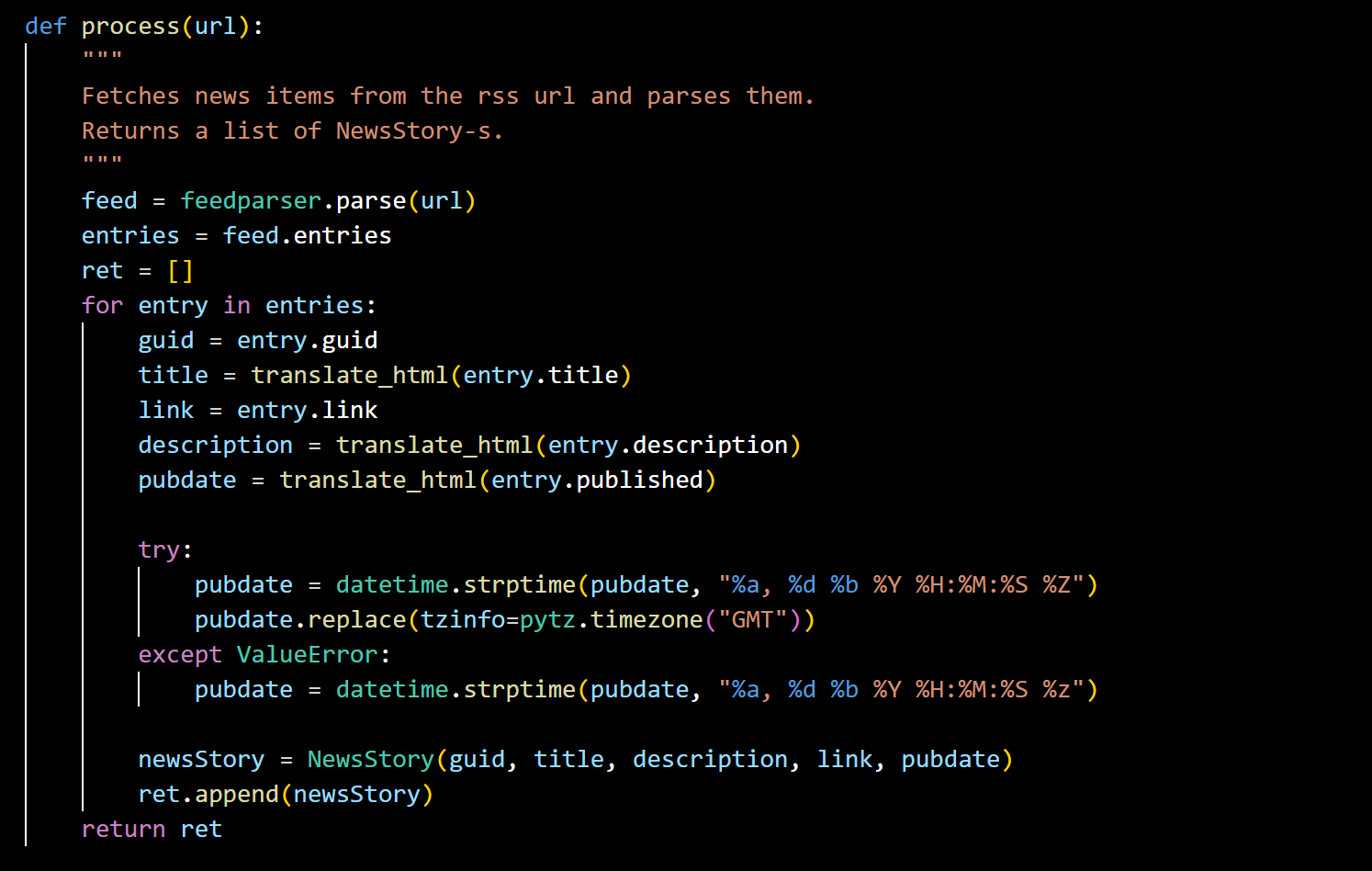
**SUBMITTED BY:**

**MOMINA HAQ(468932)**

**FAHAD ASIF(462895)**

**ABDUL MANAN (462449)**

**Process(url) Function :**

****

The **`process(url)`** function is responsible for retrieving news items from a given RSS feed URL. It utilizes the **`feedparser`** library to parse the XML data from the RSS feed. For each news item in the feed, it extracts essential information such as the GUID (a unique identifier), title, description, link, and publication date. These details are then encapsulated in instances of the **`NewsStory`** class, which is a custom class designed to represent a single news story. Finally, the function returns a list of these **`NewsStory`** instances, allowing further processing or analysis.

**NewsStory class:**

**A screen shot of a computer screen

Description automatically generated**

The **`NewsStory**` class represents a single news story. It serves as a container for storing various attributes associated with a news item, including the GUID, title, description, link, and publication date. The class is designed with data encapsulation in mind, providing methods such as **`get\_guid()`, `get\_title()`, `get\_description()`, `get\_link()`, and `get\_pubdate()`** to access these attributes individually. This encapsulation ensures that the internal state of a **`NewsStory`** instance remains protected and can only be accessed or modified through predefined methods, promoting code readability and maintainability.

**Trigger class:**

**A screen shot of a computer error

Description automatically generated**

The **`Trigger`** class acts as a base class for all types of triggers in the system. It defines a common interface method, **`evaluate()`,** which subclasses must implement. This method determines whether a given news story should trigger an alert based on the criteria defined by the specific trigger subclass. By providing a consistent interface, the **`Trigger`** class enables polymorphic behavior, allowing different types of triggers to be used interchangeably within the system.

**PhraseTrigger class:**

**A screen shot of a computer code

Description automatically generated**

The **`PhraseTrigger`** class is a subclass of **`Trigger`** that specializes in triggers based on specific phrases present in news stories. It initializes with a given phrase and provides a method, **`is\_phrase\_in()`,** to check if the phrase exists in a given text. This class serves as a foundational building block for more specific trigger subclasses like **`TitleTrigger`** and **`DescriptionTrigger`,** allowing triggers to be defined based on the textual content of news stories.

**TitleTrigger and DescriptionTrigger classes:**

A screen shot of a computer code

Description automatically generated

These subclasses of **`PhraseTrigger`** further specialize in triggers activated by specific phrases found in news titles and descriptions, respectively. They inherit the functionality of **`PhraseTrigger`** but provide more specific implementations of the **`evaluate()`** method tailored to their respective contexts. For example, **`TitleTrigger`** focuses on trigger phrases within news titles, while **`DescriptionTrigger`** targets those within descriptions.

**TimeTrigger class:**

**A screen shot of a computer code

Description automatically generated**

The **`TimeTrigger`** class represents time-based triggers, such as triggers activated before or after a specific time. It initializes with a time string, converting it to a **`datetime`** object with timezone information. This class enables triggers to be defined based on temporal conditions, such as triggering an alert if a news story is published before or after a certain date and time.

**BeforeTrigger and AfterTrigger classes:**

**A screen shot of a computer code

Description automatically generated**

These subclasses of `TimeTrigger` specialize in triggers activated before or after a specified time, respectively. They inherit the functionality of **`TimeTrigger`** and override its **`evaluate`** method to compare the publication time of a news story with the trigger time. This allows triggers to be defined based on temporal relationships, such as triggering an alert if a news story is published before a certain date or after a certain time.

**Composite Triggers: NotTrigger, AndTrigger, OrTrigger:**

A screen shot of a computer program

Description automatically generated

These classes represent composite triggers that combine multiple triggers using logical operations such as NOT, AND, and OR. Each subclass provides its implementation of the **`evaluate`** method, defining how constituent triggers are combined and evaluated to determine if a news story should trigger an alert. Composite triggers enable more complex trigger conditions to be defined by combining simpler triggers in various ways.

**Filter stories function:**

**A screen shot of a computer

Description automatically generated**

The **`filter\_stories`** function filters a list of news stories based on a list of trigger objects provided as input. It iterates through each news story and checks if any trigger in the trigger list evaluates to `True` for that story. Stories that satisfy at least one trigger condition are added to a filtered list, which is then returned. This function allows users to apply a set of predefined triggers to filter news stories based on specific criteria, such as textual content or publication date.

**Read trigger config function:**

**A computer screen shot of text

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

The **`read\_trigger\_config`** function reads a trigger configuration file and creates trigger objects based on the specified triggers and their arguments. It parses the configuration file, creating trigger instances for each defined trigger, and returns a list of these trigger objects. This function enables users to define custom trigger configurations outside of the main code, allowing for greater flexibility and customization of trigger behavior without modifying the core implementation.

**Main thread function:**

A computer code on a black background

Description automatically generated

The **`main\_thread`** function orchestrates the main execution of the program, responsible for fetching, filtering, and displaying news stories. It sets up a graphical user interface (GUI) using the **`mtTkinter`** library and continuously polls RSS feeds for news items. By applying the specified triggers, it filters news stories and displays the filtered stories in the GUI. Running indefinitely in a separate thread, it ensures the GUI remains responsive while continuously updating with new stories. This function serves as the central control hub of the program, coordinating the various components and ensuring smooth operation.

**OUTPUT:**

