**Bar Chart and Pie chart**

**Introduction:**

This code is a Java program that creates a bar chart visualization using the JFreeChart library.

The bar chart displays the user engagement rate and the total number of followers for different social media platforms like Facebook, YouTube, and Instagram.

The program takes input from the `User` class, which represents user data for each platform.

**BarChart Class:**

1.- The `BarChart` class extends the `ApplicationFrame` class from the JFreeChart library.

2.- The constructor of the `BarChart` class takes the following parameters:

3.- `applicationTitle`: The title of the application window.

4.- `chartTitle`: The title of the bar chart.

5.- `UniqueId`: A unique identifier that determines which platform's data to display (Facebook, YouTube, or Instagram).

6.- `user1`, `user2`, `user3`: Objects of the `User` class representing user data for different platforms.

**Bar Chart Creation:**

1.- The constructor checks the value of the `UniqueId` parameter to determine which platform's data to display.

2.- It creates a `JFreeChart` object using the `ChartFactory.createBarChart` method, passing in the chart title, axis labels, dataset, and other configuration options.

3.- The dataset is created using the `createDataset` methods, which are discussed in the next slide.

4.- The chart is then added to a `ChartPanel` and set as the content pane of the application window.

**Dataset Creation Methods:**

`createDatasetfacebook(User user2)`

1. - This method creates a `DefaultCategoryDataset` for Facebook.

2.- It adds the user engagement rate and total number of followers from the `user2` object to the dataset.

**createDatasetyoutube(User user1):**

1.- This method creates a `DefaultCategoryDataset` for YouTube.

2. - It adds the user engagement rate and total number of followers from the `user1` object to the dataset.

**createDatasetinstagram(User user3):**

1. - This method creates a `DefaultCategoryDataset` for Instagram.

2.- It adds the user engagement rate and total number of followers from the `user3` object to the dataset.

Usage:

1.- To use this program, you need to create instances of the `User` class with the appropriate data for each platform.

2.- Then, create an instance of the `BarChart` class and pass in the required parameters, including the `User` objects.

3.- The bar chart will be displayed in an application window.

Conclusion:

1- The code demonstrates the use of the JFreeChart library to create bar chart visualizations.

2.- It showcases object-oriented programming concepts, such as classes, objects, and methods.

3.- The program can be extended to include more social media platforms or additional data points.

**Class Report**

**Introduction**

- This code is a Java program that generates a PDF report containing data from different social media platforms like Facebook, Instagram, and YouTube.

- The program uses the Apache PDFBox library to create and manipulate PDF documents.

**Classes**

1. **report class**

- This class represents a report with properties like title, start date, end date, total engagement, and total posts.

- It has methods to set and get these properties, as well as methods to generate custom reports, save reports to files, add engagement, calculate average engagement, and print engagement details.

2**. Facebookdata class:**

- This class represents data related to Facebook, such as the number of posts, reactions, shares, and comments.

3.**Instagramdata class:**

- This class represents data related to Instagram, such as the number of shares, likes, followers, and comments.

4. **Youtubedata class**

- This class represents data related to YouTube, such as the number of views, likes, dislikes, and comments.

**GeneratePdf Method:**

1. Creating a PDF Document

- The program creates a new `PDDocument` object named `mydoc`.

2. Generating a Custom Report

- The title, start date, end date, and datasets (Facebookdata, Instagramdata, Youtubedata) for the report are defined.

- The `GeneratePdfData` method is called to print the report details.

3. **Adding Report Content to the PDF**

- A new `PDPage` object is created and added to the `mydoc`.

- A `PDPageContentStream` object is created to write content to the page.

- The font is loaded from a file using `PDType0Font.load`.

- The report title, start date, end date, and datasets are added to the PDF using `contentStream.showText`.

4**. Adding Data to the PDF**

- Instances of the `Facebookdata`, `Instagramdata`, and `Youtubedata` classes are created.

- The data from these instances is added to the PDF using `contentStream.showText`.

5. **Saving and Closing the PDF**

- The `mydoc` is saved to a file named "my\_doc.pdf" in the "E:/mypdf/" directory.

- The `mydoc` and `contentStream` objects are closed.

**Methods:**

1. `GeneratePdfData(String title, LocalDate startDate, LocalDate endDate, String... Datasets)`

- This method prints the report title, start date, end date, and datasets.

2. `saveReportToFile(String filename, String format)`

- This method saves the report to a file with the given filename and format.

3. `addEngagement(double engagement)`

- This method adds the given engagement value to the total engagement and increments the total posts.

4. `getAverageEngagement()`

- This method calculates and returns the average engagement by dividing the total engagement by the total posts.

5. `printEngagementDetails()`

- This method prints the total engagement, total posts, and average engagement.

6. `calculateAverageEngagement()`

- This method calculates and prints the average engagement, rounding it to two decimal places.

**Conclusion:**

- The program demonstrates the use of the Apache PDFBox library to generate PDF reports with data from different sources.

- It showcases object-oriented programming concepts, such as classes, objects, and methods.

- The program can be extended to include more data sources or customize the report layout and formatting.

Feel free to adapt and modify the explanation as per your requirements for the PowerPoint presentation.

**CLASS SOCIAL MEDIA ANALYTICS**

**Introduction:**

The `SocialMediaAnalytics` class serves as the entry point for the program. Its main function orchestrates the process of collecting user input, authenticating user credentials, and generating visual representations of data through bar charts and pie charts.

**Variable Declarations and Authentication:**

We are creating a variable string UniqueId to identify the platform the user is using.The unique id can only end with “y”-for youtube,”f”- for facebook, and “I”-for Instagram. Based on this input we will be getting the output of the required platform.

Then we are creating an object User Authentication class and a scanner object for taking the input.

**User Authentication:**

The user is prompted to enter their username, password, and a unique ID.The authenticateUser method from UserAuthentication class is called to verify the user's credentials.If authentication fails, an error message is printed.The finally block ensures that the scanner is closed regardless of whether an exception occurs or not.

**Data Visualization:**

For each social media platform (YouTube, Facebook, and Instagram), the program creates user objects with specific engagement metrics. These user objects represent individual social media profiles and encapsulate metrics such as likes, shares, comments, followers, etc.

The Bar chart displays user engagement rates and total numbers of followers. The Pie chart illustrates audience demographics (The country from which the audience are viewing). In addition to visualizations, the program generates a PDF report summarizing the analyzed data.

**PACKAGE AUTHENTICATION**

**CLASS USER AUTHENTICATION:**

This UserAuthentication class manages the authentication process for users accessing the social media analytics program.

1. Users Array:

This class contains a private 2D array `users` to store predefined usernames and passwords. Each row of the array represents a user, where the first element is the username and the second element is the password.

2. Constructor:

The constructor initializes the `users` array with predefined usernames and passwords. In this example, there are two users (`user1` and `user2`) with corresponding passwords (`password1` and `password2`).

3. authenticateUser Method:

This method takes two parameters: `username` and `password`, representing the user-provided credentials.

It iterates through the `users` array to find a match for the provided username.

If a matching username is found, it checks whether the provided password matches the stored password for that user.

If the password matches, it returns `true`, indicating successful authentication.

If the password does not match, it throws an `AuthenticationException` with the message "Incorrect password".

If no matching username is found, it throws an `AuthenticationException` with the message "User does not exist".

4. Exception Handling:

The method throws `AuthenticationException` in case of authentication failure, providing descriptive error messages to help identify the cause of failure.

Overall, this class encapsulates the authentication logic, allowing the main program to validate user credentials against predefined usernames and passwords stored within the `users` array.

**USER (ABSTRACT CLASS)**

**Attributes**:

uniqueId: A string representing the unique identifier of the user.

asiausers, europeusers, northamericausers, southamericausers, africausers: Doubles representing user counts in different regions.

**Abstract Methods:**

abstract double getfollowers(): This method is abstract, meaning it must be implemented by any subclass. It's meant to retrieve the number of followers.

abstract double userEngagement(): Another abstract method representing user engagement, to be implemented by subclasses.

**Youtube (Concrete Class, Inherits User):**

**Attributes:**

no\_of\_views, no\_of\_likes, no\_of\_dislikes, no\_of\_comments, total\_followers: Integers representing various metrics related to a YouTube user's activity.

**Constructors:**

Two constructors: one with basic user information and another with detailed user information including engagement metrics.

**Methods:**

userEngagement(int, int, int, int): Calculates user engagement based on views, likes, dislikes, and comments.

userEngagement(int, int, int): Overloaded method excluding dislikes.

userEngagement(int): Overloaded method with a minimum view threshold.

**Instagram (Concrete Class, Inherits User):**

**Attributes:**

no\_of\_shares, no\_of\_likes, total\_no\_of\_followers, no\_of\_comments: Integers representing various metrics related to an Instagram user's activity.

**Constructors:**

Similar to the Youtube class, with basic and detailed user information.

**Methods:**

userEngagement(): Calculates user engagement based on likes, comments, and shares.

userEngagement(int): Overloaded method with a minimum followers threshold.

**Facebook (Concrete Class, Inherits User):**

**Attributes:**

no\_of\_posts, no\_of\_reactions, no\_of\_shares, no\_of\_comments, total\_followers: Integers representing various metrics related to a Facebook user's activity.

**Constructors:**

Similar to the other classes, with basic and detailed user information.

**Methods:**

userEngagement(): Calculates user engagement based on reactions, comments, and shares.

userEngagement(double, double, double): Overloaded method allowing weighted engagement calculation.

userEngagement(int): Overloaded method with a minimum posts threshold.