IT 320
OOP REPORT

Authors:

- Jihene Gazzeh
- Jaouher El Mani
- Wala Marzouki
- Karim Ben Amar
- Nour Elloumi

Contents:

1-Data Source

This project focuses on the development of a comprehensive data analysis management system using Java, designed to incorporate core object-oriented programming (OOP) principles. Targeted at the financial domain, the system aims to track and evaluate the performance of foreign stocks within the forex market. By providing a user-friendly interface for uploading, storing, analyzing, and visualizing financial data, the system empowers users to make informed decisions.

2-System Architecture Overview

The system architecture follows a multi-layered design. It is structured into four interconnected layers: the User Layer, the Presentation Layer, the Business Logic Layer, and the Data Access Layer. Each layer is responsible for specific tasks, from user interaction to data processing and storage.

1. Architecture Layers

User Layer

The User Layer represents the end-user's interaction with the application via a web interface. Users can upload files in formats like CSV or XLSX and ask for file analysis.

Presentation Layer (Controller Layer)

Acts as the bridge between the user and the application’s core logic.

Components:

- FileUploadController:
  - ➢ Handles File Uploads: Manages the upload process, validates file types, and stores files in a predefined directory.
  - ➢ Processes Data: Triggers data parsing, cleaning, and analysis, including generating statistics and visualizations.

- Facilitates User Interaction: Serves as the bridge between users and the system, ensuring analysis results are displayed in an accessible format.

Business Logic Layer (Service Layer)

Contains the core functionality of the application.Components:

- DataAnalysisService.java
  - Provides methods to clean, normalize, and analyze data, including outlier removal and prediction using a regression model.
  - Handles file uploads and performs predictions based on price and date data.
- DataNormalizer.java
  - Implements normalization of data by scaling values between 0 and 1.
  - Extends the abstract class DataProcessor.
- DataProcessor.java
  - An abstract class serving as a blueprint for data processing tasks.
  - Defines the process method for subclasses to implement .
- FileParserService.java
  - Parses uploaded CSV or XLSX files to extract tabular data.
  - Supports file format validation and handles parsing errors gracefully.
- LinearRegressionModel.java
  - Implements linear regression for training and predicting future values based on historical price and date data.
  - Generates predictions for the next five days.

- OutlierRemover.java
  - Removes outliers from data using the interquartile range (IQR) method.
  - Extends DataProcessor for standardized processing.
- PredictionModel.java
  - An abstract class that defines the trainAndPredict method for making predictions.
  - Serves as the parent class for specific prediction models like LinearRegressionModel.
- StatisticsService.java
  - Provides statistical analysis, including descriptive stats, correlation, and regression.
  - Supports trend, seasonality, and residual decomposition in time series data.
- TimeSeriesService.java
  - Extracts and analyzes time series data, including dates and prices, from parsed files.
  - Supports additional data extraction and provides results in a structured format

Data Access Layer (Repository Layer):

Components:

- FileMetadataRepository.java
  - Defines a repository interface for managing FileMetadata objects in a MongoDB database.
  - Extends MongoRepository to provide CRUD operations for FileMetadata entities.

Data Access Layer (Model Layer):

Components:

- ❖ FileMetadata.java
  - ➢ Represents the metadata of a file, including its ID, name, content type, and size.
  - ➢ Annotated as a MongoDB document for storage in the file metadata collection, with getters and setters for all fields

2. System Workflow

1. **File Upload Handling:**

The controller accepts file uploads via a POST request to /upload. It checks if the file is empty or if the file type is valid (CSV or XLSX). It saves the uploaded file to a specified directory and stores its metadata in the database.

2. **Data Processing:**

   The uploaded file is parsed to generate a preview. Data is cleaned by removing rows with missing values and sorted. Price data is extracted for further analysis.

3. **Statistical Analysis:**
   - Descriptive statistics are calculated on the cleaned data.
   - Time series analysis is performed to identify trends and seasonality.
   - Correlation and regression analyses are conducted.
4. **Future Predictions:**

   The controller predicts future prices based on the cleaned data, using a time series model.

3-Relationship between classes and interfaces:

Polymorphism

- Definition: Polymorphism allows methods to perform different functions based on the object that is invoking them.
- Examples in code :
  - Overriding:

DataProcessor defines an abstract method process(List<Double> data), which is overridden in subclasses like DataNormalizer and OutlierRemover. Each subclass provides its own specific implementation of the process method.

- Dynamic Method Invocation:

The process method is called polymorphically through the DataProcessor reference in DataAnalysisService:
Java :
private final DataProcessor outlierRemover = new OutlierRemover();

private final DataProcessor normalizer = new DataNormalizer();

Abstraction

- Definition: Abstraction hides the implementation details and shows only the functionality to the user.
- Examples in code:
  - DataProcessor is an abstract class that defines the process method, but the details of its implementation are provided by concrete subclasses like DataNormalizer and OutlierRemover.
  - PredictionModel is an abstract class defining trainAndPredict(List<Double> prices, List<String> dates), which is implemented by the LinearRegressionModel class

Inheritance

- Definition: Inheritance allows a class to inherit properties and methods from another class.
- Examples in code:
  - DataNormalizer and OutlierRemover inherit from the DataProcessor abstract class.
  - LinearRegressionModel inherits from the PredictionModel abstract class.
  - The Spring framework's MongoRepository interface is extended by FileMetadataRepository to inherit methods for database operations.

Encapsulation

- Definition: Encapsulation restricts direct access to some of an object's components and provides controlled access via methods.
- Examples in code:
    - The FileMetadata class encapsulates its fields (id, fileName, contentType, size) by declaring them private and providing public getter and setter methods.
    - The encapsulation ensures that the fields of FileMetadata can only be accessed and modified through defined methods, safeguarding the integrity of the data.