CS320 Programming Languages
Assignment 1
(100 points+ 20 bonus)

Objectives and Skills:

In this assignment, you will use the Java regex API for data extraction to build a user application. This assignment will provide the opportunity to:

1. Learn how to use the Java regex API for extracting data.
2. Write a structured object-oriented application using Java.
3. Practice different data structures.

Tasks:

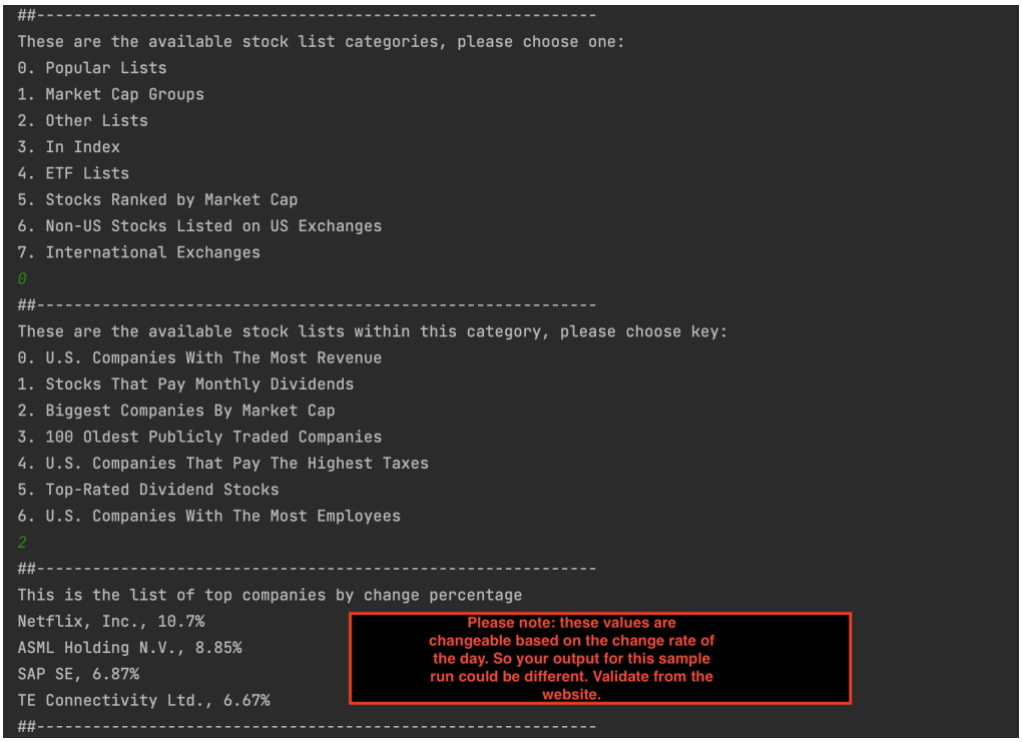
- Utilize the ***Java regex API*** to extract the stock list by different categories and provide insightful analysis using the information from the Stock analysis List.
- ***Follow the design recommendations*** to meet the ***application requirements*** (Refer to the Design Requirements section and the sample run below).
- Create the ***StockAnalyst.java*** class that implements the ***IStockAnalyst*** Interface.
- Develop the ***Client.java*** class, which should contain the main method. The Client class will be used to test your application's behavior, handle the interaction with the user, and replicate the sample run example. ***It should not include any business logic (implementation details of the application's core functionalities).***

What to Submit:

a. 1 zip file containing the following Java files:

- The ***IStockAnalyst*** interface class
- The ***StockAnalyst.java*** that implements the IStockAnalyst Interface
- Any ***helper methods or classes*** that support your implementation.
- Ensure ***well-documented code*** using inline comments for clarity.
- Please ***note in your submission if you handled the bonus case***.

Here is a sample run for the application to show how it works.



Sample run Description:

1. The application presents the user with a list of all stock categories available on the Stock Analysis website for selection.
2. The user selects a number corresponding to one of the stock categories.
3. The application displays a list of subcategories within the user-chosen top category.
4. The user selects a number corresponding to one of the existing subcategories to retrieve specific stock analysis for it.
5. The application generates a list of companies with the highest percentage change rate based on the count number provided by the user. For example, In the sample run above, the user asked for the top 4 companies.

Design recommendations:

It is recommended to use the design depicted in Figure 1 for the implementation. The IStockAnalyst Interface is provided below. If you follow this suggested design, you should create the **StockAnalyst class** that implements **the IStockAnalyst Interface**, along with the **Client.java** class for the application's main method.

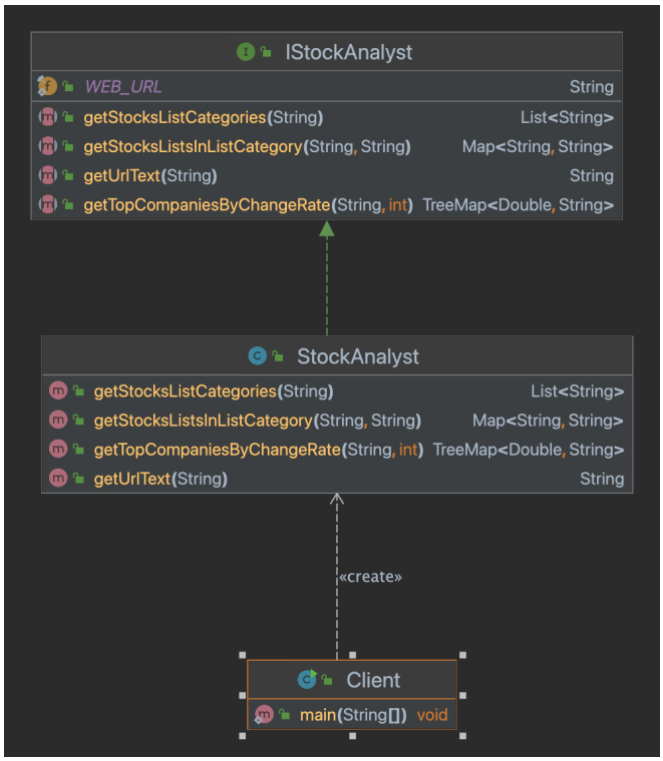


Figure 1: Recommended design for the Stock Analyst Application.

The IStockAnalyst Interface has four recommended methods to break down the logic:

1. getUrlText
2. getStocksListCategories
3. getStocksListInListCategory
4. getTopCompaniesByChangeRate

Each method is responsible for a part of the implementation steps to return the desired final output. The description, the set of parameters, and the return for each of the interface methods are as follows:

```
import java.util.*;

public interface IStockAnalyst {

    String WEB_URL = "https://stockanalysis.com/list/";

    /**
     * Method to get the text of a given web URL
     * @param url the web URL address
     * @return the web URL page text
     */
    String getUrlText(final String url);

    /**
     * Method to get the categories of stock lists
     * @param urlText the text of the page that has the stock list and their
categories
     * @return list of stock lists categories
```

```
     */
    List<String> getStocksListCategories(final String urlText);

    /**
     * Method to get the list of stocks within a stock list category
     * @param urlText the text of the page that has the stock list and their
categories
     * @param stockCategoryName the stock list category name
     * @return map that contains stock lists and their URLs. Key is stock
list name, and value is the URL
     * Example of a map entry <"Mega-Cap",
https://stockanalysis.com/list/mega-cap-stocks/>
     */
    Map<String, String> getStocksListsInListCategory(final String urlText,
final String stockCategoryName);

    // Ignore change percentages that are not numbers (e.g. "-")

    /**
     * Method to get top companies by change rate
     * @param urlText the text of the page that has the stock list
     * @param topCount how many companies to return
     * @return map that has the top companies and their change rate. Key is
the change rate and value is the company name
     */
    TreeMap<Double, String> getTopCompaniesByChangeRate(final String urlText,
int topCount);
}
```

You can download the interface from the Assignment view on Canvas.

Common assumptions

We operate using the default settings without applying filters to the stock list table, and pagination is unnecessary.

Special cases to count for in your implementation:

1. Sometimes, companies on the top list have the same percentage change rate. The method should permit the return of duplicates with the same percentage change rate as long as they belong to different companies. For example, when both Alibaba and NetEase share the exact same change rate % for the day, your return should include both if they are in the top list of the change rate.

No.	Symbol	Company Name	Market Cap	Stock Price	% Change	Revenue
53	BABA	Alibaba Group Holdi...	177.28B	74.02	7.85%	125.85B
198	NTES	NetEase, Inc.	59.46B	98.04	7.85%	13.99B
312	BIDU	Baidu, Inc.	38.21B	107.19	7.49%	18.26B
369	JD	JD.com, Inc.	30.55B	23.22	7.20%	147.73B
52	VZ	Verizon Communica...	177.54B	42.23	6.70%	134.10B

2. In the stock analysis table, there are instances where the % change is not listed and does not have a numerical value. In such cases, it is necessary to ignore change percentages that are not represented by numbers (e.g., "-").

32	DRI	Darden Restaurants, Inc.	187,384	161.66	-	19.30B
78	GPS	The Gap, Inc.	95,000	18.97	-	7.03B

3. In the stock analysis table for a specific stock sublist, the percentage change rate may appear in various column orders. Your method should consistently return the percentage change rate, regardless of its position in the table. For instance, the following examples demonstrate scenarios where the Change % is the fifth column and another where it is the sixth column in the table.

Special cases for 20 bonus points

Stock Ranked by Market Cap special case.

Most of the stocks ranked by market cap, such as Artificial Intelligence, Augmented Reality, Car Companies, or Metaverse, include two tables to provide insights about this market category. For all other categories, we have only one table. Handle the case where you can read each table and report back on the top percentage change rate for each table separately.

12 Stocks Table 1

No.	Symbol	Company Name	Market Cap	Stock Price	% Change	Revenue
1	MSFT	Microsoft Corporation	2,991.93B	402.56	0.92%	218.31B
2	TCEHY	Tencent Holdings Limited	317.47B	36.40	1.05%	84.52B
3	EA	Electronic Arts Inc.	37.08B	137.86	-0.94%	7.59B
4	TTWO	Take-Two Interactive Software,...	28.21B	165.90	0.45%	5.44B
5	SE	Sea Limited	21.72B	39.26	2.67%	12.90B
6	SKLZ	Skillz Inc.	113.03M	5.27	-2.41%	163.66M
7	AGAE	Allied Gaming & Entertainment...	43.47M	1.18	-5.60%	6.80M
8	FAZE	FaZe Holdings Inc.	27.04M	0.31	-2.50%	58.15M
9	SLE	Super League Enterprise, Inc.	7.20M	1.70	-	22.69M
10	MGAM	Mobile Global Esports Inc.	5.55M	0.27	-6.99%	-
11	VS	Versus Systems Inc.	5.52M	2.23	0.90%	454.56K
12	EBET	EBET, Inc.	3.36M	0.22	33.16%	39.18M

E-Sports ETFs Table 2

No.	Symbol	Fund Name	Stock Price	% Change	Assets
1	ESPO	VanEck Video Gaming and eSports E...	57.86	1.03%	248.62M
2	HERO	Global X Video Games & Esports ETF	19.78	-	129.06M
3	NERD	Roundhill Video Games ETF	15.37	-0.16%	20.83M

Grading Rubric

Follow recommended Design and Implementation Details (30 points) a. **Interface Class (5 points):** • *Requirements:* Presence of the IStockAnalyst interface class or equivalent. • *Assessment Criteria:* The interface includes relevant methods related to stock analysis. b. **Class Implementation (15 points):** • *Requirements:* StockAnalyst.java class that implements the IStockAnalyst Interface. • *Assessment Criteria:* Implementation correctly follows the interface. Class includes necessary methods and functionalities utilizing Java API regex as per assignment requirements. c. **Client Class (10 points):** • *Requirements:* Client.java class with a main method to interact with the user without including implementation details of core functionalities. • *Assessment Criteria:* The client class appropriately handles user interaction. No inclusion of business logic in the client class.	
Logic Breakdown and Helper Methods (30 points): • *Requirements:* Logic breakdown into standalone and independent methods. Any necessary helper methods or classes supporting the implementation are provided. • *Assessment Criteria:* o Helper methods serve their intended purpose and handle specific aspects of the application logic. o Effective logic breakdown to reduce code redundancy. o Code modularization enhances maintainability. o Well-organized and appropriately utilized. o Accurate Data Structure Usage includes proper selection and utilization of data structures based on requirements and efficient handling of data. o Accurate use of Regex Java API as needed.	
Return Accuracy and Complete Requirements Deliverables (30 points): • *Requirements:* Accurate return of results as per specifications. Complete implementation of all requirements. • *Assessment Criteria:* o Correct and complete implementation of requirements and expected return. o Output aligns with assignment requirements. o All necessary files, including interface, class implementation, and helpers, are present.	
Clean Code and Documentation (10 points): • *Requirements:* Well-documented code using inline comments. • *Assessment Criteria:* Comments provide clarity on code functionality. Code documentation enhances understanding.	
Handling Bonus Case (20 points): • *Requirements:* Optional handling of bonus cases as indicated in the requirements. • *Assessment Criteria*: Effective handling of bonus cases with accurate returns, if attempted.	
Total	100 +20

Extra information

1. What is TreeMap in Java?
2. Example of the HTML structure with two rows, each containing a 'Change %' value

```
<tr class="svelte-132bklf">
    <td class="svelte-132bklf">491</td>
    <td class="sym svelte-132bklf"><!-- HTML_TAG_START --><a
href="/stocks/avtr/">AVTR</a><!-- HTML_TAG_END --></td>
    <td class="slw svelte-132bklf">Avantor, Inc.</td>
    <td class="tr svelte-132bklf">7.04B</td>
    <td class="svelte-132bklf">21.58</td>
    <td class="svelte-132bklf" data-svelte-h="svelte-17887i7">-</td>
    <td class="svelte-132bklf">14.60B</td>
 </tr>
 <tr class="svelte-132bklf">
    <td class="svelte-132bklf">492</td>
    <td class="sym svelte-132bklf"><!-- HTML_TAG_START --><a
href="/stocks/clx/">CLX</a><!-- HTML_TAG_END --></td>
    <td class="slw svelte-132bklf">The Clorox Company</td>
    <td class="tr svelte-132bklf">7.04B</td>
    <td class="svelte-132bklf">142.22</td>
    <td class="rr svelte-132bklf">-0.79%</td>
    <td class="svelte-132bklf">17.64B</td>
</tr>
<tr class="svelte-132bklf"><td class="svelte-132bklf">493</td><td class="sym
svelte-132bklf"><!-- HTML_TAG_START --><a href="/stocks/post/">POST</a><!--
HTML_TAG_END
```