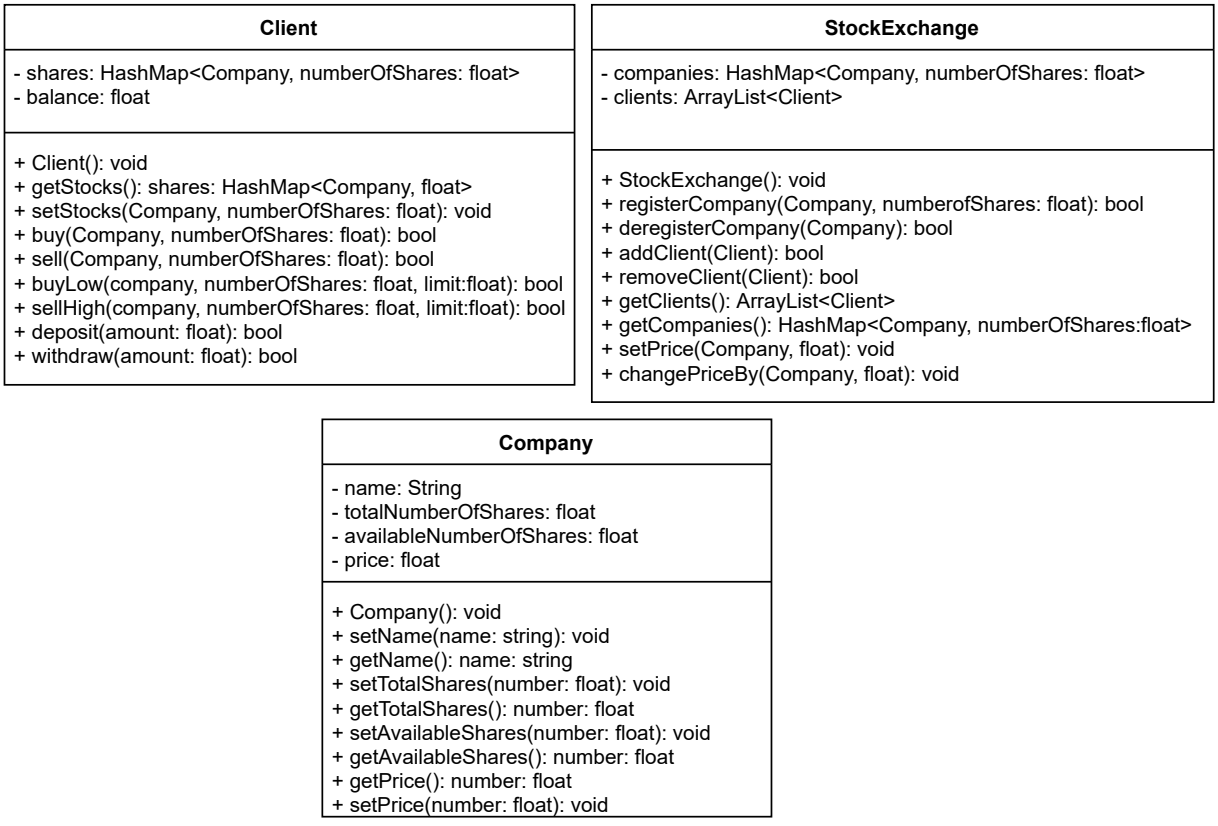
Developing a Concurrent Program Using Java
Coursework Assignment (2020-21)

Module:	Concurrency.
Module code:	CS-210.
Module coordinator:	Dr Alma Rahat (a.a.m.rahat@swansea.ac.uk).
Allocated marks:	**20**% of the total module marks.
Submission deadline:	**11:00 on 26 March 2021 (Friday)**.
Submission site:	Information to be released.

Scenario

A start-up company wants to develop a platform for trading shares. They have done some preliminary work, and captured their core requirements in the following (partial) UML class diagrams.



Essentially, they want to have *Clients* in their system who can buy and sell shares concurrently. This may involve transactions that happen either instantly at the current prices, or dynamically, i.e. when the price of the stock hits a predefined limit (e.g. *buyLow* and *sellHigh*), given that the client has sufficient balance. A *Company* simply consists of the name, the total number of shares, the number of shares available for sale, and the current price per share. The *StockExchange* will hold the information regarding who the clients are and which companies

are available on the platform. The prices are set and changed using *setPrice* and *changePriceBy* methods respectively within the *StockExchange*. The *changePriceBy* method essentially either adds or subtracts value to current price for a share of a company. Please note that the price of a share cannot be negative.

Tasks

In this assignment, your task is to design and implement an object oriented program (in **Java 8**) focusing on the correctness of the core sequential and concurrent functionalities: multiple clients buying and selling shares independently.

Please also include a short *headless* demonstration application that illustrates the functionalities.

> Please note that the test engineers at the company have already developed a range of automated tests in Java 8 using the above UML class diagrams. The company will use their test suite for validating a developer's solution. If they cannot test the behaviours as defined in the UML diagram, they will simply reject the solution and find another developer to provide a solution that works in the way they envisage the program to work.

Submission

The elements that we would require from you:

Code Submit your Java code file(s).

Documentation You are expected to write a short document (no more than a page; at least 11pt font size) where you provide the following:

- Diagram: A Labelled Transition System (LTS) diagram showing the key interactions in the system. You only need to show us the component process(es). You do not need to use LTSA or write FSP code to produce this; a simple diagram using any drawing software (e.g. `www.draw.io`) would suffice. We simply want to see your conceptual understanding, and encourage you to start coding concurrent programs in Java starting from a transition graph.
- Answer: What measures have you taken to protect against race conditions?
- Answer: What measures have you taken to avoid deadlock?

> **We will release specific instructions on how to submit these in due course.**

Assessed Learning Outcomes (ALOs)

The assessed learning outcomes of this assignment are as follows.

ALO_1 To examine a given specification, and identify subtle and complex problems within the context of a concurrent scenario.

ALO_2 To design and implement an object oriented concurrent program using Java (and associated language features).

Marking Criteria

Below we provide a table describing the mark allocation. Please note that the following allocation is indicative only, and may change following moderation.

Topic	Expectation	Indicative percentage of marks (%)
Problem analysis (ALO_1)	Correctly identify the concurrent processes in the given scenario and determine their roles.	10%
Conceptual understanding (ALO_1, ALO_2)	Infer concurrent issues (e.g. race condition and deadlock), and identify appropriate Java language features to deal with these.	30%
Basic object-oriented implementation (ALO_1, ALO_2)	There is a minimal implementation of all required classes, adhering to the specification. There is a basic *headless* `main` program showcasing and testing functionalities. The code compiles and passes tests for core sequential functionalities.	20%
Concurrent implementation (ALO_2)	Appropriate use of threads. Passes the concurrent and stress tests, clearly demonstrating correct behaviour.	35%
Code quality (ALO_2)	The code is written following standard practices.	5%

You will be marked in each category based on the level at which you meet the expectations described in the table above. The level descriptions are given below.

Excellent	Very good	Good	Adequate	Unacceptable
$70-100\%$	$60-69\%$	$50-59\%$	$40-49\%$	$0-39\%$

Please note that we may hold vivas as part of the assessment process. If you are invited for a viva, you will be expected to use your submission on the system to demonstrate functionalities and answer questions.

This is an individual coursework, and please adhere to guidelines about academic misconduct which can be found in the following link:

`myuni.swansea.ac.uk/academic-life/academic-misconduct/`

Please note: The consequences of being found guilty of academic misconduct can be extremely serious and have a profound effect on your results or further progression. The penalties range from a written reprimand to cancellation of all of your marks for the level and **withdrawal** from the University.