CSCE 156/156H: Project Overview

Dr. Chris Bourke

Spring 2020

If you don't have the time to do it right, then you'll have to find the time to do it over.

1 Introduction

Over the course of this semester, you will incrementally build a substantial database-backed application in Java. In each phase of the project you will focus on a particular component, which will have *deliverables* that you must hand in by a certain date. These deliverables may include source code, non-trivial test cases, database schemas and a well-written technical design document.

Each phase builds upon prior phases and may also require updates and modifications to prior phases. It is important that you understand the entire scope of the project. You should read all of the assignments to get a better understanding of where the project will be going.

The iterative nature of this project means that it is important that you do not fall behind. In each phase it is also important that you have a good, well-thought design to make subsequent phases easier to design, extend and implement. Poor designs, bad implementations, bugs and broken code will mean subsequent phases of this project will suffer. Investing your time and resources upfront will minimize your *technical debt* and mitigate the need to update or refactor your design later on. Remember one of the Golden Rules of coding: only code that which you would not mind having to maintain.

2 Problem Statement

TBF[1] Financial is a financial services corporation that has hired you to design and implement a simple financial management system to replace their aging AS400 green-screen system

[1] Too Big to Fail

that they currently use. It will be your responsibility to design a new multi-tier system from scratch that is object-oriented, written in Java, and supports TBF's business model by implementing their business rules and providing the functionality as stated below.

TBF manages portfolios of various assets for their clients. The application that you will build will need to support 3 different kinds of assets.

- **Deposit Accounts** are FDIC (Federal Deposit Insurance Corporation) insured accounts like Savings Accounts, CDs (Certificates of Deposit), and MMAs (Money Market Accounts). Each Deposit Account has a balance (its value) and an APR (annual percentage rate).
- **Stocks** are publicly traded shares of a company, business or corporation such as Google (GOOG) or Berkshire Hathaway (BRK.B). The value of a stock asset in a portfolio is the total number of shares multiplied by the share price.
- **Private Investments** are investments in non-publicly traded business enterprises such as ownership in a closely held corporation, franchise or an investment property or other tangible asset. Instead of shares, a private investment consists of the asset's total worth (its value) and a *stake* in the investment (a percentage that is owned by the owner of the portfolio).

Each investment consists of various pieces of data that define their value, annual rate of return, and a measure of risk. Each asset also has a *unique* alpha-numeric code which identifies it (in the old AS400 system) and a label to describe what the asset is.

2.1 Risk Measures

Depending on the type of asset, there is a certain amount of *risk* associated with it. Risk measures the relative exposure an investment has–the probability that an investment's return will be different than expected. Risky investments may not return as much as expected or may even *lose* value. There are several ways to measure such volatility.

2.1.1 Beta Measure

The *beta* measure is a measure of the relative risk of stocks and other highly traded commodities. It is a measure of risk relative to a general market (usually the S&P 500). Let r_a be the *return* of an asset and let r_b be the return of the overall market. Then beta is measured as

$$\beta = \frac{\mathsf{Cov}(r_a, r_b)}{\mathsf{Var}(r_b)},$$

where Cov and Var are the covariance and variance respectively. Equivalently, beta can be measured as

$$\beta = \rho_{a,b}\left(\frac{\sigma_a}{\sigma_b}\right),$$

where $\rho_{a,b}$ is the *correlation* of the two returns and σ_a, σ_b are their respectively volatilities.

By definition, the beta measure of the benchmark market itself is always 1. In general, beta is unbounded (that is, $\beta \in (-\infty, \infty)$) but in practice it usually falls in the interval $[-1, 2]$. A low beta value ($\beta < 1$) can either indicate a non-volatile asset (such as a T-Bill) or a volatile investment not highly correlated with the market (commodities such as crude oil or gold). A high beta value ($\beta > 1$) indicates a volatile asset that generally moves with the market. For example, Google stock has a beta value of $\beta = 1.15$. A 1% increase/decrease in the benchmark market would generally correlate to a 1.15% increase/decrease in the value of Google stock.

Negative beta values mean that the asset is negatively correlated with the market. For example, $\beta = -1$ would mean that a 1% *decrease* in the market would correspond to a 1% *increase* in the asset's value. Gold typically has a negative beta measure as it is a hedge against inflation.

A higher absolute value of the beta measure indicates a riskier (more volatile) asset but provide an opportunity for higher returns (whether or not this is positive or negative doesn't matter–you can hedge investments in either direction).

Beta measures are widely available on any financial data website (Yahoo Finance, `http://finance.yahoo.com/` for example).

2.1.2 Omega Measure

There are many complex measures of risk for private investments that vary greatly depending on the type of investment. For this project, we'll introduce a simple measure that we'll call the *omega* measure[2].

In particular, the omega measure for a Private Investment has a *base* risk measure $b \in [0, 1]$ (zero being no risk, 1 being extremely risky) that is determined by a financial expert using proprietary formulas. The omega measure is further augmented by adding an *offset* o that is determined by the following function.

$$o = e^{(-125{,}500/v)}$$

where v is the *total* value of the private investment. Thus, the final omega measure of a private investment is

$$\omega = b + o$$

[2]This is not a real measure, it has been created for this project

2.1.3 Aggregate Risk Measure

For an individual portfolio the *aggregate* risk measure can be computed as follows. Suppose that the portfolio contains assets $a_1, \ldots, a_n$ with values $v_1, \ldots, v_n$. Let

$$V = \sum_{i=1}^{n} v_i$$

be the total value of the portfolio.

The aggregate risk measure is then the weighted sum of all assets with respect to their risk measure:

$$\mathcal{R} = \sum_{i=1}^{n} \varphi_i \left(\frac{v_i}{V}\right),$$

where

$$\varphi_i = \begin{cases} \beta_i & \text{if } a_i \text{ is a stock} \\ \omega_i & \text{if } a_i \text{ is a private investment} \\ 0 & \text{if } a_i \text{ is a deposit account} \end{cases}$$

The risk is zero for deposit accounts as they are FDIC insured.

2.2 Rates of Return

Assets also have various rates of return that depend on their type.

For deposit accounts, interest is compounded at some periodic interval (annually, monthly, daily). The usual estimate for an annual rate of return is to use the Annual Percentage Yield (APY) formula,

$$APY = e^{APR} - 1$$

(where APR is on the scale $[0, 1]$).

Stocks and Private investments have a *base rate* of return that measures how much the asset's value is expected to increase on an annual basis. In addition, both may pay quarterly *dividends* which contribute to the expected annual return on top of the base rate of return. Thus, the expected annual return of a stock or private investment is its base rate of return multiplied by its value plus the value of its 4 quarterly dividends.

For all types of assets, the *rate* of return is the expected annual return divided by its total value.

2.3 Brokers

Each portfolio consists of any number of assets that are managed by an SEC (Securities and Exchange Commission) certified broker. For tax purposes, one person exclusively owns each portfolio. The owner may also *optionally* designate a beneficiary of the portfolio who will receive ownership of the portfolio upon the owner's death. In addition, each portfolio also has per-asset annual fees and commissions associated with it.

There are two types of brokers: Expert brokers and Junior brokers. Portfolios managed by a Junior broker have a $75.00 per-asset annual fee and are assessed a commission equal to 1.25% of the total annual expected return of the portfolio. Portfolios managed by an expert broker have a $0 per-asset annual fee and a 3.75% commission rate. Each broker also has a unique alphanumeric SEC identification code. Annual commissions are paid based on the *return* of the asset, not its value.

Finally, each person in the system, regardless of role (owner, broker, beneficiary) is represented with the same basic data. Every person has an alphanumeric code that uniquely identifies them (in the old system), a last name, first name, and an address (street, city, state, zip, country). In addition, each person can be associated with any number of email addresses.

3 Project Outline

Over the course of this semester you will iteratively design an application to support OBF's business model. Development has been broken down into 6 phases:

- Phase I: Data Representation & Electronic Data Interchange (EDI) – in the first phase you will design and implement the objects that will form a basis for the system and create parsers to read data from flat files, generate instances of your objects and export them to an interchange format (XML and/or JSON).
- Phase II: Summary Report – In this phase you will further refine your objects and define relationships between them to generate a summary report that aggregates pieces of data together.
- Phase III: Database Design – This phase focuses on designing a relational database to model your objects and support your application
- Phase IV: Database Connectivity – You will refactor your code to load your objects to your database rather than from flat files
- Phase V: Database Persistence – You will implement and use an API to persist (save) data to your database.
- Phase VI: Sorted List ADT – In this phase you design and implement a sorted list ADT and integrate it into your application

A Examples

To illustrate how this system should work, we've provided several examples.

A.1 Private Investment

Suppose we have a private investment with the following values.

Name	Cluckin's restaurant chain
Quarterly Dividend	$35,000.00
Base Rate of Return	3.0%
Omega Measure (base)	0.32
Total Value	$1,212,500.00

The omega measure would be calculated as follows.

$$\omega = b + e^{(-125{,}500/v)} = 0.32 + e^{(-125{,}500/1212500.00)} = 1.22167$$

The expected annual return is the base rate of return times the value plus its four *quarterly* dividends:

$$.03 \cdot \$1,212,500.00 + 4 \cdot \$35,000.00 = \$176,375$$

Thus, the *rate* of return would be the annual return divided by the investment's value:

$$\frac{\$176,375}{\$1,212,500} = 0.14546391752 = 14.55\%$$

A.2 Stock

Suppose we have a stock asset with the following properties.

Quarterly Dividend	$2.75
Base Rate of Return	4.6%
Beta Measure	0.79
Share Price	$97.75

The risk measure is simply given by the beta measure, 0.79. Further, suppose that there are 50 shares on a particular portfolio. The expected annual return is the base rate of return times the value plus its dividends:

$$.046 \cdot \$97.75 \cdot 50 + 4 \cdot \$2.75 \cdot 50 = \$774.83$$

Thus, the *rate* of return would be the annual return divided by the investment's value:

$$\frac{\$774.83}{\$97.75 \cdot 50} = 0.15853299232 = 15.85\%$$

B Project Setup Requirements

To ensure that your projects will run on the webgrader you *must* adhere to the following requirements and procedures. Failure to do so may make it impossible to grade your project and you will not receive credit.

B.1 Setup Requirements (Eclipse)

- You *must* include a `readme.md` file at the root of your project with your name and contact info. If you choose to work in pairs, both names/contact info should be included.
- All data files *must* be included in a directory named `data` at the root of your project.
- Any external JAR libraries should be placed in a directory named `lib` at the root of your project. Your project should look something like the following. To add external

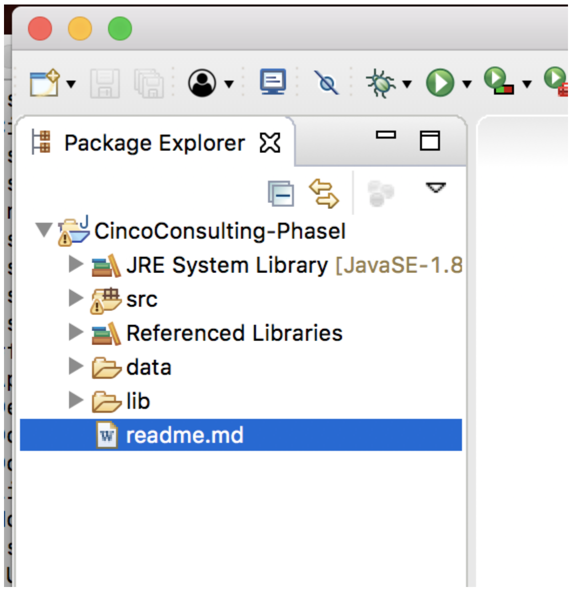


Figure 1: Your project setup should look something like this figure.

  JAR files to your project do the following.

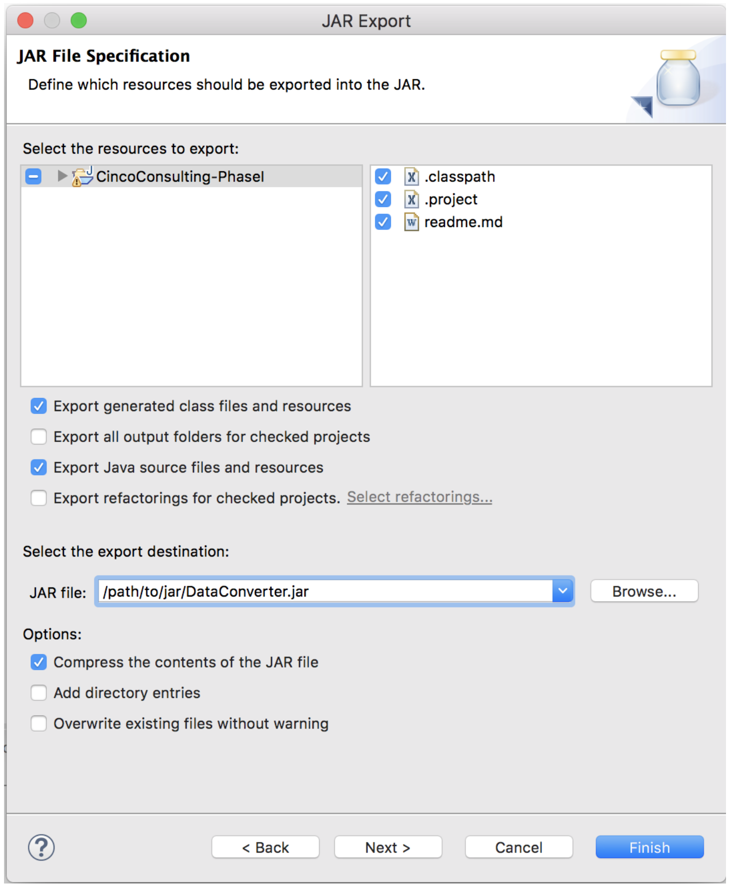
  1. Drag and drop your JAR file to this folder, be sure to select "copy files"
  2. Right click the new JAR file in your `lib` folder and select `Build Path` → `Add to Build Path`

- Your `main` method *must* be in the package and class name specified by the assignment.

B.2 Exporting Your Project (Eclipse)

To export your project for submission to webgrader, do the following:

1. Run your program at least once, this creates a "Launch Configuration" in Eclipse
2. Right click your project and select `Export...`
3. Under the Java folder, select `JAR file`, click `Next`
4. Be sure to check `Export Java source files and resources` and select the location where you want the JAR file saved. It should look something like the following.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [–] CincoConsulting-PhaseI
- [✓] .classpath
- [✓] .project
- [✓] readme.md

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [✓] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /path/to/jar/DataConverter.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

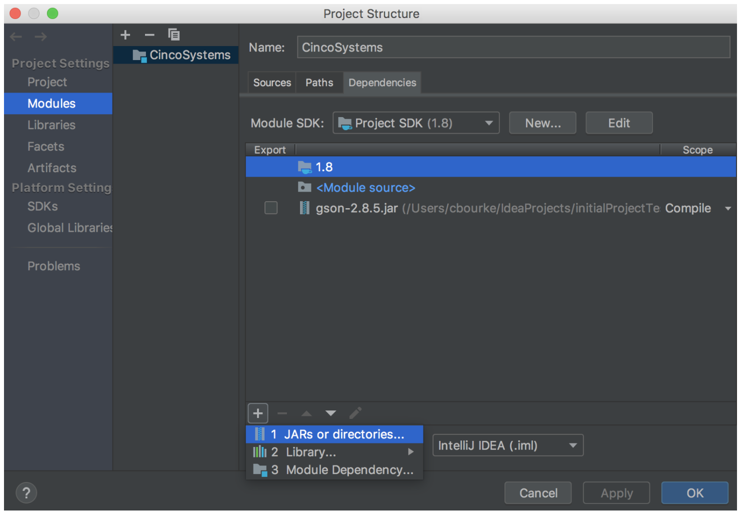
5. You can now click `Finish` and it should save the project as a JAR file which you will then turn in. The JAR file should contain all libraries, classes, source files and other data files necessary, but you should be sure it works by running the webgrader and addressing any issues.

B.3 Exporting Your Project (IntelliJ)

These instructions are for IntelliJ 2019.3. Steps for other versions may vary.

First, you may need to add an external JAR file(s) to your project. In IntelliJ:

1. Create a `lib` folder at the root of your object and drag/drop the JAR file to that directory.
2. Select `File` → `Project Structure...`
3. Select `Modules` at the left navigation
4. Change the tab to `Dependencies`
5. Click the plus button at the bottom left as in the image and select `JARs or directories...` and select the JAR(s) in your `lib` folder



Name: CincoSystems

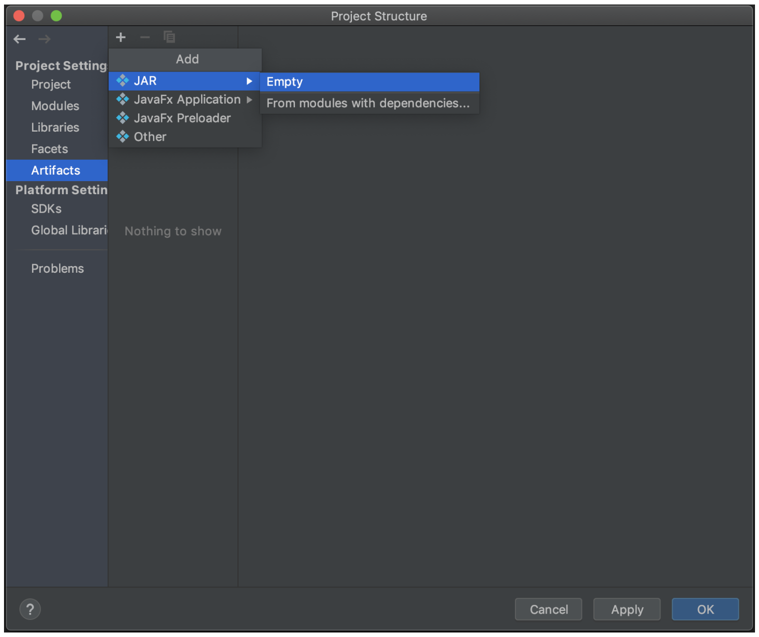
Module SDK: Project SDK (1.8)

1.8
<Module source>
gson-2.8.5.jar (/Users/cbourke/IdeaProjects/initialProjectTe Compile

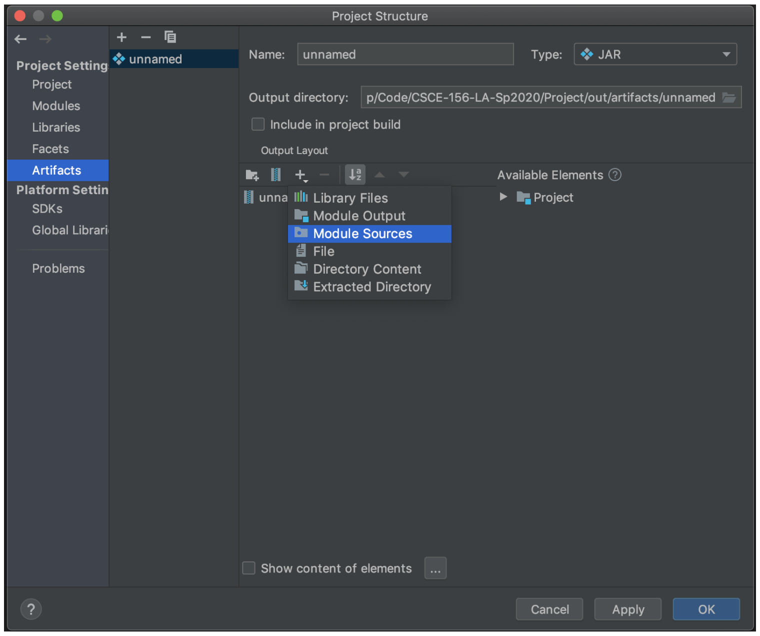
IntelliJ IDEA (.iml)

To export your project for submission to webgrader using IntelliJ, do the following:

1. Select `File` → `Project Structure...`
2. Select `Artifacts` at the left navigation
3. Click the plus button and select `JAR` → `Empty` as in the image.

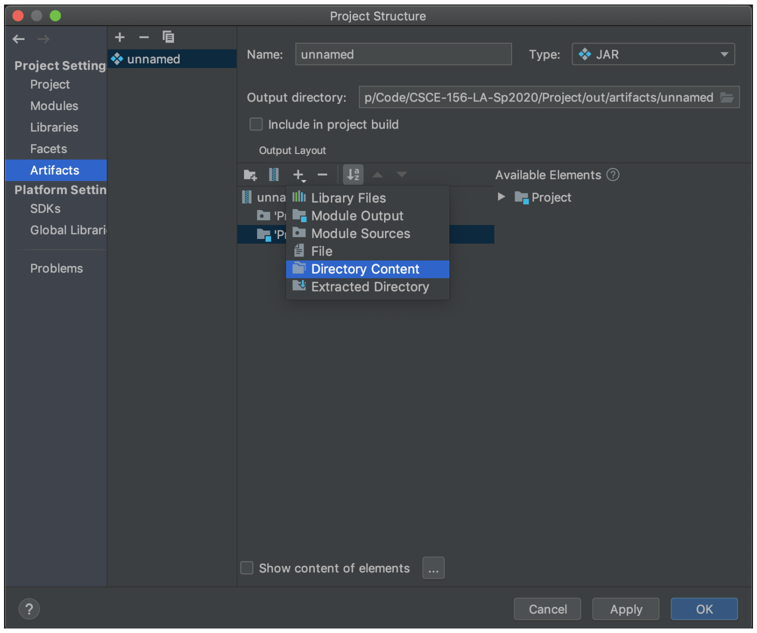


4. Click the plus button and select `Module Sources`



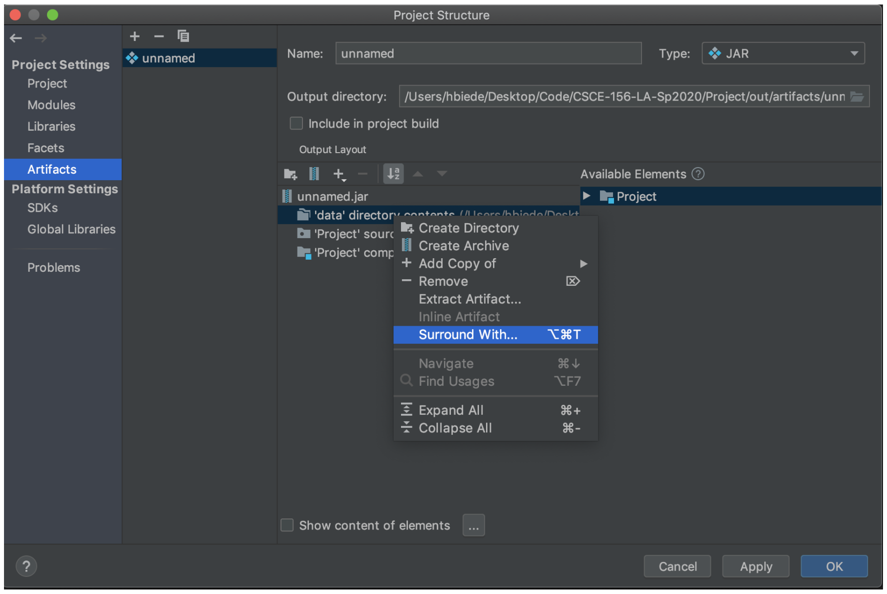
Click OK. Repeat this for `Module Output` as well.

5. Click the plus button again and select `Directory Content` and select the `data` folder.



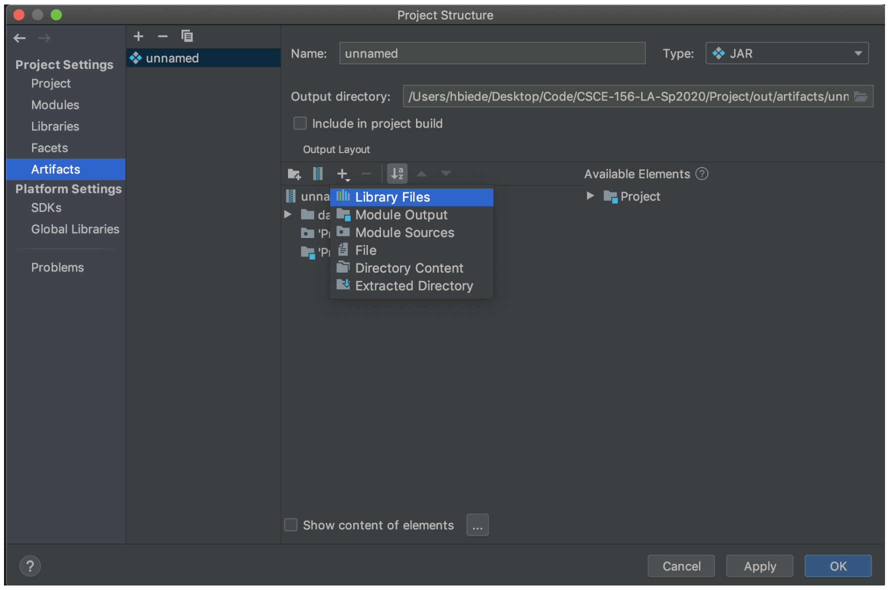
Name: unnamed
Type: JAR
Output directory: p/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/unnamed
Include in project build

6. Right click the result and select `Surround With...` and select `Directory` and select OK



Name: unnamed
Type: JAR
Output directory: /Users/hbiede/Desktop/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/unn
Include in project build

7. Click the plus again and select `Library Files`, select all, then press ok.

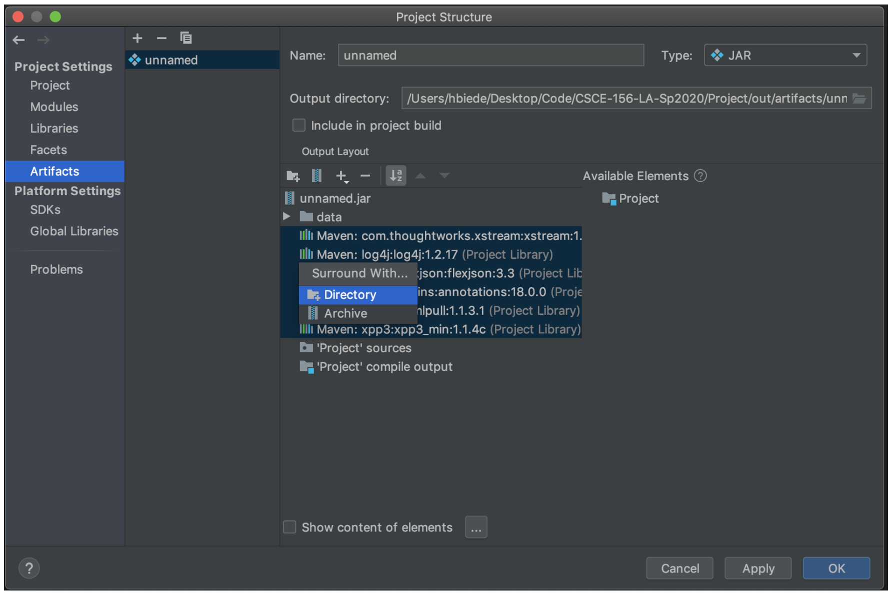


Name: unnamed Type: JAR

Output directory: /Users/hbiede/Desktop/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/unn

Include in project build

8. Select all the library jar files then right click and select `Surround with...` then `Directory` and name the directory `lib`



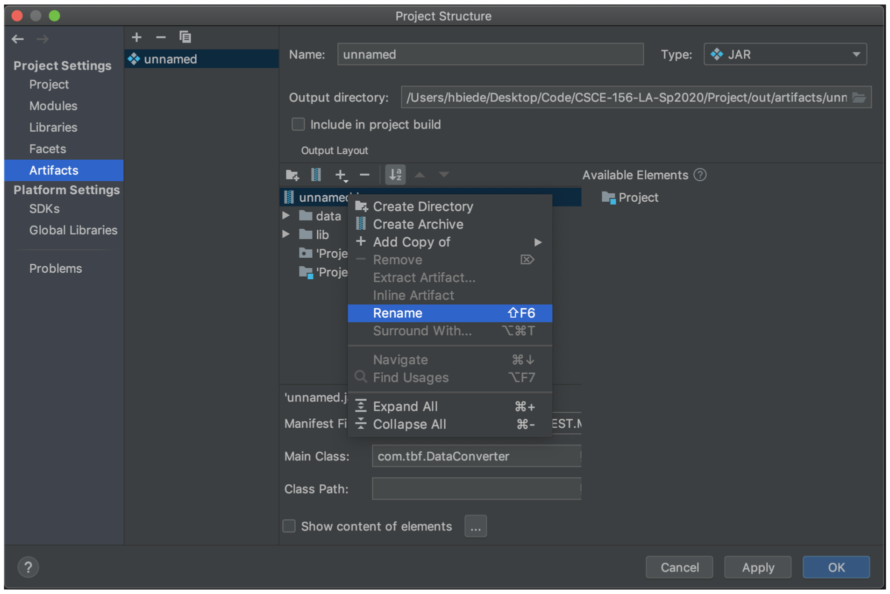
Name: unnamed Type: JAR

Output directory: /Users/hbiede/Desktop/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/unn

Include in project build

```
unnamed.jar
data
Maven: com.thoughtworks.xstream:xstream:1.
Maven: log4j:log4j:1.2.17 (Project Library)
:json:flexjson:3.3 (Project Lib
ins:annotations:18.0.0 (Proje
nlpull:1.1.3.1 (Project Library)
Maven: xpp3:xpp3_min:1.1.4c (Project Library)
'Project' sources
'Project' compile output
```

9. Right click `unnamed.jar`, select `Rename` and enter `Project.jar`



Name: unnamed Type: JAR

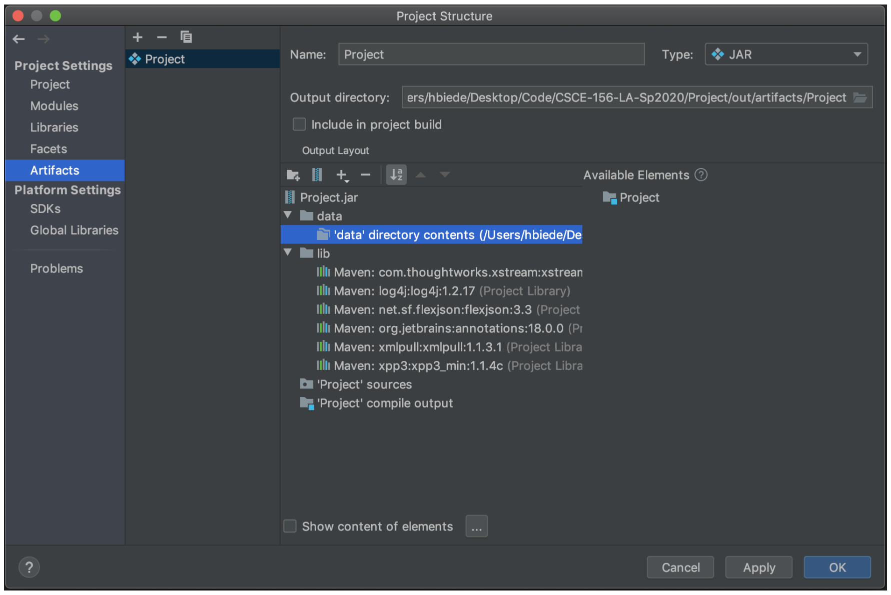
Output directory: /Users/hbiede/Desktop/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/unn

☐ Include in project build

Main Class: com.tbf.DataConverter

Class Path:

10. The result should look something like this:



Name: Project Type: JAR

Output directory: ers/hbiede/Desktop/Code/CSCE-156-LA-Sp2020/Project/out/artifacts/Project

☐ Include in project build

```
Project.jar
▼ data
    'data' directory contents (/Users/hbiede/De
▼ lib
    Maven: com.thoughtworks.xstream:xstrean
    Maven: log4j:log4j:1.2.17 (Project Library)
    Maven: net.sf.flexjson:flexjson:3.3 (Project
    Maven: org.jetbrains:annotations:18.0.0 (Pr
    Maven: xmlpull:xmlpull:1.1.3.1 (Project Libra
    Maven: xpp3:xpp3_min:1.1.4c (Project Libra
  'Project' sources
  'Project' compile output
```

11. Now to actually build your JAR file, select `Build` → `Build Artifacts` and select your jar file and the `build` option.

C Partner Policy & Procedure

You may choose to work alone or with a single partner (i.e. *pairs*) for each phase of the project. You may change partners between each phase if you choose. If you do choose to work in pairs, you must adhere to the following guidelines.

- All work must be the result of an equal collaborative effort by each partner. You may not simply partition the work between you.
- Turn in only one copy of the design document with both of your names on it
- You must turn in only one electronic copy under the login of the partner whose last name comes first alphabetically
- You must follow any additional policies regarding late passes or other items as described in the syllabus
- You are *highly encouraged* to use some sort of revision control system such as Git. However, you must ensure that your team's codebase and artifacts are not publicly accessible. Failure to do so will be considered a violation of the department's academic integrity policy.
- In order to work in pairs, you *must* join a group together in Canvas under the "Project Partners" group set. **Do not** create your own group set, use an existing one.