# Final Project Requirements

This is a team project. You will be working on the project in a team of two people.

**Note**: if there are students who are willingly refused to work on the project, let the instructors know by **14.12.2022 @18:00**.

**DEADLINE to submit FINAL PROJECT (to Blackboard): January 08th @23:59 (SHARP! No extensions)**

The project is about IKEA Products. You are going to develop a UI (console based) which will help the user to work on the dataset.

Explore the dataset (especially the datatypes of columns). Yes/No and True/False values must be stored as Boolean whereas id, item id, price and the dimensions seem to be numbers.

The main functionalities the program needs to provide are the following:

1. **List** all the entities **(20 points)**
    a. List randomly selected 20 entities
    b. List top 20 entities
    c. List bottom 20 entities
    d. **Note**: all lists should be followed by the number of entities listed.
    e. Once list method returns the selected entities the user should be able to choose to display
        i. **all the fields** of each entity.
        ii. only the **selected fields** of each entity.
2. **Sort** the entities **(30 points)**
    a. Based on **any field**
    b. In **any order** (i.e., ASC, DESC)
    c. Once the sort is over, the initial menu must be displayed to the user. If user needs to sort again, or list the already sorted entities, it should be possible as well.
3. **Search** entities based on any given field and value **(15 points)**
    a. **string fields** and values must be checked based on **contains** not exact equality.
    b. **non-string fields** and values must be checked based on **exact equality**.
4. **List column names (5 points)**
5. **Filter** entities **(30 points)**
    a. Based on any given field or set of fields and according to some rules:
    b. string fields
        i. contains
        ii. null (or missing)
    c. numeric fields
        i. equal (eq)
        ii. greater than (gt)
        iii. less than (lt)
        iv. greater and equal to (ge)
        v. less and equal to (le)
        vi. between (bt)

vii. null (or missing)

d. boolean
    i. equal (eq)

e. Examples:
    i. select all the products designed by *Carina Bengs* under category of *Beds*.
    ii. Select all the *bed frames* which are cheaper than *$1000*.
    iii. Select all the products which *cannot be sold online* and has recently been *updated in price*.

**Several important notes:**

1. To make is simple to use, make sure you take inputs of both fields and values in the same format as you list them.
2. Menu should be interactive, as the user might want to perform some sequence of requests.
    a. E.g., user might want to sort the entities based on the date (**request 1**) and pick top 20 entities to list (**request 2**).
3. To increase the reusability, after all the requests or sub-requests, return a collection of results so that you can print them, or use it in some subsequent requests.
4. You are free to use any notes or internet resources while developing the project, but **not misuse** them.
5. If ANY kind of cheating is detected, then **both sides (Cheater and Provider) will end up getting 0 pts**! The case will then be reported to the honor code committee.

**Technical aspects:**

1. You may use any topics we covered during the semester to maximum extend.
2. What might come handy are: collections, functional interface and stream api, exception handling, file reading, OOP principles (at least Encapsulation and method delegation), enums, etc.
3. Make sure you are not printing any stack trace of any exception, instead show t a nice error message formatted in a nice way.
4. Make sure you use methods wisely; each method should be responsible as specific as possible task.

**Prosses of development:**

1. You are expected to team up and discuss the project.
2. Share responsibilities.
3. Create a public github repository and put a nice description of your project in the README.md file.
4. Both team members will be committing to the same remote repository every time they have some feature ready and TESTED.
    a. Before committing, make sure you test the currently developed functionality and whether it suits to the entire application.
5. There should be at one or two commit and push performed by the team starting from the second week.
6. Once finished, write a report about the process, and submit it to the Blackboard Assignment Grader as well as the compressed (.zip) file of the project.
    a. The name of files:
        i. {TeamNo}_{Member1FullName}_{Member2FullName}_PROJECT
        ii. {TeamNo}_{Member1FullName}_{Member2FullName}_REPORT
    b. **Note**: Only source files and data files (used and generated) are to be included in the project file to submit. DO NOT SUBMIT the entire project folder if you are using any IDE. Or else this will be penalized.

**Final Report:**

1. A brief report about the responsibilities of each team member and their percentage (e.g., 50-50 or 40-60)
2. User manual, i.e., how the system must be used by the end user.
    a. Guidelines and snapshots would be enough per each request type.
3. Link to the public github repository
    a. Also add the link to the profiles of the team members.

**BONUS**:

1. There might be additional options for the user to use:
    a. Export a whole list of designers (exclude those values which start with a code (e.g., 002.756.74) these are not needed to be stored). Please also note that, each record might contain several designers as well (e.g., A Fredriksson/J Hultqvist/W Chong). They must be separated too + **10 points**.
    b. Products must be separated by each category and stored in separate .csv files. (i.e., **chairs.csv** will hold only the records that belong to the *chairs* category: about 481 records). However, in the exported category files, there should not be a column *category* anymore + **10 points**.
2. After search and filter requests we always print the results on the console.
    a. Giving an extra option to the user to EXPORT the result in the form of a report to a .csv file AFTER LISTING IT + **10 points**.
    b. **Note**: make sure you have proper headers (the names of the columns) and proper names for the files (hint: you may use the request info as the name of the file, since multiple reports should not overwrite)