EcoBike is an innovative player in the market of short-distance transportation solutions. They are expanding their business from a traditional shop to a worldwide presence on the internet. To enable that step, they are asking you to build a barebones software system to help them enable that transition. As they are aware that within the time you are given you can only design a prototype application, they are asking you to focus on their 3 core products, namely: folding bikes, electric bikes and so-called speedelecs, high-speed fully electric bikes. There is a clear aim to expand their online product portfolio in the short term, so do take that into consideration when designing your software system!

Specifically, EcoBike gave you the following file format and wants you to develop an application around it.

An example of such a file looks as follows:

(obviously, you are not allowed to change the file format in any way!)

FOLDING BIKE Brompton; 20; 6; 9283; TRUE; black; 1199

SPEEDELEC Peugeot; 45; 5426; TRUE; 8000; blue; 875

E-BIKE Gazelle; 49; 16455; TRUE; 16000; red; 1499

FOLDING BIKE Dahon; 16; 1; 11109; FALSE; white; 899

E-BIKE Koga; 48; 15488; TRUE; 21000; red; 1899

The order of the properties (brand, gears, battery capacity, ...) is fixed. The properties per folding bike, speedelec and e-bike are listed below:

A folding bike is characterized by:

- A brand
- The size of the wheels (in inch)
- The number of gears
- The weight of the bike (in grams)
- The availability of lights at front and back (TRUE/FALSE)
- A color
- The price

An speedelec is characterized by:

- A brand
- The maximum speed (in km/h)
- The weight of the bike (in grams)

- The availability of lights at front and back (TRUE/FALSE)
- The battery capacity (in mAh)
- A color
- The price

An e-bike is characterized by:

- A brand
- The maximum speed (in km/h)
- The weight of the e-bike (in grams)
- The availability of lights at front and back (TRUE/FALSE)
- The battery capacity (in mAh)
- A color
- The price

EcoBike asks you to design and implement a program that:

- Reads in the file ecobike.txt when starting the program.
- Write an equals() method for each class (except for the class that contains the main() method).
- To enable user interaction, please provide a command line interface with System.out.*. This interface should look like:

Please make your choice:

1 - Show the entire EcoBike catalog

2 – Add a new folding bike

3 – Add a new speedelec

4 – Add a new e-bike

5 – Find the first item of a particular brand

6 – Write to file

7 – Stop the program

Option 1

All products are shown on screen in the following format:

E-BIKE Koga with 15488 mAh battery and head/tail light.

Price: 1899 euros.

FOLDING BIKE Dahon with 1 gear(s) and no head/tail light.

Price: 899 euros.

SPEEDELEC Peugeot with 5426 mAh battery and head/tail light.

Price: 875 euros.

Option 2, 3 & 4

Through questions, you ask the user to fill in all the necessary fields that compose either a new folding bike (Option 2), a speedelec (Option 3) or an e- bike (Option 4). Use System.in

Option 5

Implement the search function, which allows you to find the products of a certain type (e-bikes, folding bikes and speedelecs) of a given brand and with the given parameters. The list of parameters can be any from empty (then the search works only by the name of the brand) to the full (when all parameters specific to this type of bike are set).

You can implement the search function yourself or using the "binary search", which is available in the class Collections.

Since sorting can be computationally expensive, you are given the option to use multithreading so that the application remains responsive during the search.

Make sure that you do not copy the data structure containing all the products to search for it (work on the "original" one) and make sure that new items cannot be added during sorting.

Option 6

The data should be written to file, in the same format so that the application can read in the file again!

Option 7

The application stops.

Some important things to consider for this assignment:

- Think about the usefulness of applying inheritance.
- When writing to the file ecobike.txt, the old version of the file should be overwritten. Maintain the format so that your program can read in the file again!
- The filename ecobike.txt should not be hardcoded in your Java program. Please make sure to let the user provide it when starting the program (either as an explicit question to the user or as a "command line input").
- Write unit tests.
- Your program should also work well, without exceptions.
- Take care to have a nice programming style, in other words, make use of code indentation, whitespaces, logical identifier names, etc.

Overview of the composition of your grade:

1. Implementation of all points of the task:
    - working capacity (if application is not working - final score = 0)
    - understood / misunderstood the task
    - percentage of completion

2. Structure and extensibility:
    - inheritance and polymorphism
    - visibility (public, private, …) of attributes/methods
    - separation of domain and infrastructure

3. Code readability:
    - length and complexity of methods
    - length of parameter lists
    - well-chosen identifier names
    - formatting
    - comments
    - ease of navigation

4. Friendly interface:
    - outputting the contents of catalogue (large - page by page)
    - input data:
        - correct prompt with an exact indication of the type of input data
        - type checking
        - value checking (price cannot be negative, etc.), if the value is not valid, the application does not crash but offers to enter in the correct form
    - informational confirmation of the fulfillment / non-fulfillment of the action
    - when exiting the application, give a warning if there is unsaved data
    - specifying the file name (info message about specifying the path / location in the directory)

5. Work with files:
    - reading: - message output if the format of the data in the file is not valid
    - record: - no new data - we are not saving
        - there is new data - add data to the file
        - saving data in a given form

6. Multithreading:
    - correct implementation
    - synchronization

7. Search:
    - own implementation
    - binary search

8. Tests:
    - correctness of testing
    - coverage volume