

Team Project – Program Description

EungGu Yun

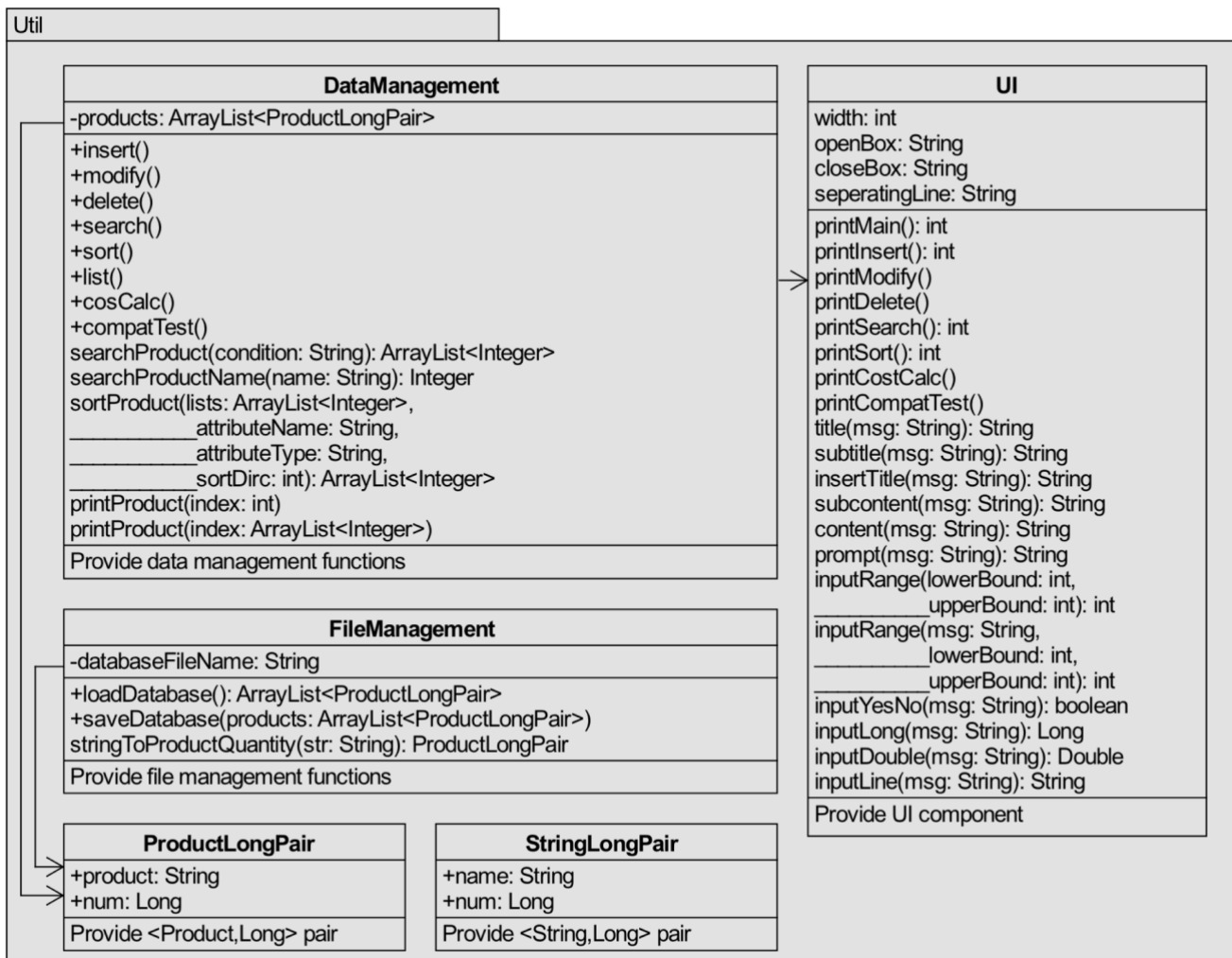
SeungHo Han

I. Outline

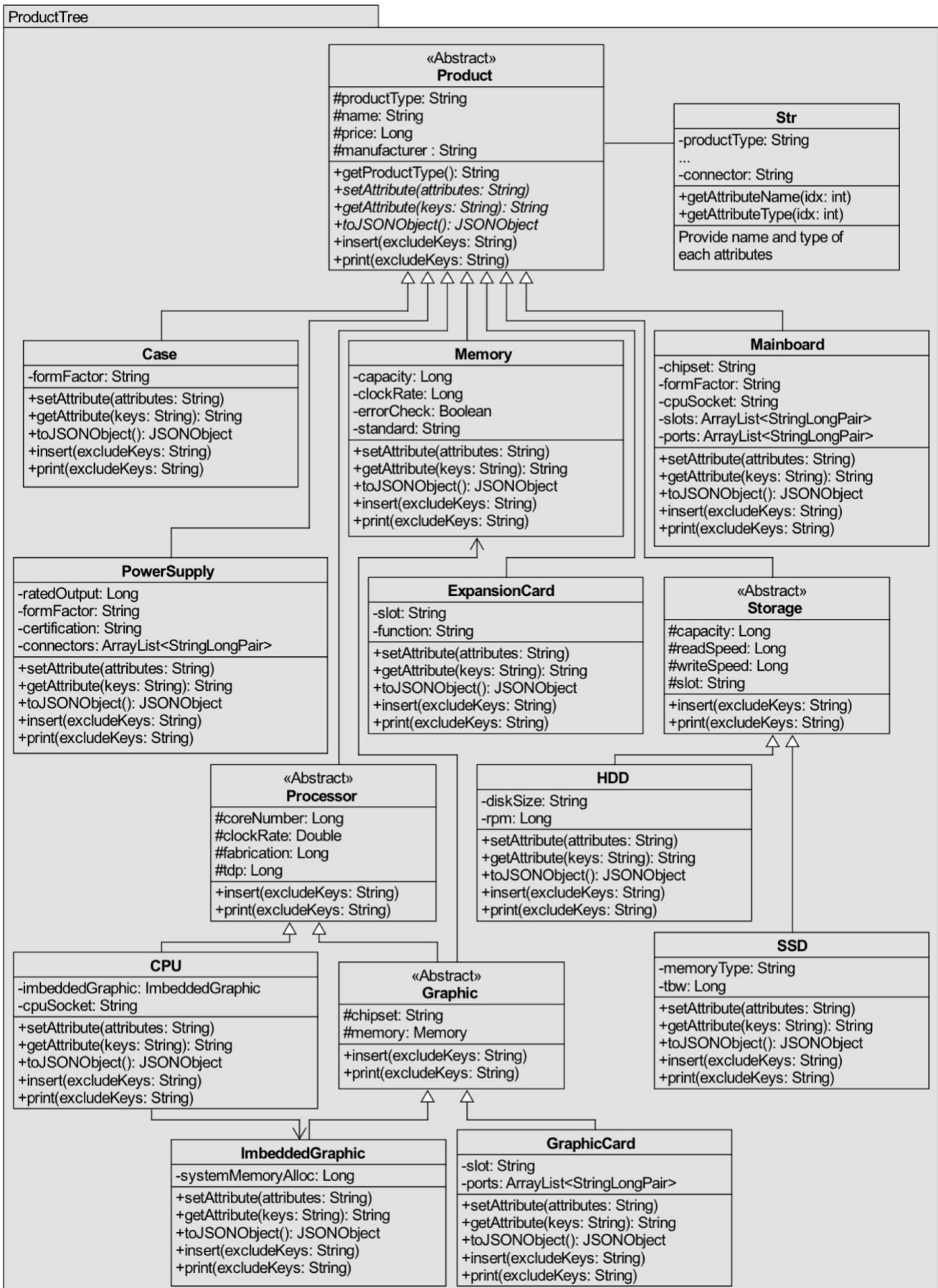
This program (Computer Hardware Manage System) is a program for managing computer parts such as CPU, Mainboard, Memory, etc. To represent these computer parts, we constructed **ProductTree** package. Also, we made **Util** package that provides functions for handling file, data of products, and UI. Within program, we choose JSON for data transmission. So, we add JSON library (fangyidong/json-simple) as **JSON** package to process JSON format.

II. UML

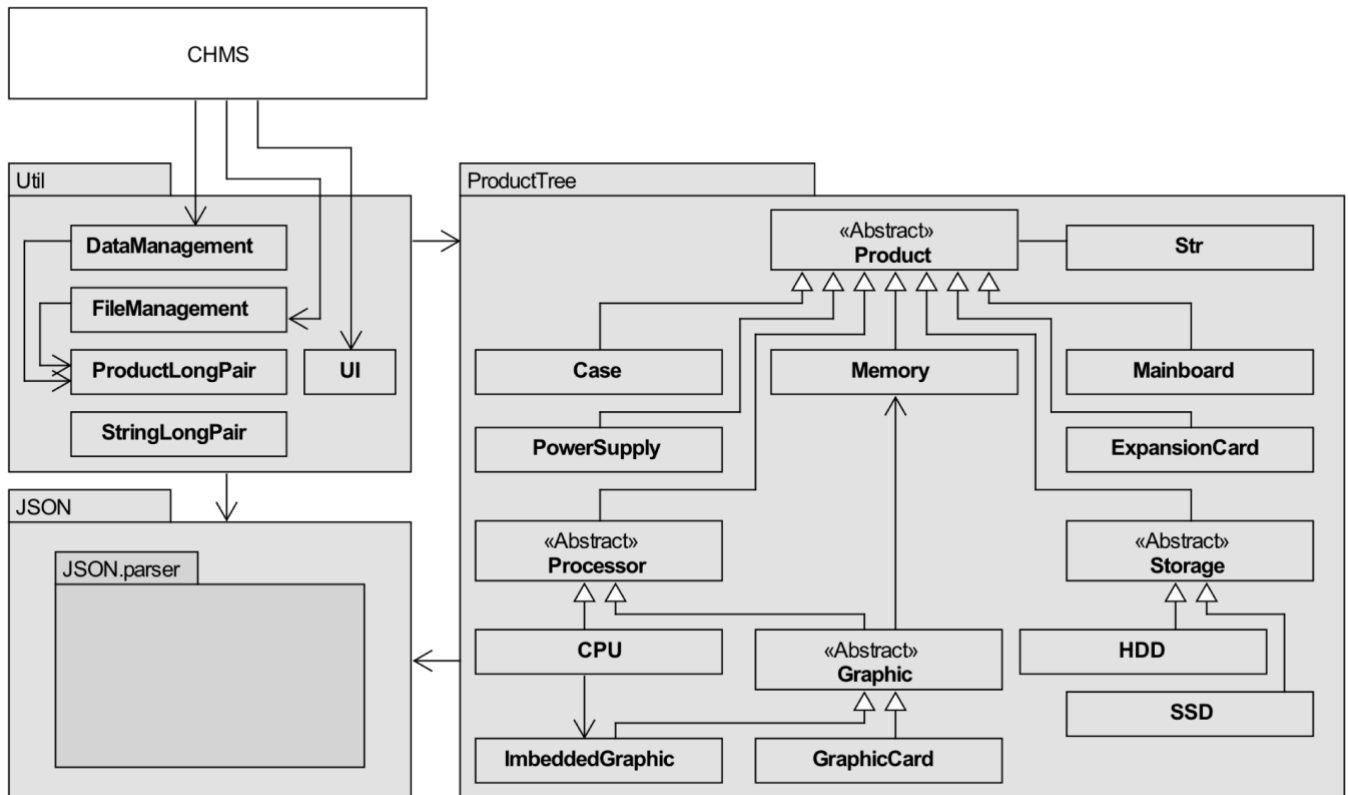
1. Util Package



2. ProductTree Package



3. Default Package



III. Program Structure

1. Util Package

Util package contains three types of classes. First, management classes – **FileManagement**, **DataManagement**. These classes provide functions to handle file and database. Second, UI component – **UI**. This class provide predefined message styles and console input validation. Third, data containers – **ProductLongPair**, **StringLongPair**. These classes provide containers which can store two different types of data.

① FileManagement Class

A. Variables

- String **databaseFileName**: Stores database file name.

B. Methods

- `ArrayList<ProductLongPair> loadDatabase():`

Reads database file and converts into array of Product object and quantity value pair. Database contains information of computer parts as JSON string per line. So, **loadDatabase()** reads each lines and uses **stringToProductQuantity()** to convert JSON string into pair of Product object and quantity value. Then adds these pairs to array. If there doesn't exist database file, it creates new one.

- **void saveDatabase**(ArrayList<ProductLongPair> products):

Gets data of computer parts and writes them into database file. Product object can be converted to JSON object by **toJSONObject()**. So, **saveDatabase()** combines Product

object and quantity value as one JSON object to make JSON String. Then writes these Strings line by line to file.

- ProductLongPair **stringToProductQuantity**(String str):

Converts JSON string into pair of Product object and quantity value. **stringToProductQuantity()** parses JSON string using **JSONParser** from JSON library. Then it makes Product object depending on the type of computer part and makes Long variable to store quantity. This methods is only used in the same class.

② **DataManagement** Class

A. Variables

- ArrayList<ProductLongPair> **products**: Stores computer parts informations and their quantities.

B. Methods

- void **insert**():

Provides user interface to insert computer part informations. It gets a product type from user and calls **insert()** in each product type class to get the appropriate information for each class. When user enters product name that already exist, it accumulates quantity of that product. Finally, it prints inserted product's information using **printProduct()**.

- void **modify**():

Provides user interface to modify computer part informations with product name. It gets a product name from user and search **products** array using **searchProductName()** with name. If a product exists, **modify()** calls **insert()** of that product to modify its attributes except name. Then it prints modified information of the product using **printProduct()**. When user enters product name which is not exist in **products** array, it prints error and returns to main menu.

- void **delete**():

Provides user interface to delete computer part informations with product name. It gets a product name from user and search products array using **searchProductName()** with name. If a product exists, **delete()** prints information of the product using **printProduct()** and asks if user really wants to delete it. If user enters y(yes), it deletes product from **products** array. If user enters n(no), it returns to main menu. When user enters product name which is not exist in **products** array, it prints error and returns to main menu.

- void **search**():

Provides user interface to search computer part informations with user-set condition. It gets an attribute name to search from user. And gets search mode – "Exist", "Match" or "Range". **search()** uses **searchProduct()** to search **products** array. Then it prints informations of the searched products using **printProduct()**.

- void **sort**():

Provides user interface to sort computer part informations with user-set condition. it gets an attribute name to sort from user. And gets sorting direction – "Ascending" or "Descending". **sort()** uses **searchProduct()** to search **products** array and **sortProduct()** to sort those products. Then it prints informations of the sorted products using

printProduct().

- void **list()**:

Provides user interface to print list of computer part informations. It simply calls **printProduct()** to print all products in **products** array.

- void **costCalc()**:

Provides user interface to calculate total cost of selected computer parts. It gets names from user and add products to calculation stand by list. When user ends adding products, it prints the products list and total cost.

- void **compatTest()**:

Provides user interface to check compatibility of selected computer parts. It gets each type of products from. Then, it checks compatibility of four sets – (CPU, Mainboard), (Graphic Card, Mainboard), (CPU+Graphic Card, Power Supply), (Mainboard, Case). Finally, it prints the compatibility results if each set is compatible, incompatible or warning.

- ArrayList<Integer> **searchProduct**(String condition):

Gets a search condition as JSON string and returns indexes of corresponding products in **products** array. "Exist" mode searches for the products array if the product has the corresponding attribute, "Match" mode searches if the product has the attribute and it matches, and "Range" mode searches if the product has the attribute and it is in the lower bound and upper bound.

- Integer **searchProductName**(String name):

Gets a name of product and returns index of corresponding product in **products** array. If product doesn't exist, it returns -1.

- ArrayList<Integer> **sortProduct**(ArrayList<Integer> lists, String attributeName, String attributeType, int sortDir):

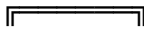
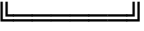
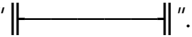
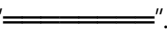
Gets an indexes of products to sort, a key attribute, and sort direction and returns sorted indexes of products. It can handle three attribute types – "String", "Long", and "Double". If an attribute type is "String", it sorts products in lexicographical order. It uses insertion sort algorithm to sort products.

- void **printProduct**(int index), void **printProduct**(ArrayList<Integer> index):

Gets an index or indexes of products to print information. It calls **print()** of each product object to print information of that product.

③ UI Class

A. Variables

- int **width**: Stores predefined UI box's width - 73.
- String **openBox**: Stores predefined UI box's upper part – "".
- String **closeBox**: Stores predefined UI box's lower part – "".
- String **seperatingLine**: Stores predefined UI box's separator part – "".
- String **inputEndLine**: Stores predefined UI input's lower part – "".

B. Methods

- int **printMain**(): Prints predefined Main menu UI and returns user input – Function to run.
- int **printInsert**(): Prints predefined Insert function UI and returns user input – Product type.
- void **printModify**(): Prints predefined Modify function UI.

- void **printDelete()**: Prints predefined Delete function UI.
- int **printSearch()**: Prints predefined Search function UI and returns user input – Attribute.
- int **printSort()**: Prints predefined Sort function UI and returns user input – Attribute.
- int **printCostCalc()**: Prints predefined Cost calculation function UI.
- int **printCompatTest()**: Prints predefined Compatibility test function UI.
- String **title**(String msg): Returns UI box's upper part with title.
- String **subtitle**(String msg): Returns UI box's inner part with subtitle.
- String **subcontent**(String msg): Returns UI box's inner part with content aligns to subtitle.
- String **content**(String msg): Returns UI box's inner part with content.
- String **prompt**(String msg): Returns UI box's outer part with prompt message.
- int **inputRange**(String msg, int lowerBound, int upperBound), int **inputRange**(int lowerBound, int upperBound): Prints message and returns int value within range.
- boolean **inputYesNo**(String msg): Prints message and returns boolean value.
- Long **inputLong**(String msg): Prints message and returns Long value.
- Double **inputDouble**(String msg): Prints message and returns Double value.
- String **inputLine**(String msg): Prints message and returns String value.

④ **ProductLongPair** Class

A. Variables

- Product **product**: Stores instance of Product object.
- Long **num**: Stores Long value – usually quantity of product.

⑤ **StringLongPair** Class

A. Variables

- String **name**: Stores String value – usually port's, slot's, or connector's name .
- Long **num**: Stores Long value – usually port's, slot's, or connector's number.

2. **ProductTree** Package

Product package contains classes that corresponding to types of computer parts – "CPU", "Mainboard", "Memory", "Graphic Card", "HDD", "SSD", "Power Supply", "Case", and "Expansion card". Every classes except **Str** class inherit **Product** abstract class and implement/override five methods from it – **insert()**, **print()**, **getAttribute()**, **setAttribute()**, **toJSONObject()**. These class inheritance structures allow us to handle different types of computer parts data using same methods.

Because classes' variables are corresponded to attributes of their product types, and methods are same as **Product**'s method except **getProductType()**, we omit them.

① **Product** Abstract Class

A. Variables

- String **productType**: Stores type of product – "CPU", "Mainboard", etc...
- String **name**: Stores name of product. Used as ID in the program.
- Long **price**: Stores price of product.
- String **manufacturer**: Stores manufacturer of product.

B. Methods

- String **getProductType()**: Returns type of product, **productType**.

- void **setAttribute**(String attributes): Gets attributes as JSON string and set variables.
- String **getAttribute**(String keys): Gets attribute names as JSON string and returns required attributes as JSON string.
- JSONObject **toJSONObject**(): Converts attributes into JSONObject and returns.
- void **insert**(String excludeKeys): Provides user interface to set attributes of product. It doesn't set attributes which are required not to set.
- void **print**(String excludeKeys): Prints attributes of product except attributes which are required not to print.

② **Processor** Abstract Class inherits **Product**

③ **Graphic** Abstract Class inherits **Processor**

④ **Storage** Abstract Class inherits **Product**

⑤ **CPU** Class inherits **Processor**

⑥ **Mainboard** Class inherits **Product**

⑦ **Memory** Class inherits **Product**

⑧ **GraphicCard** Class inherits **Graphic**

⑨ **ImbeddedGraphic** Class inherits **Graphic**

⑩ **HDD** Class inherits **Storage**

⑪ **SSD** Class inherits **Storage**

⑫ **PowerSupply** Class inherits **Product**

⑬ **Case** Class inherits **Product**

⑭ **ExpansionCard** Class inherits **Product**

⑮ **Str** Class

A. Variables: Variables are just String representation of its variable names.

B. Methods

- String **getAttributeName**(int idx): Returns an attribute name corresponding to the index.
- String **getAttributeType**(int idx): Returns an attribute type corresponding to the index.

3. Default Package

Default package contains three subpackages – **Util** package, **ProductTree** package, and **JSON** package. **CHMS** class only contains **main()** method. So it is just starting point of the program.

① **CHMS** Class

A. Variables

- String **databaseFileName**: Stores predefined database file name – "database.txt".

B. Methods

- void **main**(String[] args):

When program starts, CHMS creates **FileManagement** object and calls **loadDatabase()** to read database file. Then it creates **DataManagement** object to handle computer parts data. After that, it prints main menu until user wants to exit program. Finally, it calls **saveDatabase()** to write database file.