

**Element 2**

***Karolis Abramovicius***

**Programming Module Code: *CP40061E***

***BS(c) Computer Science***

**Student ID: *21378026***

**Module Leader: Fehmida Mohamedali**

# Table of Contents

[Table of Contents 2](#_Toc532138836)

[Introduction 3](#_Toc532138837)

[The User Guide for the Application 4](#_Toc532138838)

[Class Diagram for MODULE 8](#_Toc532138839)

[Class Diagram for MODULES 10](#_Toc532138840)

[Further Discussion and Future Plans for the Application 16](#_Toc532138841)

[References 17](#_Toc532138842)

[Appendix 18](#_Toc532138843)

# Introduction

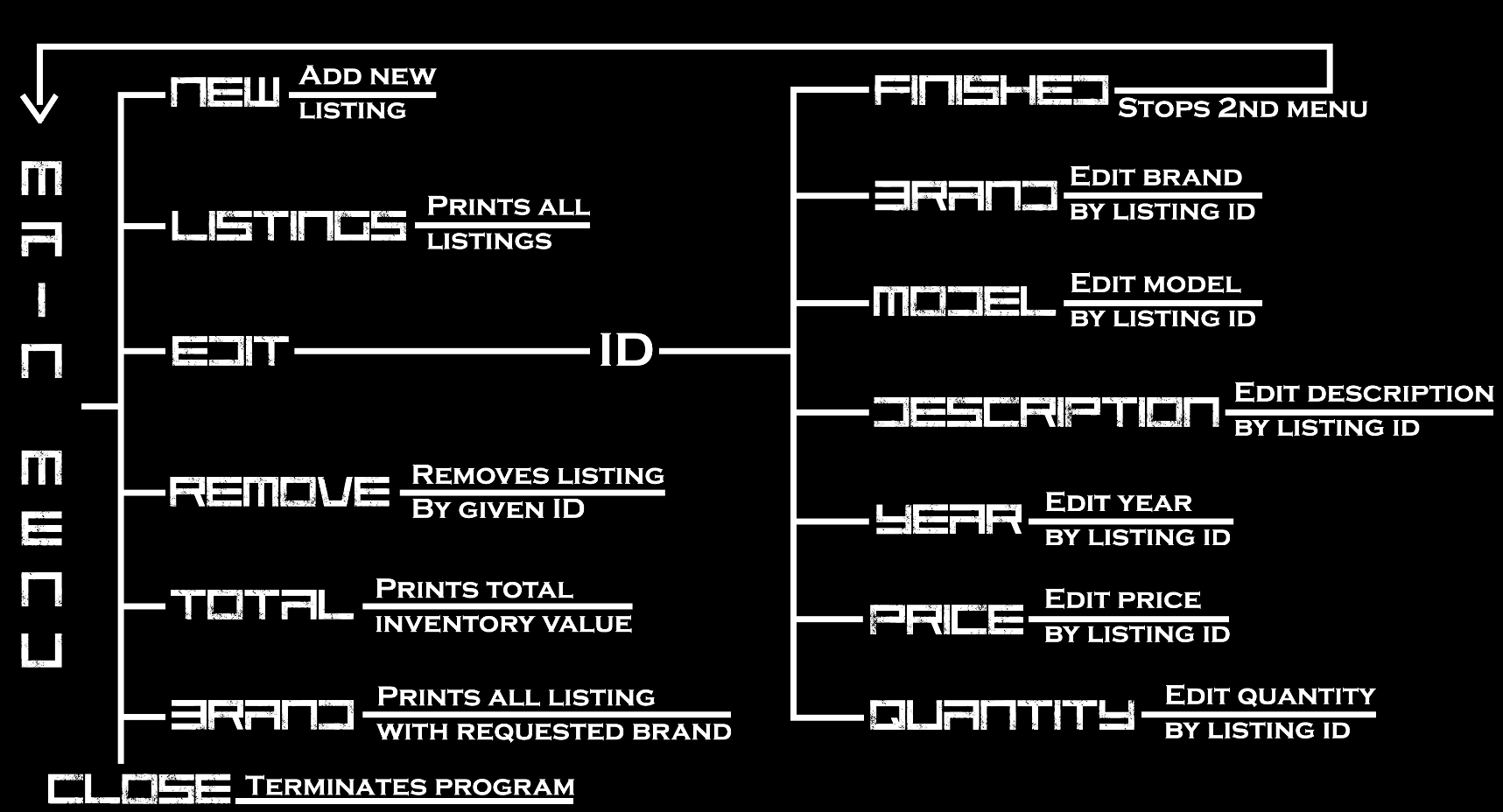
Aim of this application is to help companies, who is selling some sort of items, keep track of their products flow.

My developed program let user to store new listings, edit existing listings, delete listings, get total value of user inventory or find specific listings that he or she needs at the time.

I have implemented double menu, which helps to smoothly go through all the functions that my program provides: add new listing, view all existing listings, edit specific listing by its ID, remove specific listing by its ID, find total value of all existing listings, find all listings that has some specific brand. Secondary menu is developed for editing listings, it has different option that You can choose from, either You want to edit listing brand, model, description, year of release, price or quantity available.

To develop my program, I have used Java features like reading from and writing to file to store all data and make program more reliable and interactive. Also, I have used while, for loops, constructor, functions called setters and getters, array list, plenty of different type variables like booleans, strings, integers and double. I have used ‘switch’ to develop menu, then I have used ‘.toLowerCase()’ function to prevent crashes while entering capital and lowercase letters into the menu and plenty of other small functions of java to help me work the very best result.

Menu Diagram with menu options provided below



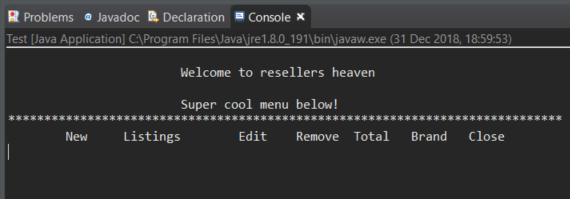
# The User Guide for the Application

**General information:**

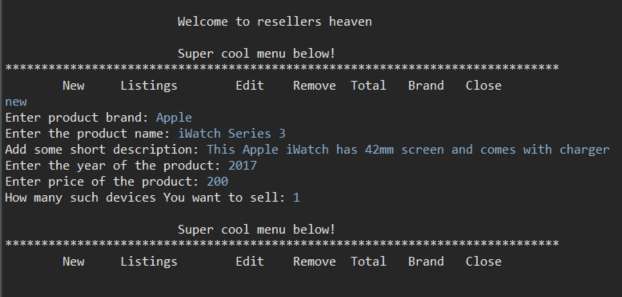
My developed application has **primary** **menu** used for main processes like adding new listing, printing out all existing listings, removing listing, getting total value of all listing inventory, searching for listings by brand and closing the program. Also, I have implemented **secondary menu**, which is used for editing specific listing brand, model, description, year, price and quantity. I have made a quick tour through my application and showed all possible operations with screenshots below!

**Please note:** my program is developed to read data from file and write data to file, I have a few listings already created in my data file for demonstrating purposes that You might see when we will try out listings option in the menu to see all listings. Also, I developed my application to be user eye friendly, so my outputs take a lot of space, that’s why not in all screenshots You can see full view of the program – it just wouldn’t fit in word document screen without a scroll bar!

**When program is launched**, console will print out welcoming message and user will also get all possible options of primary menu in his or her console screen:



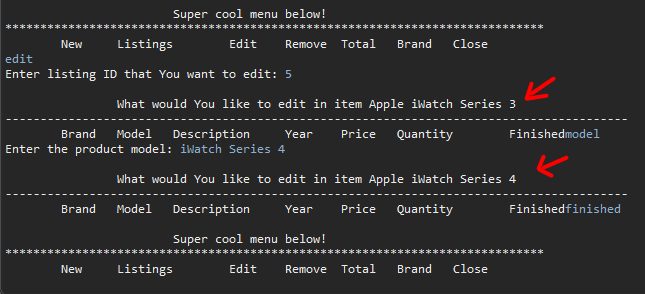
**First option** – enter ‘new’ to console and it will let You to create a new listing, will ask for listing details and will store it in listing array. After details are entered, my application will print out menu again:



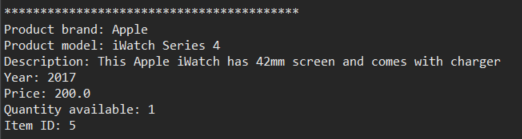
**Second option** – enter ‘listings’ to console and it will print out all existing listings, with them, we will see our new listing ‘Apple iWatch Series 3’. After details are entered, my application will print out menu again:



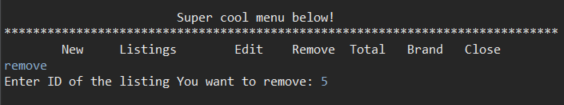
**Third option** – enter ‘edit’ to the console and application will ask for ID of listing that we want to edit. Let’s enter 5, which is out new listing ID. Then it will show secondary menu where we can choose what we want to edit from all the settings that we had to enter when creating a new listing : brand, model, description, year, price, quantity. Let’s change the model **from series 3 to series 4**. After finished editing, we need to write in ‘finished’ and we will get back to our primary menu:



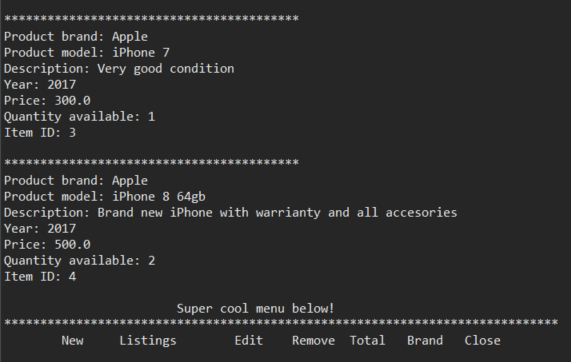
If we enter ‘listings’ now, we will see that **model have been changed** to iWatch Series 4:



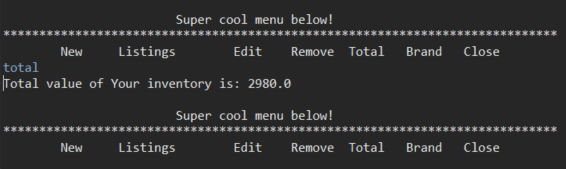
**Fourth option** – enter ‘remove’ to the console and application will ask for ID of the listing that we want to delete, let’s also insert 5, which will delete our freshly created listing. After finished deleting, application will print out menu to the console:



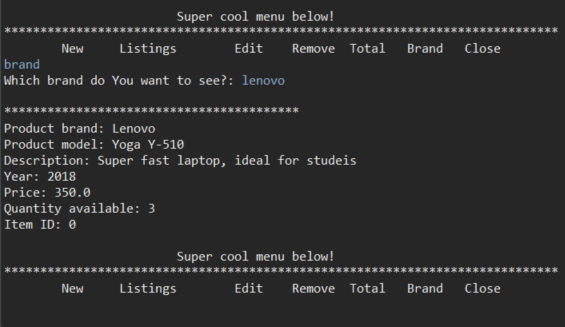
If we check listings, our Apple iWatch Series 4 does not exist anymore:



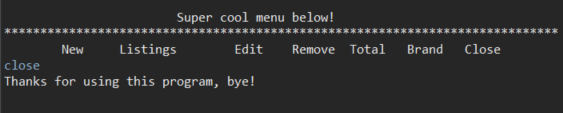
**Fifth option** – enter ‘total’ to the console and application will sum prices multiplied by the quantity of all listings and print out the value of our inventory to the console screen. After inventory value is printed, application will print out primary menu to the console:



**Sixth option** – enter ‘brand’ and console will ask which brand listing we are looking for and print all matching to the console. After that application will print out primary menu to the console:



**Seventh option** – enter ‘close’ and application will print our goodbye message and terminate its work:



# Class Diagram for MODULE

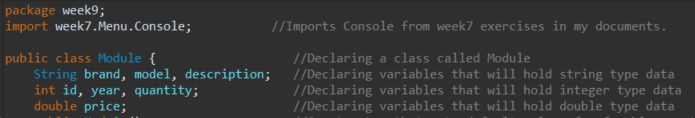
# 

# Class Diagram for MODULES

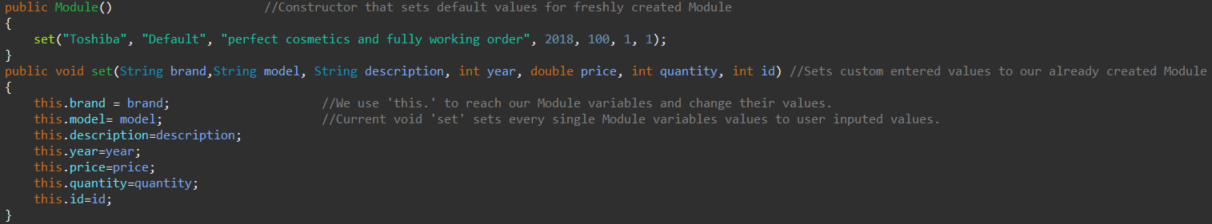
# 

# Purpose of each class

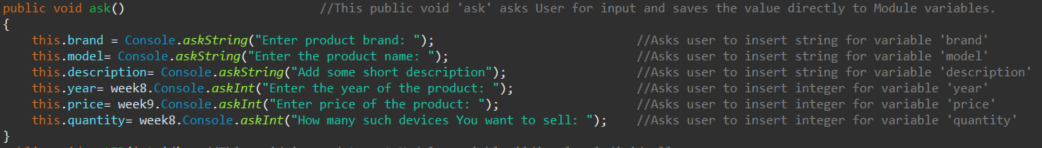
**Adding screenshots with grey comments inside. Java features used will be explained within the code in screenshots**

**1st – Module class code screenshots with explanations:**

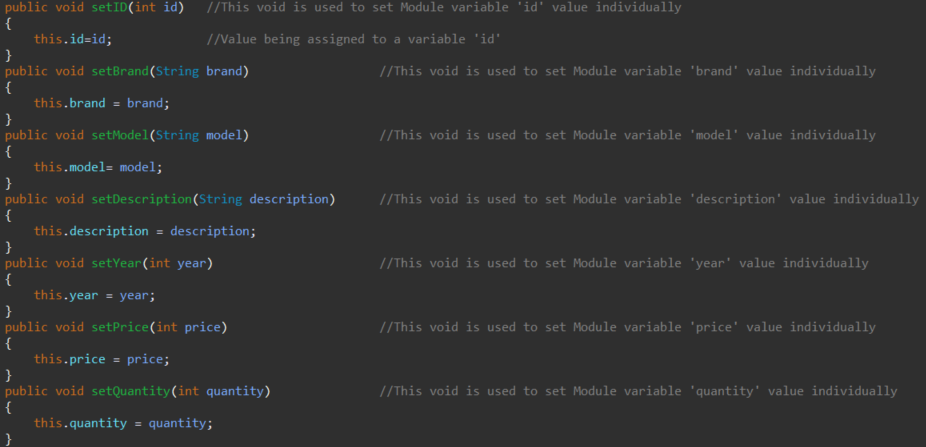
**Here is my default and custom constructors**



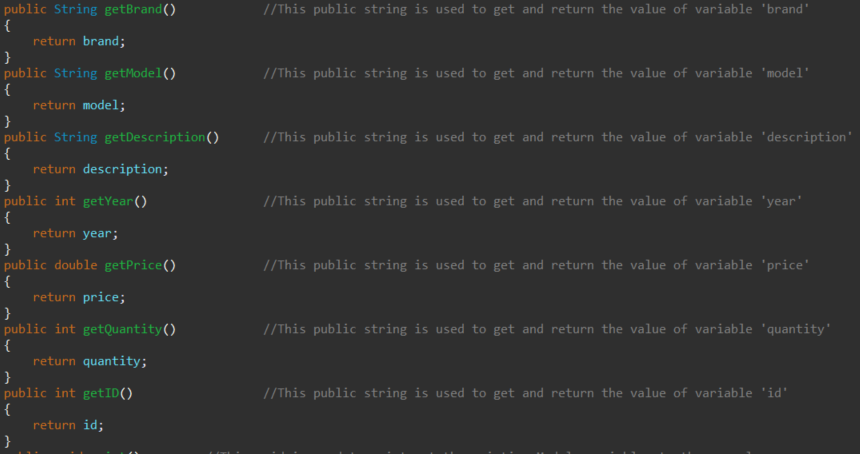
**Void that asks user input and stores it to Module variables**



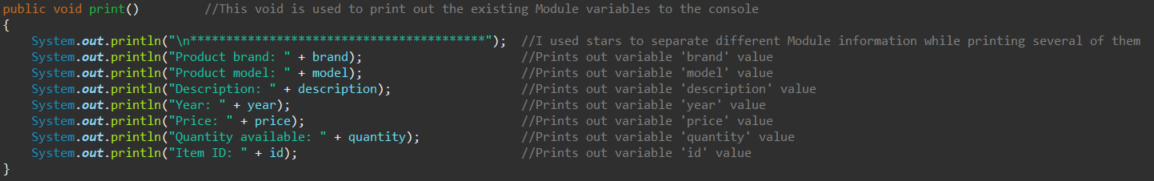
**Bunch of setters for individual variables**



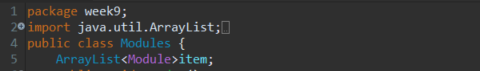
**Bunch of getters for individual variables**



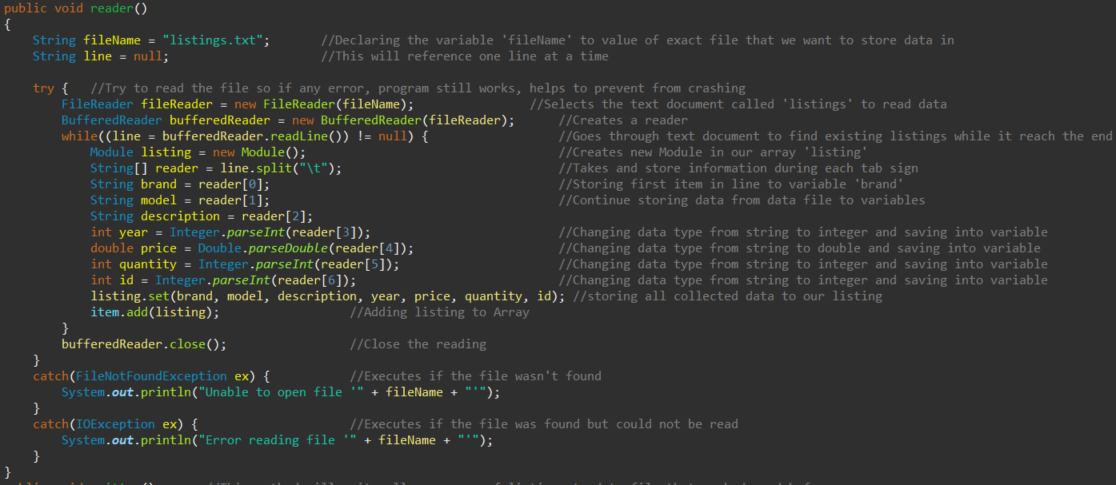
**Void to print out listing to console**



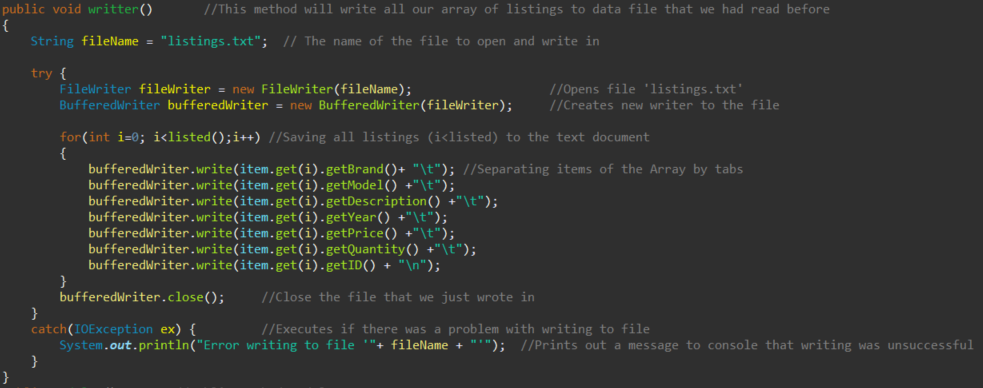
**2nd- Modules class code screenshots with explanations:**



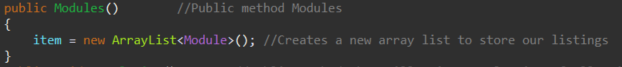
**Public void reader used to read data from file and store it to array**



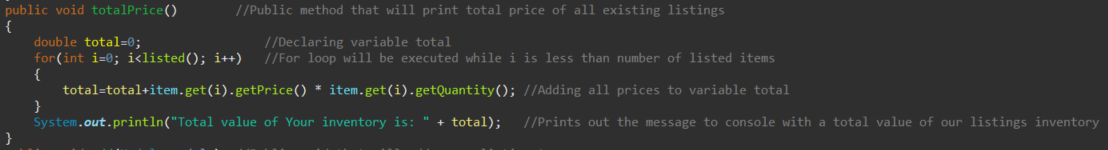
**Public void writer used to write data to file from array**



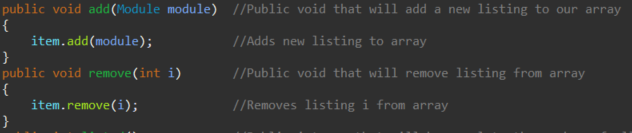
**Public Modules used to create an Array named “item”**



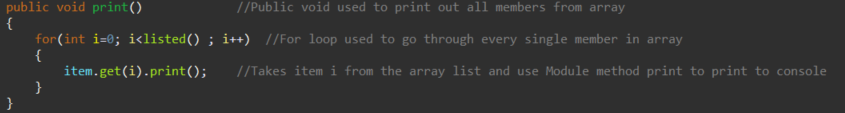
**Public void used to work out a total value of inventory and print it out to console**



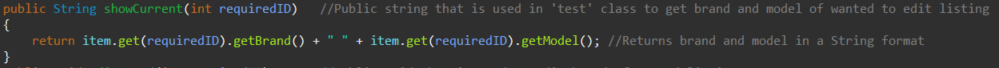
**Public voids used to add new listing to the array and remove listing from array**



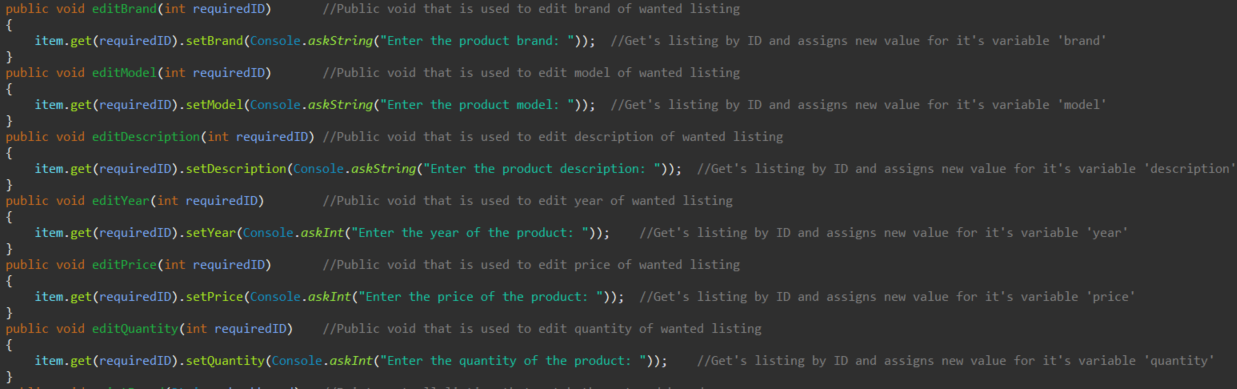
**Public void that prints all array elements to the console**



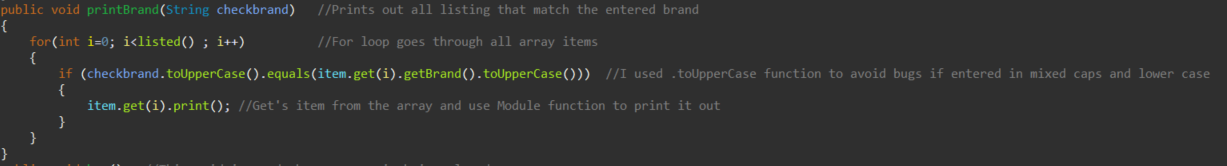
**Public void that finds listing by ID and works out it’s brand and model**



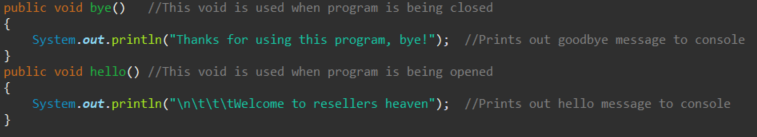
**Bunch of voids that sets the new value of variable in the listing**



**Void that prints out all listing that are made by required brand**



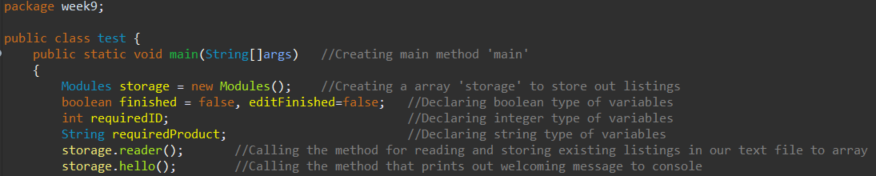
**Two voids, one prints welcoming message and other one goodbye message**



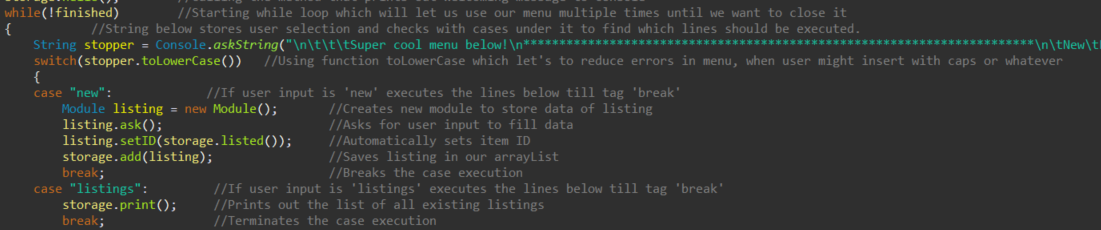
**3rd – Test class code with explanations**

**This class is used to connect other two classes, run menu and application itself, since it is only class that contains main() method.**

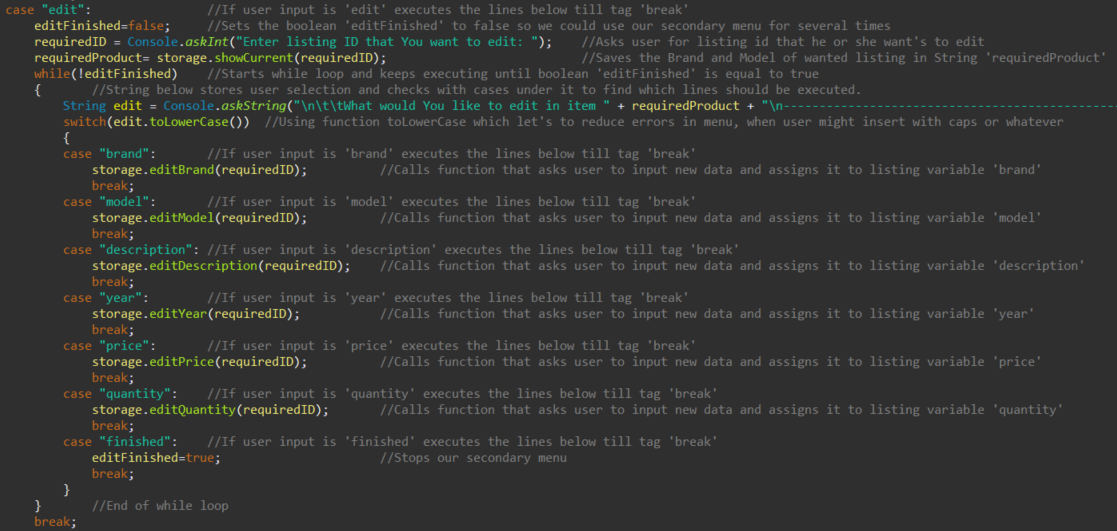
**Calling hello message, storing data from file to array and declaring variables**



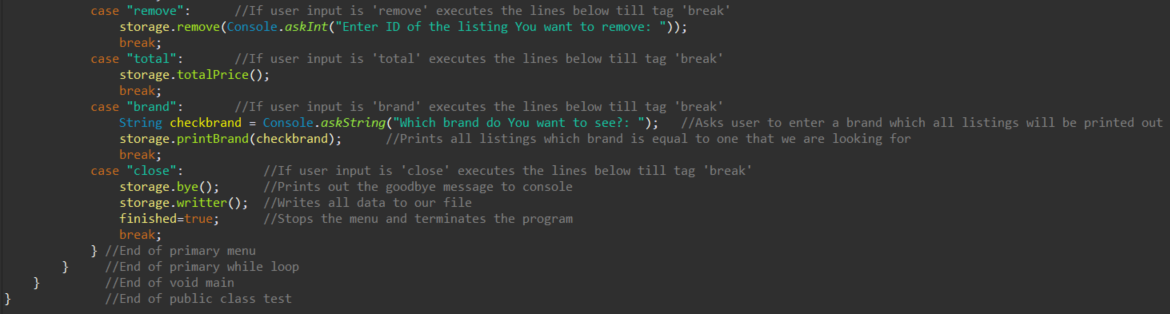
**Primary menu with options (String stopper is too long to fit in screenshot)**



**Case edit has a secondary menu inside of it as shown in screenshot**



**Other part of primary menu and the end of the class**



**Note: All screenshots have been taken in continues order to keep the integrity. If You connect each screenshot ending with next screenshot starting You can make picture of all program code. I hope You can see text clearly as I tried to zoom screenshots as close as possible to fit in the document.**

# Further Discussion and Future Plans for the Application

During my free time I am fixing phones and laptops. I buy them faulty, fix them and then sell them. I was inspired to create this program to help myself optimize the track of devices going through my hands. With help of this program I will be able to store all the devices that I have owned, I will be able to see how much money I made and how much time I spent. For the future, to finalize the application, I am planning to add features like storing phone numbers of people that I bought and sold device to, add parameters like price that I bought for and date and after that the price that I sold for and date when I sold the device, I will also consider adding and saving IMEI codes just in case of a crime, so I would always know what device I had and who sold it to me, so I could help officers to find the thief. I am planning to implement all the missing function in the time of one month, because I am excited to start my own statistics and documentation.

Also, in the further future, when I learn to develop mobile applications, I am planning to transfer this application on mobile platform and maybe even let it go online in google play store and apple store so that other people with needs like me could use my designed application to make their life easier.

# References

* I have used information that I’ve learned from lectures and seminars.
* My own general programming knowledge

# Appendix

Full copy of code listings for all classes provided below.

Modules:

package week9;

import java.util.ArrayList;

import java.io.\*;

public class Modules {

ArrayList<Module>item;

public void reader()

{

String fileName = "listings.txt"; //Declaring the variable 'fileName' to value of exact file that we want to store data in

String line = null; //This will reference one line at a time

try { //Try to read the file so if any error, program still works, helps to prevent from crashing

FileReader fileReader = new FileReader(fileName); //Selects the text document called 'listings' to read data

BufferedReader bufferedReader = new BufferedReader(fileReader); //Creates a reader

while((line = bufferedReader.readLine()) != null) { //Goes through text document to find existing listings while it reach the end

Module listing = new Module(); //Creates new Module in our array 'listing'

String[] reader = line.split("\t"); //Takes and store information during each tab sign

String brand = reader[0]; //Storing first item in line to variable 'brand'

String model = reader[1]; //Continue storing data from data file to variables

String description = reader[2];

int year = Integer.parseInt(reader[3]); //Changing data type from string to integer and saving into variable

double price = Double.parseDouble(reader[4]); //Changing data type from string to double and saving into variable

int quantity = Integer.parseInt(reader[5]); //Changing data type from string to integer and saving into variable

int id = Integer.parseInt(reader[6]); //Changing data type from string to integer and saving into variable

listing.set(brand, model, description, year, price, quantity, id); //storing all collected data to our listing

item.add(listing); //Adding listing to Array

}

bufferedReader.close(); //Close the reading

}

catch(FileNotFoundException ex) { //Executes if the file wasn't found

System.out.println("Unable to open file '" + fileName + "'");

}

catch(IOException ex) { //Executes if the file was found but could not be read

System.out.println("Error reading file '" + fileName + "'");

}

}

public void writter() //This method will write all our array of listings to data file that we had read before

{

String fileName = "listings.txt"; // The name of the file to open and write in

try {

FileWriter fileWriter = new FileWriter(fileName); //Opens file 'listings.txt'

BufferedWriter bufferedWriter = new BufferedWriter(fileWriter); //Creates new writer to the file

for(int i=0; i<listed();i++) //Saving all listings (i<listed) to the text document

{

bufferedWriter.write(item.get(i).getBrand()+ "\t"); //Separating items of the Array by tabs

bufferedWriter.write(item.get(i).getModel() +"\t");

bufferedWriter.write(item.get(i).getDescription() +"\t");

bufferedWriter.write(item.get(i).getYear() +"\t");

bufferedWriter.write(item.get(i).getPrice() +"\t");

bufferedWriter.write(item.get(i).getQuantity() +"\t");

bufferedWriter.write(item.get(i).getID() + "\n");

}

bufferedWriter.close(); //Close the file that we just wrote in

}

catch(IOException ex) { //Executes if there was a problem with writing to file

System.out.println("Error writing to file '"+ fileName + "'"); //Prints out a message to console that writing was unsuccessful

}

}

public Modules() //Public method Modules

{

item = new ArrayList<Module>(); //Creates a new array list to store our listings

}

public void totalPrice() //Public method that will print total price of all existing listings

{

double total=0; //Declaring variable total

for(int i=0; i<listed(); i++) //For loop will be executed while i is less than number of listed items

{

total=total+item.get(i).getPrice() \* item.get(i).getQuantity(); //Adding all prices to variable total

}

System.out.println("Total value of Your inventory is: " + total); //Prints out the message to console with a total value of our listings inventory

}

public void add(Module module) //Public void that will add a new listing to our array

{

item.add(module); //Adds new listing to array

}

public void remove(int i) //Public void that will remove listing from array

{

item.remove(i); //Removes listing i from array

}

public int listed() //Public integer that will be equal to the number of all existing listings

{

return item.size(); //.size() function let's us to simply get the size of our array

}

public void print() //Public void used to print out all members from array

{

for(int i=0; i<listed() ; i++) //For loop used to go through every single member in array

{

item.get(i).print(); //Takes item i from the array list and use Module method print to print to console

}

}

public String showCurrent(int requiredID) //Public string that is used in 'test' class to get brand and model of wanted to edit listing

{

return item.get(requiredID).getBrand() + " " + item.get(requiredID).getModel(); //Returns brand and model in a String format

}

public void editBrand(int requiredID) //Public void that is used to edit brand of wanted listing

{

item.get(requiredID).setBrand(Console.askString("Enter the product brand: ")); //Get's listing by ID and assigns new value for it's variable 'brand'

}

public void editModel(int requiredID) //Public void that is used to edit model of wanted listing

{

item.get(requiredID).setModel(Console.askString("Enter the product model: ")); //Get's listing by ID and assigns new value for it's variable 'model'

}

public void editDescription(int requiredID) //Public void that is used to edit description of wanted listing

{

item.get(requiredID).setDescription(Console.askString("Enter the product description: ")); //Get's listing by ID and assigns new value for it's variable 'description'

}

public void editYear(int requiredID) //Public void that is used to edit year of wanted listing

{

item.get(requiredID).setYear(Console.askInt("Enter the year of the product: ")); //Get's listing by ID and assigns new value for it's variable 'year'

}

public void editPrice(int requiredID) //Public void that is used to edit price of wanted listing

{

item.get(requiredID).setPrice(Console.askInt("Enter the price of the product: ")); //Get's listing by ID and assigns new value for it's variable 'price'

}

public void editQuantity(int requiredID) //Public void that is used to edit quantity of wanted listing

{

item.get(requiredID).setQuantity(Console.askInt("Enter the quantity of the product: ")); //Get's listing by ID and assigns new value for it's variable 'quantity'

}

public void printBrand(String checkbrand) //Prints out all listing that match the entered brand

{

for(int i=0; i<listed() ; i++) //For loop goes through all array items

{

if (checkbrand.toUpperCase().equals(item.get(i).getBrand().toUpperCase())) //I used .toUpperCase function to avoid bugs if entered in mixed caps and lower case

{

item.get(i).print(); //Get's item from the array and use Module function to print it out

}

}

}

public void bye() //This void is used when program is being closed

{

System.out.println("Thanks for using this program, bye!"); //Prints out goodbye message to console

}

public void hello() //This void is used when program is being opened

{

System.out.println("\n\t\t\tWelcome to resellers heaven"); //Prints out hello message to console

}

}

Module:

package week9;

import week7.Menu.Console; //Imports Console from week7 exercises in my documents.

public class Module { //Declaring a class called Module

String brand, model, description; //Declaring variables that will hold string type data

int id, year, quantity; //Declaring variables that will hold integer type data

double price; //Declaring variables that will hold double type data

public Module() //Constructor that sets default values for freshly created Module

{

set("Toshiba", "Default", "perfect cosmetics and fully working order", 2018, 100, 1, 1);

}

public void set(String brand,String model, String description, int year, double price, int quantity, int id) //Sets custom entered values to our already created Module

{

this.brand = brand; //We use 'this.' to reach our Module variables and change their values.

this.model= model; //Current void 'set' sets every single Module variables values to user inputed values.

this.description=description;

this.year=year;

this.price=price;

this.quantity=quantity;

this.id=id;

}

public void ask() //This public void 'ask' asks User for input and saves the value directly to Module variables.

{

this.brand = Console.*askString*("Enter product brand: "); //Asks user to insert string for variable 'brand'

this.model= Console.*askString*("Enter the product name: "); //Asks user to insert string for variable 'model'

this.description= Console.*askString*("Add some short description: "); //Asks user to insert string for variable 'description'

this.year= week8.Console.*askInt*("Enter the year of the product: "); //Asks user to insert integer for variable 'year'

this.price= week9.Console.*askInt*("Enter price of the product: "); //Asks user to insert integer for variable 'price'

this.quantity= week8.Console.*askInt*("How many such devices You want to sell: "); //Asks user to insert integer for variable 'quantity'

}

public void setID(int id) //This void is used to set Module variable 'id' value individually

{

this.id=id; //Value being assigned to a variable 'id'

}

public void setBrand(String brand) //This void is used to set Module variable 'brand' value individually

{

this.brand = brand;

}

public void setModel(String model) //This void is used to set Module variable 'model' value individually

{

this.model= model;

}

public void setDescription(String description) //This void is used to set Module variable 'description' value individually

{

this.description = description;

}

public void setYear(int year) //This void is used to set Module variable 'year' value individually

{

this.year = year;

}

public void setPrice(int price) //This void is used to set Module variable 'price' value individually

{

this.price = price;

}

public void setQuantity(int quantity) //This void is used to set Module variable 'quantity' value individually

{

this.quantity = quantity;

}

public String getBrand() //This public string is used to get and return the value of variable 'brand'

{

return brand;

}

public String getModel() //This public string is used to get and return the value of variable 'model'

{

return model;

}

public String getDescription() //This public string is used to get and return the value of variable 'description'

{

return description;

}

public int getYear() //This public string is used to get and return the value of variable 'year'

{

return year;

}

public double getPrice() //This public string is used to get and return the value of variable 'price'

{

return price;

}

public int getQuantity() //This public string is used to get and return the value of variable 'quantity'

{

return quantity;

}

public int getID() //This public string is used to get and return the value of variable 'id'

{

return id;

}

public void print() //This void is used to print out the existing Module variables to the console

{

System.***out***.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); //I used stars to separate different Module information while printing several of them

System.***out***.println("Product brand: " + brand); //Prints out variable 'brand' value

System.***out***.println("Product model: " + model); //Prints out variable 'model' value

System.***out***.println("Description: " + description); //Prints out variable 'description' value

System.***out***.println("Year: " + year); //Prints out variable 'year' value

System.***out***.println("Price: " + price); //Prints out variable 'price' value

System.***out***.println("Quantity available: " + quantity); //Prints out variable 'quantity' value

System.***out***.println("Item ID: " + id); //Prints out variable 'id' value

}

}

TEST

package week9;

public class test {

public static void main(String[]args) //Creating main method 'main'

{

Modules storage = new Modules(); //Creating a array 'storage' to store out listings

boolean finished = false, editFinished=false; //Declaring boolean type of variables

int requiredID; //Declaring integer type of variables

String requiredProduct; //Declaring string type of variables

storage.reader(); //Calling the method for reading and storing existing listings in our text file to array

storage.hello(); //Calling the method that prints out welcoming message to console

while(!finished) //Starting while loop which will let us use our menu multiple times until we want to close it

{ //String below stores user selection and checks with cases under it to find which lines should be executed.

String stopper = Console.*askString*("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\tNew\tListings\tEdit\tRemove\tTotal\tBrand\tClose\n");

switch(stopper.toLowerCase()) //Using function toLowerCase which let's to reduce errors in menu, when user might insert with caps or whatever

{

case "new": //If user input is 'new' executes the lines below till tag 'break'

Module listing = new Module(); //Creates new module to store data of listing

listing.ask(); //Asks for user input to fill data

listing.setID(storage.listed()); //Automatically sets item ID

storage.add(listing); //Saves listing in our arrayList

break; //Breaks the case execution

case "listings": //If user input is 'listings' executes the lines below till tag 'break'

storage.print(); //Prints out the list of all existing listings

break; //Terminates the case execution

case "edit": //If user input is 'edit' executes the lines below till tag 'break'

editFinished=false; //Sets the boolean 'editFinished' to false so we could use our secondary menu for several times

requiredID = Console.*askInt*("Enter listing ID that You want to edit: "); //Asks user for listing id that he or she want's to edit

while(!editFinished) //Starts while loop and keeps executing until boolean 'editFinished' is equal to true

{ requiredProduct= storage.showCurrent(requiredID); //Saves the Brand and Model of wanted listing in String 'requiredProduct'

//String below stores user selection and checks with cases under it to find which lines should be executed.

String edit = Console.*askString*("\n\t\tWhat would You like to edit in item " + requiredProduct + "\n-----------------------------------------------------------------------------------------\n\tBrand\tModel\tDescription\tYear\tPrice\tQuantity\tFinished");

switch(edit.toLowerCase()) //Using function toLowerCase which let's to reduce errors in menu, when user might insert with caps or whatever

{

case "brand": //If user input is 'brand' executes the lines below till tag 'break'

storage.editBrand(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'brand'

break;

case "model": //If user input is 'model' executes the lines below till tag 'break'

storage.editModel(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'model'

break;

case "description": //If user input is 'description' executes the lines below till tag 'break'

storage.editDescription(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'description'

break;

case "year": //If user input is 'year' executes the lines below till tag 'break'

storage.editYear(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'year'

break;

case "price": //If user input is 'price' executes the lines below till tag 'break'

storage.editPrice(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'price'

break;

case "quantity": //If user input is 'quantity' executes the lines below till tag 'break'

storage.editQuantity(requiredID); //Calls function that asks user to input new data and assigns it to listing variable 'quantity'

break;

case "finished": //If user input is 'finished' executes the lines below till tag 'break'

editFinished=true; //Stops our secondary menu

break;

}

} //End of while loop

break;

case "remove": //If user input is 'remove' executes the lines below till tag 'break'

storage.remove(Console.*askInt*("Enter ID of the listing You want to remove: "));

break;

case "total": //If user input is 'total' executes the lines below till tag 'break'

storage.totalPrice();

break;

case "brand": //If user input is 'brand' executes the lines below till tag 'break'

String checkbrand = Console.*askString*("Which brand do You want to see?: "); //Asks user to enter a brand which all listings will be printed out

storage.printBrand(checkbrand); //Prints all listings which brand is equal to one that we are looking for

break;

case "close": //If user input is 'close' executes the lines below till tag 'break'

storage.bye(); //Prints out the goodbye message to console

storage.writter(); //Writes all data to our file

finished=true; //Stops the menu and terminates the program

break;

} //End of primary menu

} //End of primary while loop

} //End of void main

} //End of public class test