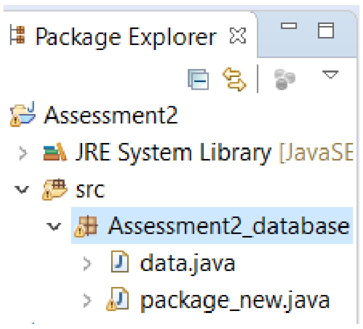
1. Overview

This program represents a store that allows the user to purchase iPhones, amend their pricing and stock, view your inventory and there are a number of features that will be explained in detail in this documentation.

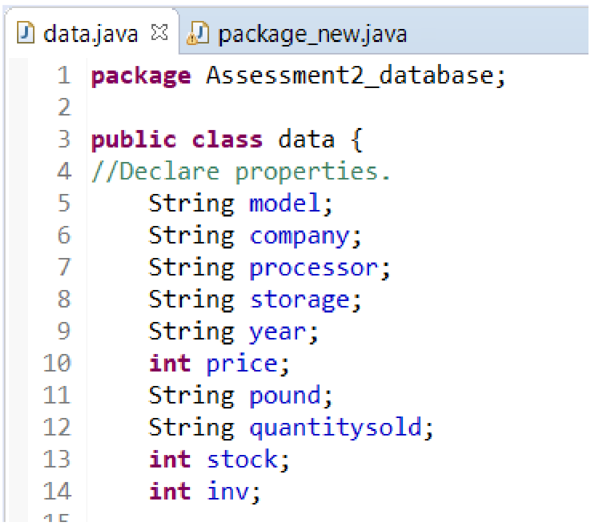
This program consists of 2 different classes:



data

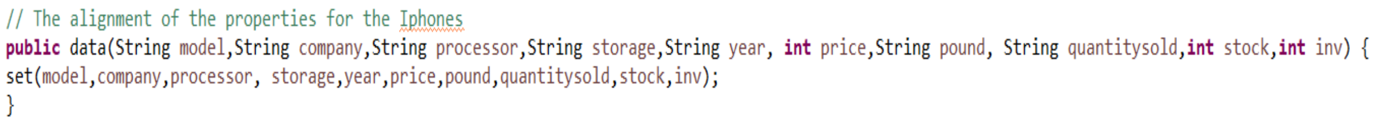
After making the public class **data**, we declare the strings and integers that are used to define the different iPhones and make them unique.

They're the model (**model**), the company that made the phone (**company**), the processor that runs on the phone (**processor**), the amount of storage the phone holds (**storage**), the release date (**year**), the price of the product (**price**), the currency (**pound**), the amount of prototypes sold of each one of the products (**quantitysold**), the amount of prototypes currently in stock (**stock**), and inventory (**inv**)



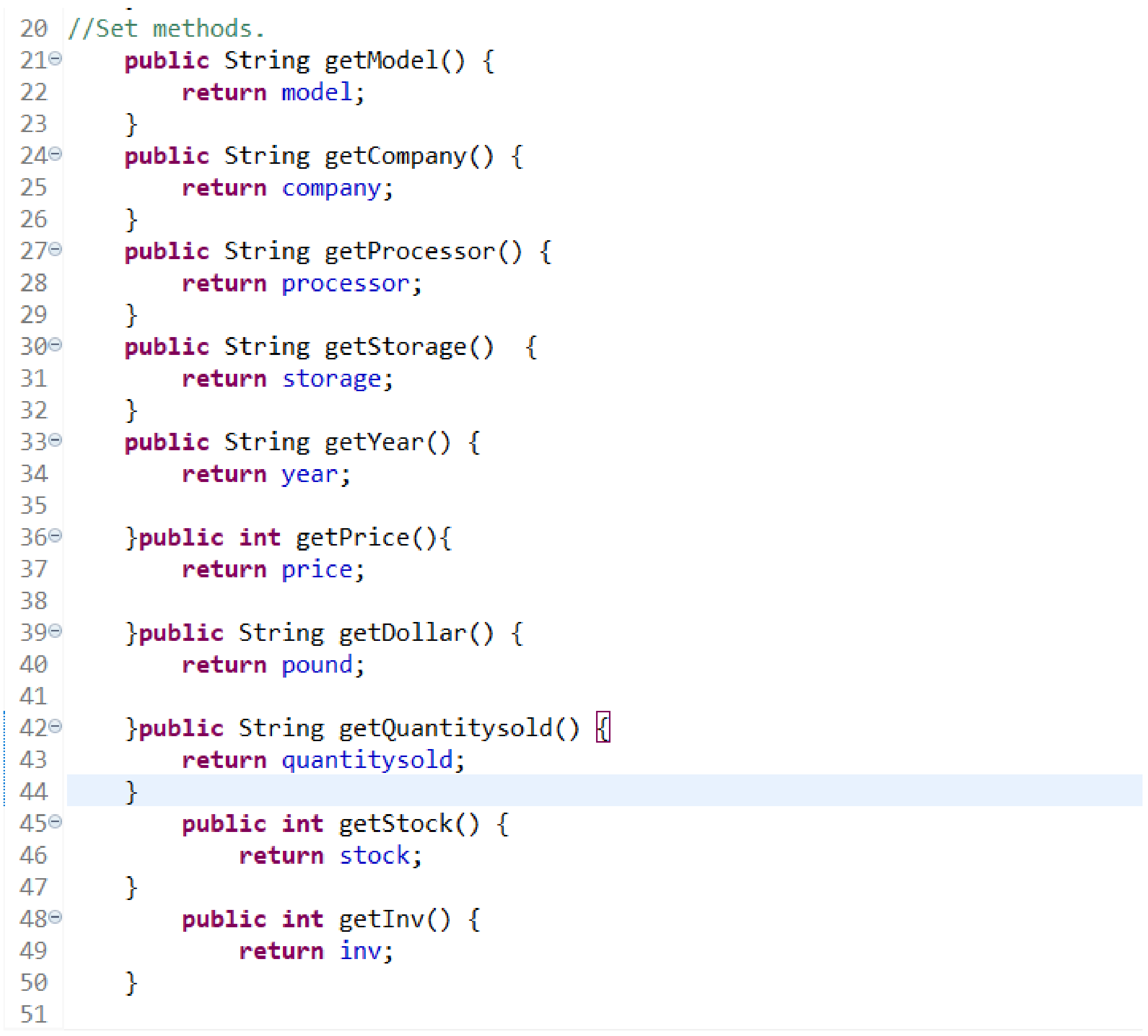
```java
package Assessment2_database;

public class data {
//Declare properties.
    String model;
    String company;
    String processor;
    String storage;
    String year;
    int price;
    String pound;
    String quantitysold;
    int stock;
    int inv;
```

Afterwards, the alignment of the strings and integers is set. This is to be used when creating objects from this class.



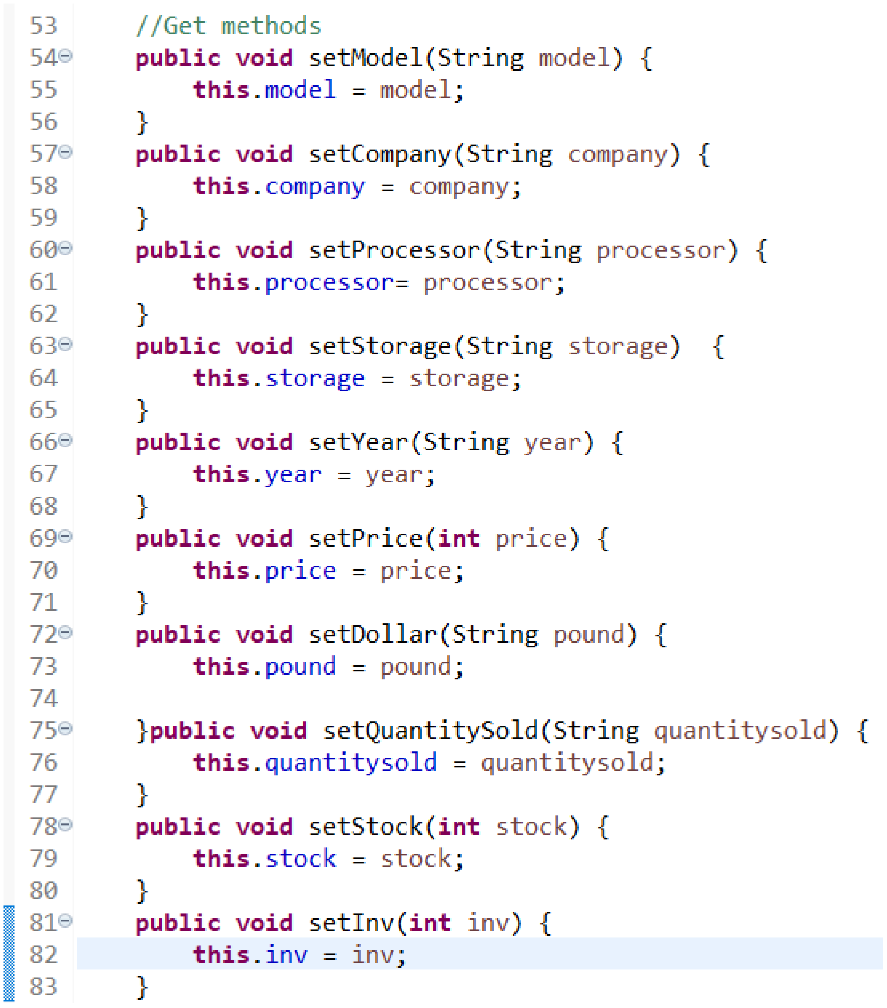
```java
// The alignment of the properties for the Iphones
public data(String model,String company,String processor,String storage,String year, int price,String pound, String quantitysold,int stock,int inv) {
set(model,company,processor, storage,year,price,pound,quantitysold,stock,inv);
}
```

This sections contains the declaration of the set methods which will later be used to change any values of the objects.



```java
//Set methods.
    public String getModel() {
        return model;
    }
    public String getCompany() {
        return company;
    }
    public String getProcessor() {
        return processor;
    }
    public String getStorage()  {
        return storage;
    }
    public String getYear() {
        return year;

    }public int getPrice(){
        return price;

    }public String getDollar() {
        return pound;

    }public String getQuantitysold() {
        return quantitysold;
    }
        public int getStock() {
            return stock;
    }
        public int getInv() {
            return inv;
    }

```

Get methods are also used to retrieve information about the iPhones.

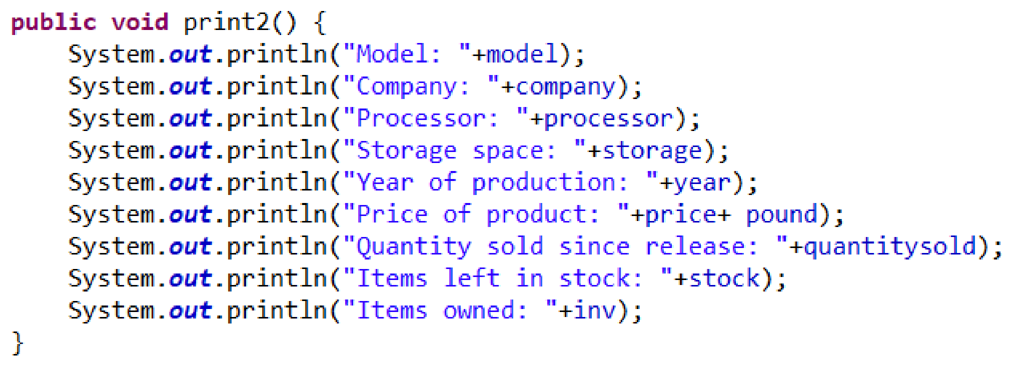


```
    //Get methods
    public void setModel(String model) {
        this.model = model;
    }
    public void setCompany(String company) {
        this.company = company;
    }
    public void setProcessor(String processor) {
        this.processor= processor;
    }
    public void setStorage(String storage)  {
        this.storage = storage;
    }
    public void setYear(String year) {
        this.year = year;
    }
    public void setPrice(int price) {
        this.price = price;
    }
    public void setDollar(String pound) {
        this.pound = pound;

    }public void setQuantitySold(String quantitysold) {
        this.quantitysold = quantitysold;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
    public void setInv(int inv) {
        this.inv = inv;
    }
```

This is the set method for the whole class and can be used to add new products by overwriting a blank placeholder model in the model array in the **package_new** class. This is due to the limitations the Array class, which enforces the declaring of static number of objects within an array.

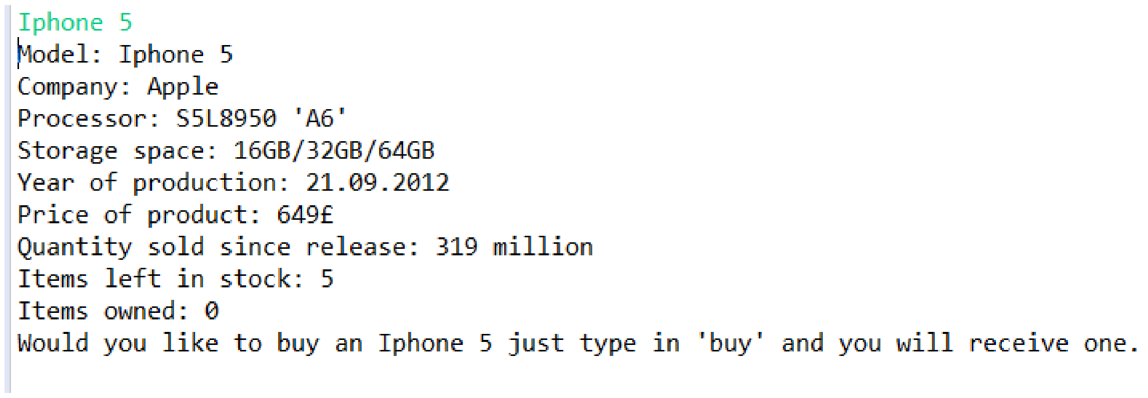
```java
public void set(String model, String company, String processor ,
        String storage, String year, int price,String pound, String quantitysold,int stock,int inv) {
    setModel(model);
    setCompany(company);
    setProcessor(processor);
    setStorage(storage);
    setYear(year);
    setPrice(price);
    setDollar(pound);
    setQuantitySold(quantitysold);
    setStock(stock);
    setInv(inv);
    }
```

This is a pre-defined print function. It prints out the whole information about the products - model, company, processor, storage space, release date, price of the products, quantity of prototypes sold since release, items in stock, and inventory number.



```
public void print2() {
    System.out.println("Model: "+model);
    System.out.println("Company: "+company);
    System.out.println("Processor: "+processor);
    System.out.println("Storage space: "+storage);
    System.out.println("Year of production: "+year);
    System.out.println("Price of product: "+price+ pound);
    System.out.println("Quantity sold since release: "+quantitysold);
    System.out.println("Items left in stock: "+stock);
    System.out.println("Items owned: "+inv);
}
```

This is the result when the user inputs the model of the phone in the menu. In the screenshot below, “iPhone 5” was used as input.

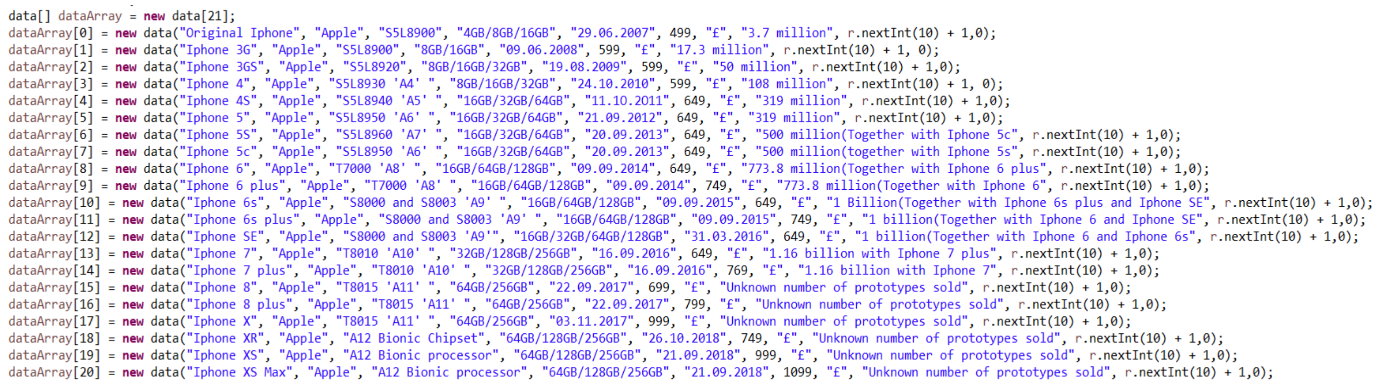


```
Iphone 5
Model: Iphone 5
Company: Apple
Processor: S5L8950 'A6'
Storage space: 16GB/32GB/64GB
Year of production: 21.09.2012
Price of product: 649£
Quantity sold since release: 319 million
Items left in stock: 5
Items owned: 0
Would you like to buy an Iphone 5 just type in 'buy' and you will receive one.
```

Note: The items left in stock integer is set to a random number between 1 and 10.

This is all the information for the first class **data**.

These are all the products that are created (21 in total). An array has been used to store that information and to later minimize the lines of code used to check all products.



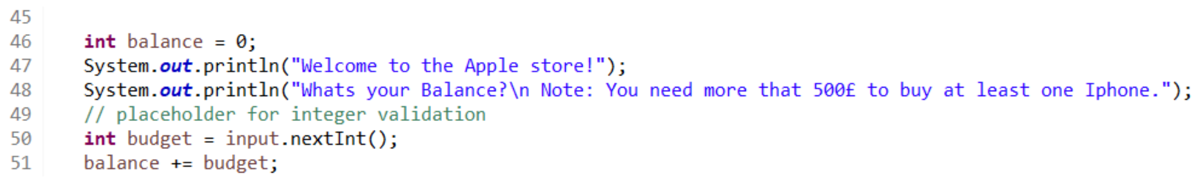
```java
data[] dataArray = new data[21];
dataArray[0] = new data("Original Iphone", "Apple", "S5L8900", "4GB/8GB/16GB", "29.06.2007", 499, "£", "3.7 million", r.nextInt(10) + 1,0);
dataArray[1] = new data("Iphone 3G", "Apple", "S5L8900", "8GB/16GB", "09.06.2008", 599, "£", "17.3 million", r.nextInt(10) + 1, 0);
dataArray[2] = new data("Iphone 3GS", "Apple", "S5L8920", "8GB/16GB/32GB", "19.08.2009", 599, "£", "50 million", r.nextInt(10) + 1,0);
dataArray[3] = new data("Iphone 4", "Apple", "S5L8930 'A4' ", "8GB/16GB/32GB", "24.10.2010", 599, "£", "108 million", r.nextInt(10) + 1, 0);
dataArray[4] = new data("Iphone 4S", "Apple", "S5L8940 'A5' ", "16GB/32GB/64GB", "11.10.2011", 649, "£", "319 million", r.nextInt(10) + 1,0);
dataArray[5] = new data("Iphone 5", "Apple", "S5L8950 'A6' ", "16GB/32GB/64GB", "21.09.2012", 649, "£", "319 million", r.nextInt(10) + 1,0);
dataArray[6] = new data("Iphone 5S", "Apple", "S5L8960 'A7' ", "16GB/32GB/64GB", "20.09.2013", 649, "£", "500 million(Together with Iphone 5c", r.nextInt(10) + 1,0);
dataArray[7] = new data("Iphone 5c", "Apple", "S5L8950 'A6' ", "16GB/32GB/64GB", "20.09.2013", 649, "£", "500 million(together with Iphone 5s", r.nextInt(10) + 1,0);
dataArray[8] = new data("Iphone 6", "Apple", "T7000 'A8' ", "16GB/64GB/128GB", "09.09.2014", 649, "£", "773.8 million(Together with Iphone 6 plus", r.nextInt(10) + 1,0);
dataArray[9] = new data("Iphone 6 plus", "Apple", "T7000 'A8' ", "16GB/64GB/128GB", "09.09.2014", 749, "£", "773.8 million(Together with Iphone 6", r.nextInt(10) + 1,0);
dataArray[10] = new data("Iphone 6s", "Apple", "S8000 and S8003 'A9' ", "16GB/64GB/128GB", "09.09.2015", 649, "£", "1 Billion(Together with Iphone 6s plus and Iphone SE", r.nextInt(10) + 1,0);
dataArray[11] = new data("Iphone 6s plus", "Apple", "S8000 and S8003 'A9' ", "16GB/64GB/128GB", "09.09.2015", 749, "£", "1 billion(Together with Iphone 6 and Iphone SE", r.nextInt(10) + 1,0);
dataArray[12] = new data("Iphone SE", "Apple", "S8000 and S8003 'A9'", "16GB/32GB/64GB/128GB", "31.03.2016", 649, "£", "1 billion(Together with Iphone 6 and Iphone 6s", r.nextInt(10) + 1,0);
dataArray[13] = new data("Iphone 7", "Apple", "T8010 'A10' ", "32GB/128GB/256GB", "16.09.2016", 649, "£", "1.16 billion with Iphone 7 plus", r.nextInt(10) + 1,0);
dataArray[14] = new data("Iphone 7 plus", "Apple", "T8010 'A10' ", "32GB/128GB/256GB", "16.09.2016", 769, "£", "1.16 billion with Iphone 7", r.nextInt(10) + 1,0);
dataArray[15] = new data("Iphone 8", "Apple", "T8015 'A11' ", "64GB/256GB", "22.09.2017", 699, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
dataArray[16] = new data("Iphone 8 plus", "Apple", "T8015 'A11' ", "64GB/256GB", "22.09.2017", 799, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
dataArray[17] = new data("Iphone X", "Apple", "T8015 'A11' ", "64GB/256GB", "03.11.2017", 999, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
dataArray[18] = new data("Iphone XR", "Apple", "A12 Bionic Chipset", "64GB/128GB/256GB", "26.10.2018", 749, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
dataArray[19] = new data("Iphone XS", "Apple", "A12 Bionic processor", "64GB/128GB/256GB", "21.09.2018", 999, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
dataArray[20] = new data("Iphone XS Max", "Apple", "A12 Bionic processor", "64GB/128GB/256GB", "21.09.2018", 1099, "£", "Unknown number of prototypes sold", r.nextInt(10) + 1,0);
```

When the program starts, this is the output

Welcome to the Apple store!

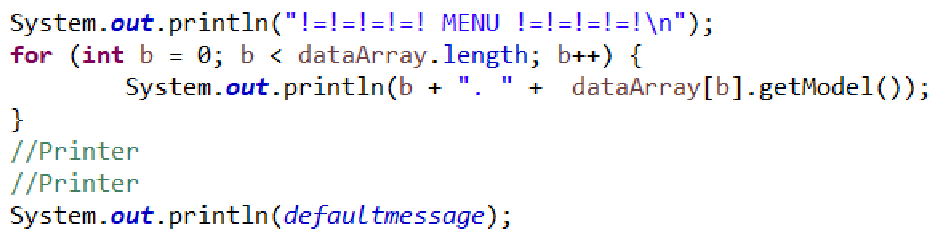
What's your balance?

Note: An input of more than £500 is needed in order to buy the very first and cheapest product. The number you input will be stored in the integer balance.



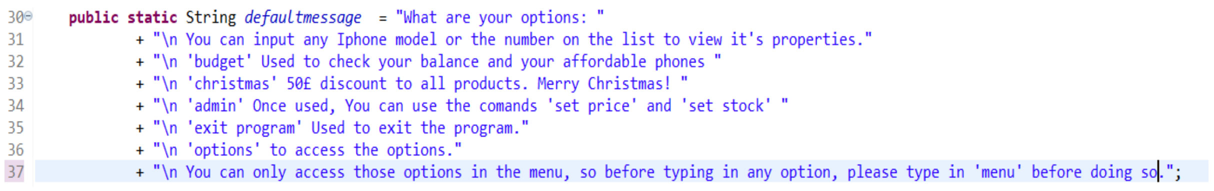
```java

int balance = 0;
System.out.println("Welcome to the Apple store!");
System.out.println("Whats your Balance?\n Note: You need more that 500£ to buy at least one Iphone.");
// placeholder for integer validation
int budget = input.nextInt();
balance += budget;
```

After the user inputs their balance, they get redirected to the menu of the program. The menu shows the models of the iPhones.



```
System.out.println("!=!=!=!=! MENU !=!=!=!=!\n");
for (int b = 0; b < dataArray.length; b++) {
        System.out.println(b + ". " +  dataArray[b].getModel());
}
//Printer
//Printer
System.out.println(defaultmessage);
```

The program also has declared a default message which states the options available to the user. This is a reusable message and can also be printed by inputting “options” in the menu.



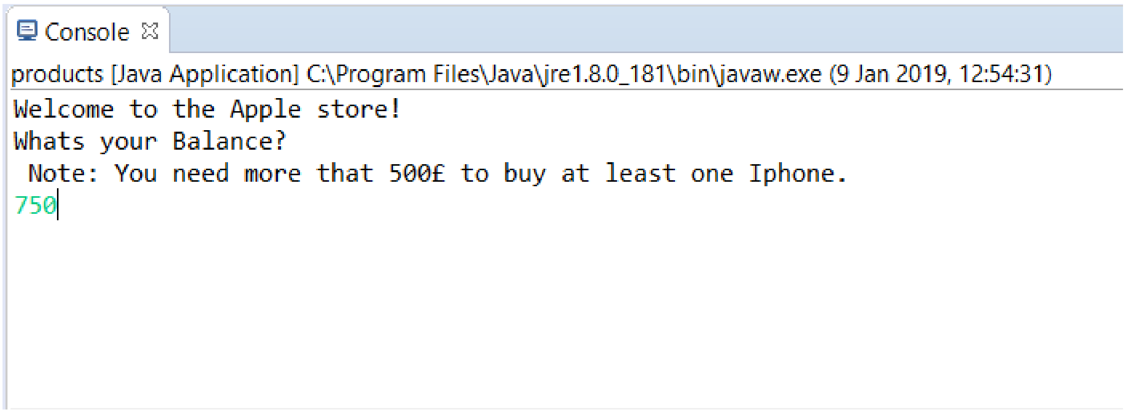
```
    public static String defaultmessage  = "What are your options: "
            + "\n You can input any Iphone model or the number on the list to view it's properties."
            + "\n 'budget' Used to check your balance and your affordable phones "
            + "\n 'christmas' 50£ discount to all products. Merry Christmas! "
            + "\n 'admin' Once used, You can use the comands 'set price' and 'set stock' "
            + "\n 'exit program' Used to exit the program."
            + "\n 'options' to access the options."
            + "\n You can only access those options in the menu, so before typing in any option, please type in 'menu' before doing so.";
```

The next section explains the functionality of each option.

The first is the budget. If you type in budget after you input your balance it will display your affordable phones depending on your balance and will allow you to buy the iPhones showed. After you purchase a product, your budget will change, and the list will be changed.

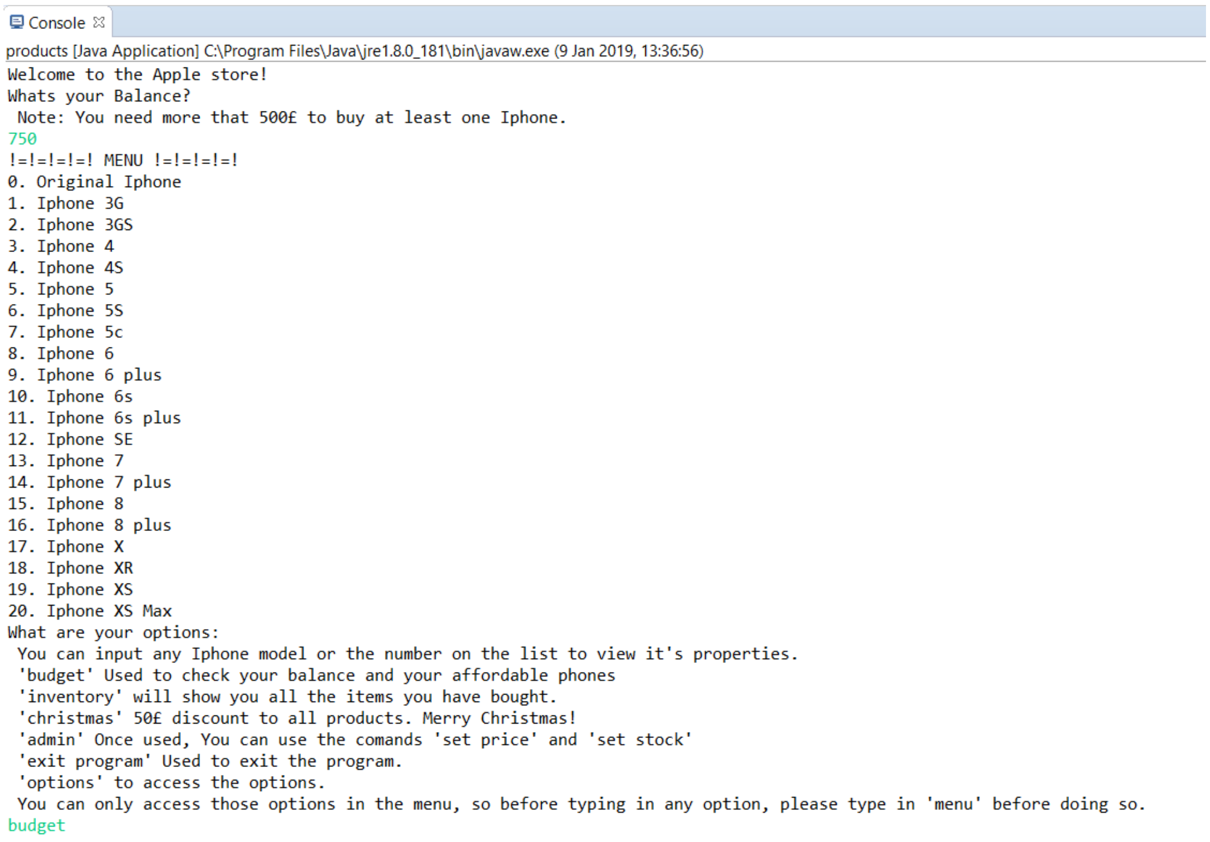
For an example, I am inputting that my balance is 750£

After I press enter, the menu will appear.



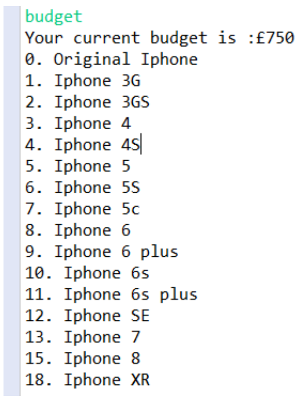
```
products [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (9 Jan 2019, 12:54:31)
Welcome to the Apple store!
Whats your Balance?
 Note: You need more that 500£ to buy at least one Iphone.
750
```

After that we must write “**budget**” to access the budget function.



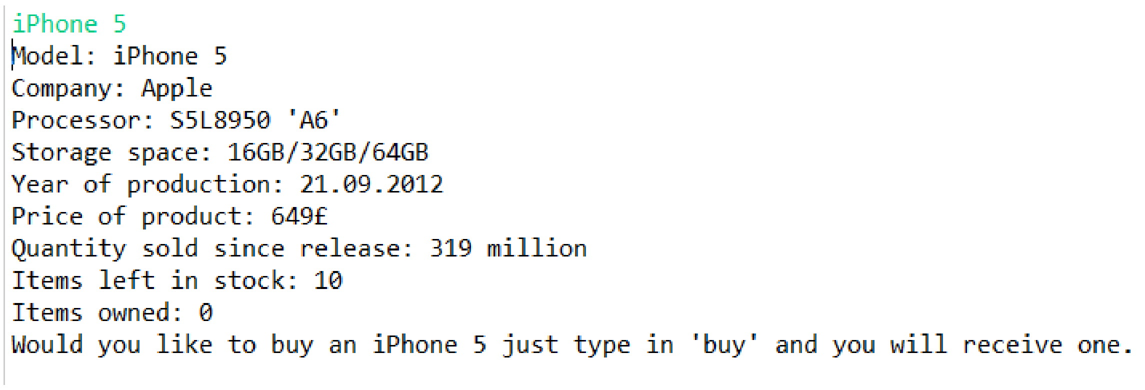
```
Console ☒
products [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (9 Jan 2019, 13:36:56)
Welcome to the Apple store!
Whats your Balance?
 Note: You need more that 500£ to buy at least one Iphone.
750
!=!=!=!=! MENU !=!=!=!=!
0. Original Iphone
1. Iphone 3G
2. Iphone 3GS
3. Iphone 4
4. Iphone 4S
5. Iphone 5
6. Iphone 5S
7. Iphone 5c
8. Iphone 6
9. Iphone 6 plus
10. Iphone 6s
11. Iphone 6s plus
12. Iphone SE
13. Iphone 7
14. Iphone 7 plus
15. Iphone 8
16. Iphone 8 plus
17. Iphone X
18. Iphone XR
19. Iphone XS
20. Iphone XS Max
What are your options:
 You can input any Iphone model or the number on the list to view it's properties.
 'budget' Used to check your balance and your affordable phones
 'inventory' will show you all the items you have bought.
 'christmas' 50£ discount to all products. Merry Christmas!
 'admin' Once used, You can use the comands 'set price' and 'set stock'
 'exit program' Used to exit the program.
 'options' to access the options.
 You can only access those options in the menu, so before typing in any option, please type in 'menu' before doing so.
budget
```

As you can see, it appears that our budget is 750 and these are all the iPhones we can currently afford. If we want to buy or an example the iPhone 5…



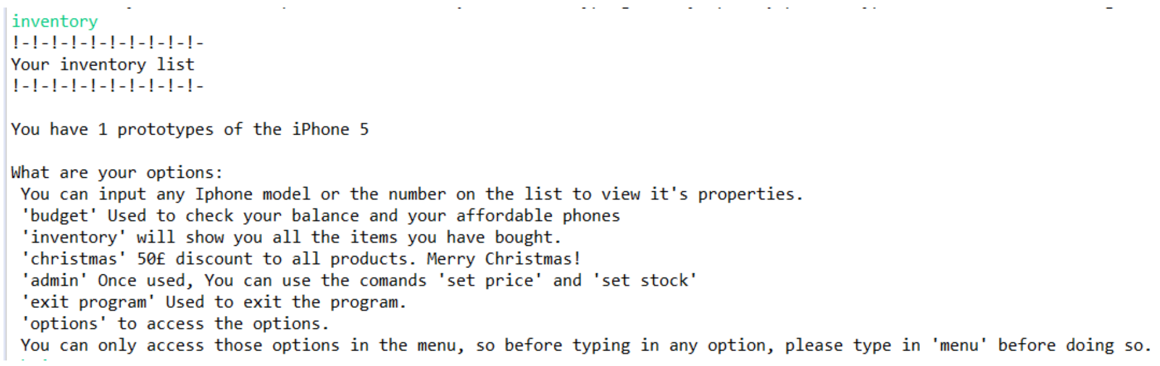
```
budget
Your current budget is :£750
0. Original Iphone
1. Iphone 3G
2. Iphone 3GS
3. Iphone 4
4. Iphone 4S
5. Iphone 5
6. Iphone 5S
7. Iphone 5c
8. Iphone 6
9. Iphone 6 plus
10. Iphone 6s
11. Iphone 6s plus
12. Iphone SE
13. Iphone 7
15. Iphone 8
18. Iphone XR
```

As we type in iPhone 5, the information about the product appears and in order to buy it, we need to type in “buy”



```
iPhone 5
Model: iPhone 5
Company: Apple
Processor: S5L8950 'A6'
Storage space: 16GB/32GB/64GB
Year of production: 21.09.2012
Price of product: 649£
Quantity sold since release: 319 million
Items left in stock: 10
Items owned: 0
Would you like to buy an iPhone 5 just type in 'buy' and you will receive one.
```

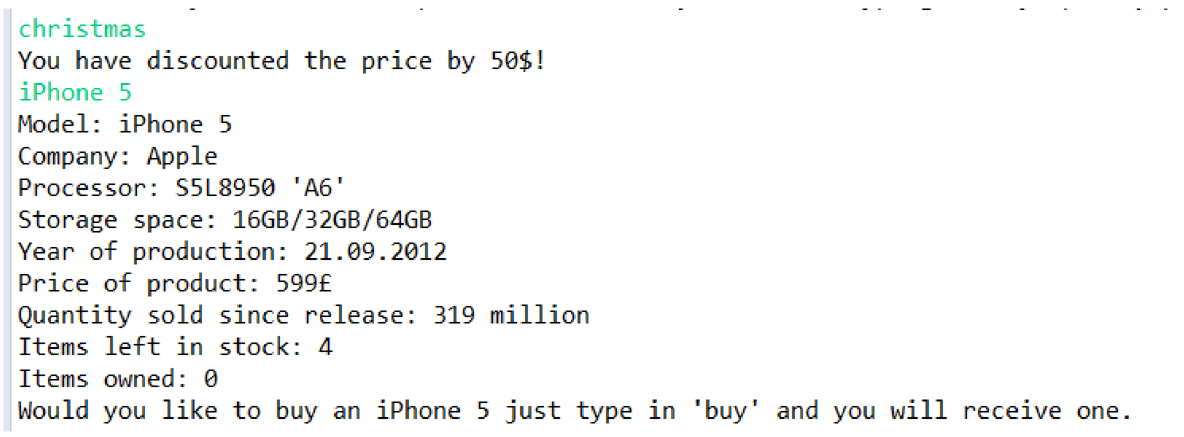
After we type in “menu” we are back at the menu and we can look over the next function which is the inventory. We simply type in inventory as it says in the option menu and see our inventory



```
inventory
!-!-!-!-!-!-!-!-!-!-
Your inventory list
!-!-!-!-!-!-!-!-!-!-

You have 1 prototypes of the iPhone 5

What are your options:
 You can input any Iphone model or the number on the list to view it's properties.
 'budget' Used to check your balance and your affordable phones
 'inventory' will show you all the items you have bought.
 'christmas' 50£ discount to all products. Merry Christmas!
 'admin' Once used, You can use the comands 'set price' and 'set stock'
 'exit program' Used to exit the program.
 'options' to access the options.
 You can only access those options in the menu, so before typing in any option, please type in 'menu' before doing so.
```

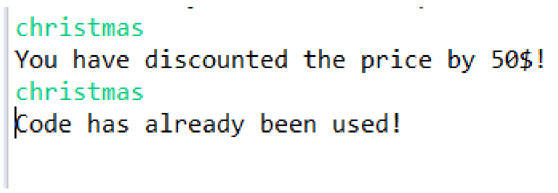
When you type in inventory you get your inventory list where you can see all the iPhones you have purchased.

The next function of the program is the “Christmas” discount coupon, which allows you to lower each of the iPhones price by 50£. When you are in the menu if you type in “Christmas” You will get the discount.



```
christmas
You have discounted the price by 50$!
iPhone 5
Model: iPhone 5
Company: Apple
Processor: S5L8950 'A6'
Storage space: 16GB/32GB/64GB
Year of production: 21.09.2012
Price of product: 599£
Quantity sold since release: 319 million
Items left in stock: 4
Items owned: 0
Would you like to buy an iPhone 5 just type in 'buy' and you will receive one.
```

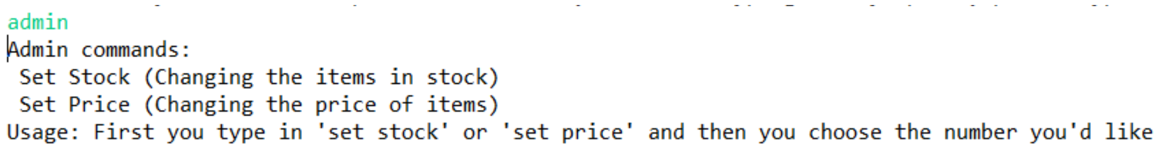
As you can see here, the “**Christmas”** function has been inputted and we can see that the iPhone 5, which is usually worth £649, now costs £50 less, which means that the code worked.



```
christmas
You have discounted the price by 50$!
christmas
Code has already been used!
```

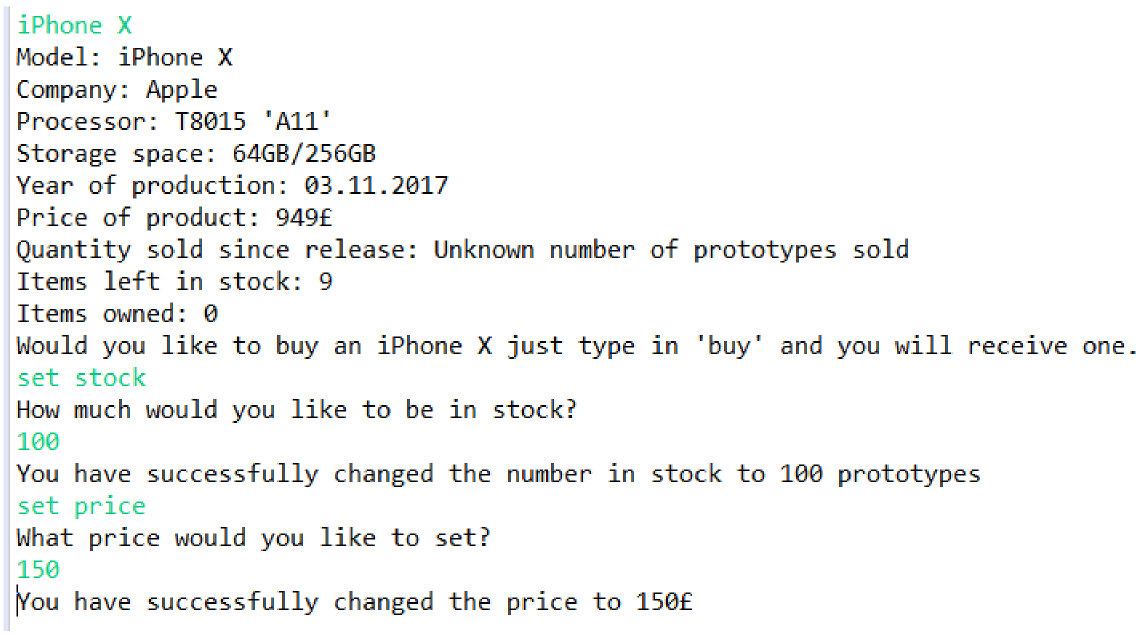
After the code has been used ones, it cannot be used again. If the price of the phone is lower than 50£, the price will be set to 0£.

Next, I will talk about the admin function. With this feature, the user can change the stock or the price of the iPhones simply but typing in “Set stock” or “Set price” and then typing in an integer to change it to. In order for the admin function to be enabled, once the user is inside the program, the user needs to type in the menu chat “admin” and it will be available, otherwise the commands would not be accessible.

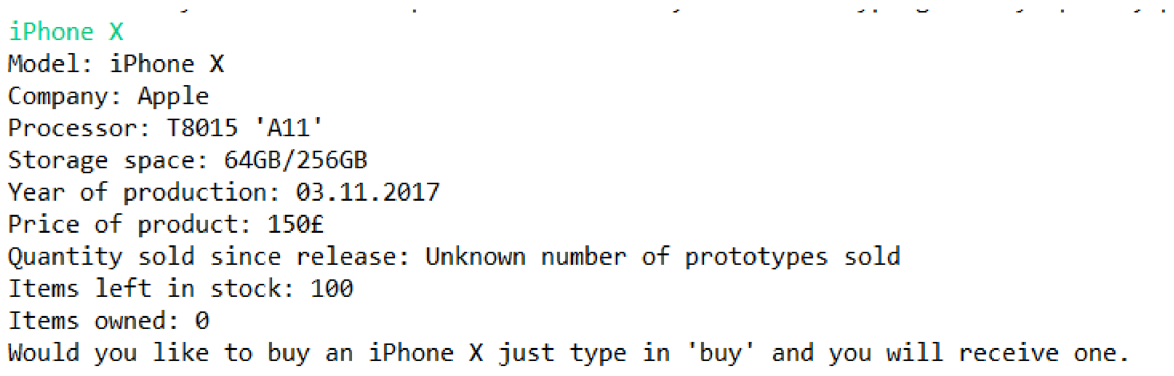

```
admin
Admin commands:
 Set Stock (Changing the items in stock)
 Set Price (Changing the price of items)
Usage: First you type in 'set stock' or 'set price' and then you choose the number you'd like
```

As you can see I have enabled the admin commands. Once I have done that I can go back to the menu and choose any iPhone and I can change its stock values and price amount.

For this example, I am going to choose the iPhone X, which is my personal phone.

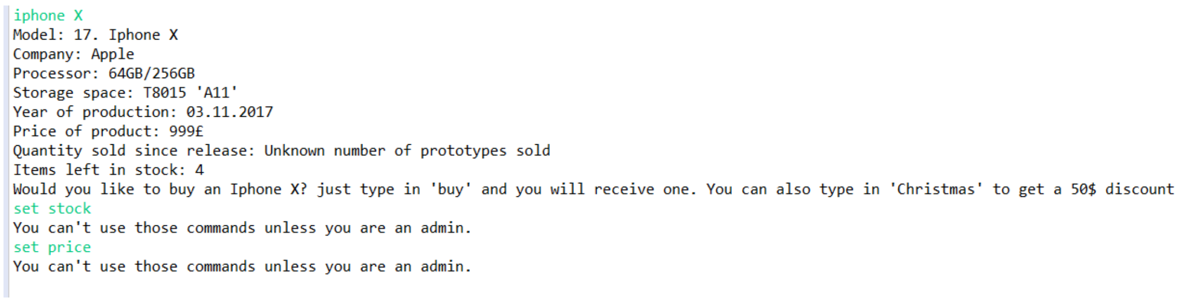

```
iPhone X
Model: iPhone X
Company: Apple
Processor: T8015 'A11'
Storage space: 64GB/256GB
Year of production: 03.11.2017
Price of product: 949£
Quantity sold since release: Unknown number of prototypes sold
Items left in stock: 9
Items owned: 0
Would you like to buy an iPhone X just type in 'buy' and you will receive one.
set stock
How much would you like to be in stock?
100
You have successfully changed the number in stock to 100 prototypes
set price
What price would you like to set?
150
You have successfully changed the price to 150£
```

After going into the menu accessing the admin commands and choosing to review the iPhone X, we set the number in stock to be 100 and the price to be 150£



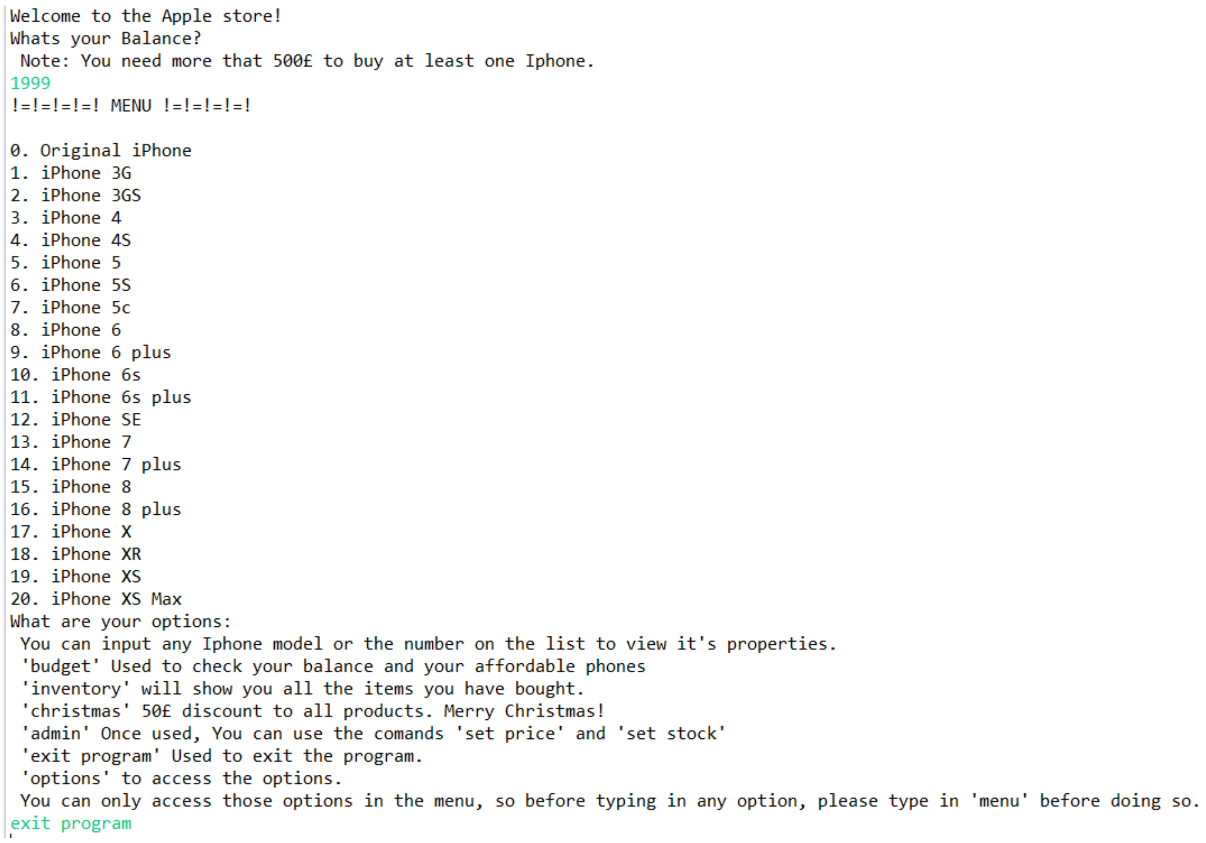
```
iPhone X
Model: iPhone X
Company: Apple
Processor: T8015 'A11'
Storage space: 64GB/256GB
Year of production: 03.11.2017
Price of product: 150£
Quantity sold since release: Unknown number of prototypes sold
Items left in stock: 100
Items owned: 0
Would you like to buy an iPhone X just type in 'buy' and you will receive one.
```

We can see that the values have changed. The stock has changed from “9” to “100” and the price has changed from “£949” to “£150”. However, if we haven’t typed in admin before trying to set the stock and price of the items, it would not be possible.



```
iphone X
Model: 17. Iphone X
Company: Apple
Processor: 64GB/256GB
Storage space: T8015 'A11'
Year of production: 03.11.2017
Price of product: 999£
Quantity sold since release: Unknown number of prototypes sold
Items left in stock: 4
Would you like to buy an Iphone X? just type in 'buy' and you will receive one. You can also type in 'Christmas' to get a 50$ discount
set stock
You can't use those commands unless you are an admin.
set price
You can't use those commands unless you are an admin.
```

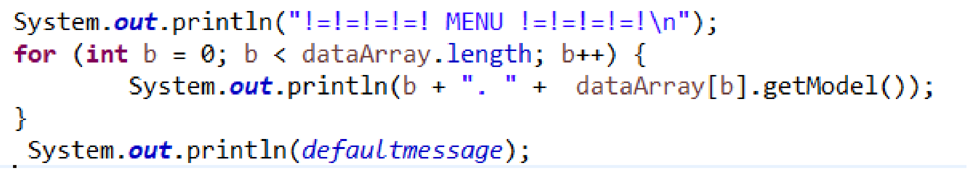
In order to exit the program, you need to type in “exit program” and inputting this, you will break the main loop that stops the whole program.



```
Welcome to the Apple store!
Whats your Balance?
 Note: You need more that 500£ to buy at least one Iphone.
1999
!=!=!=!=! MENU !=!=!=!=!

0. Original iPhone
1. iPhone 3G
2. iPhone 3GS
3. iPhone 4
4. iPhone 4S
5. iPhone 5
6. iPhone 5S
7. iPhone 5c
8. iPhone 6
9. iPhone 6 plus
10. iPhone 6s
11. iPhone 6s plus
12. iPhone SE
13. iPhone 7
14. iPhone 7 plus
15. iPhone 8
16. iPhone 8 plus
17. iPhone X
18. iPhone XR
19. iPhone XS
20. iPhone XS Max
What are your options:
 You can input any Iphone model or the number on the list to view it's properties.
 'budget' Used to check your balance and your affordable phones
 'inventory' will show you all the items you have bought.
 'christmas' 50£ discount to all products. Merry Christmas!
 'admin' Once used, You can use the comands 'set price' and 'set stock'
 'exit program' Used to exit the program.
 'options' to access the options.
 You can only access those options in the menu, so before typing in any option, please type in 'menu' before doing so.
exit program
```

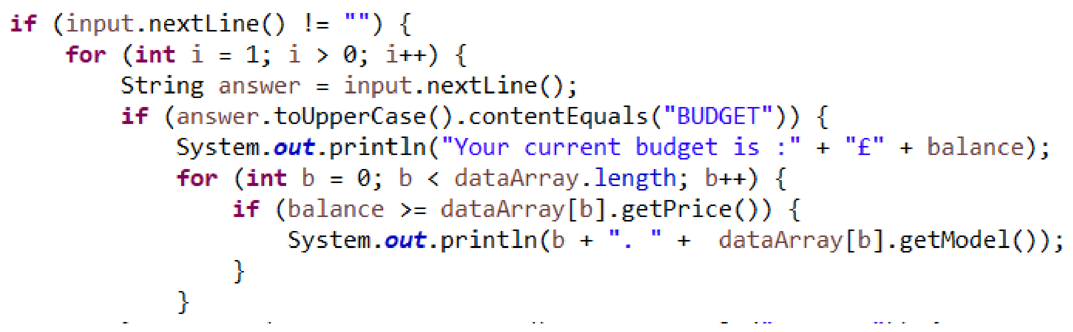
Now, in order to explain each function of the program, I will go by each one and explain how it works.

As you were previously informed, each iPhone is stored in an Array and without needing to go by every iPhone one by one, we use a for loop that does that for ourselves.



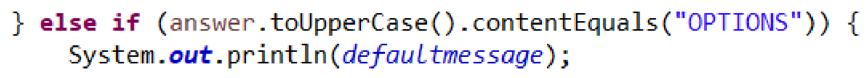
```
System.out.println("!=!=!=!=! MENU !=!=!=!=!\n");
for (int b = 0; b < dataArray.length; b++) {
        System.out.println(b + ". " +  dataArray[b].getModel());
}
 System.out.println(defaultmessage);
```

This is how our menu is being printed. The for loop goes by each dataArray and prints out the number in the list plus the model of the iPhones and we also have the default message, which informs us about the options of the program.



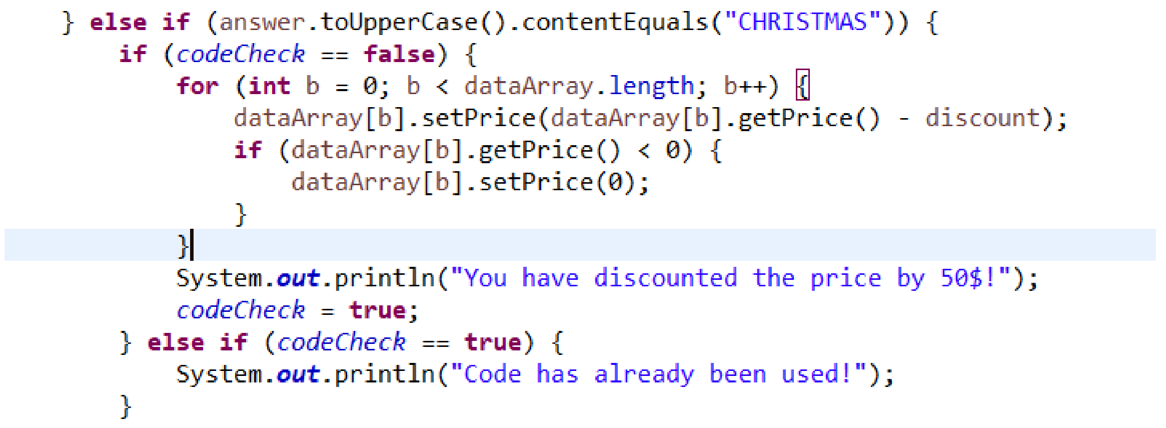
```
if (input.nextLine() != "") {
    for (int i = 1; i > 0; i++) {
        String answer = input.nextLine();
        if (answer.toUpperCase().contentEquals("BUDGET")) {
            System.out.println("Your current budget is :" + "£" + balance);
            for (int b = 0; b < dataArray.length; b++) {
                if (balance >= dataArray[b].getPrice()) {
                    System.out.println(b + ". " +  dataArray[b].getModel());
                }
            }
```

This is another function, which is the **budget**. The first loop is in an infinite for loop which constantly helps us exit the functions when needed, breaking the loop or continuing the loop. This infinite for loop also works for the **exit program** function. In order for **budget** to work, firstly the program prints out your current budget and then another for loop starts which states that if the balance is higher that all of the items' cost amounts, print those items.



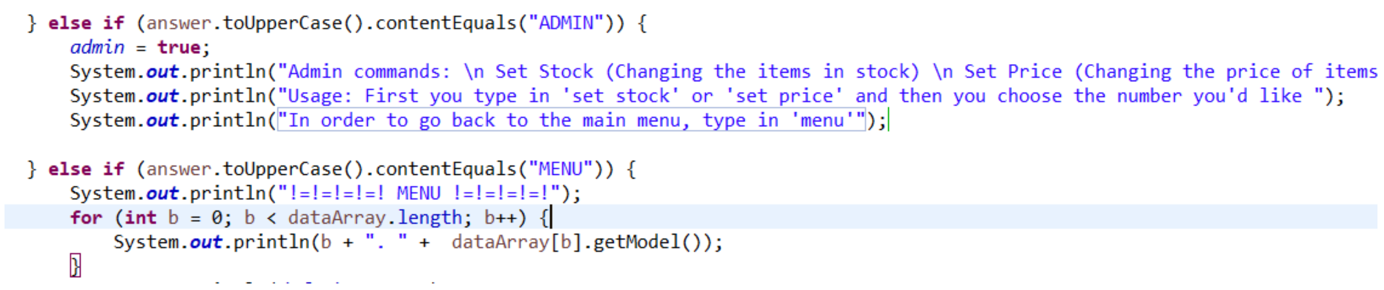
```
} else if (answer.toUpperCase().contentEquals("OPTIONS")) {
    System.out.println(defaultmessage);
```

The **options** function is the simplest one in the program. It doesn't require a for loop or such, it states that if the user inputs **options** in the menu, the defaultmessage appears, which is all the options, as previously mentioned.



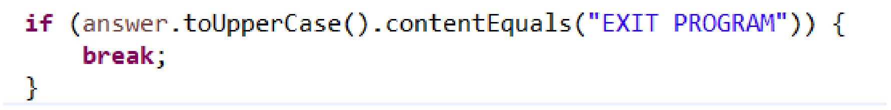
```
} else if (answer.toUpperCase().contentEquals("CHRISTMAS")) {
    if (codeCheck == false) {
        for (int b = 0; b < dataArray.length; b++) {
            dataArray[b].setPrice(dataArray[b].getPrice() - discount);
            if (dataArray[b].getPrice() < 0) {
                dataArray[b].setPrice(0);
            }
        }
        System.out.println("You have discounted the price by 50$!");
        codeCheck = true;
    } else if (codeCheck == true) {
        System.out.println("Code has already been used!");
    }
```

This function is the discount coupon for the program. It is used with a Boolean codeCheck, which is false when the program starts. The way it works is the program goes by each item with the help of the for loop and setting the price using the set method by getting the current price using the get method and subtracting it to the discount integer, which is £50. The program states that in case the price is lower that £50, and you use the coupon code, the price is automatically set to £0. If the code has already been used, the program prints out “Code has already been used”.



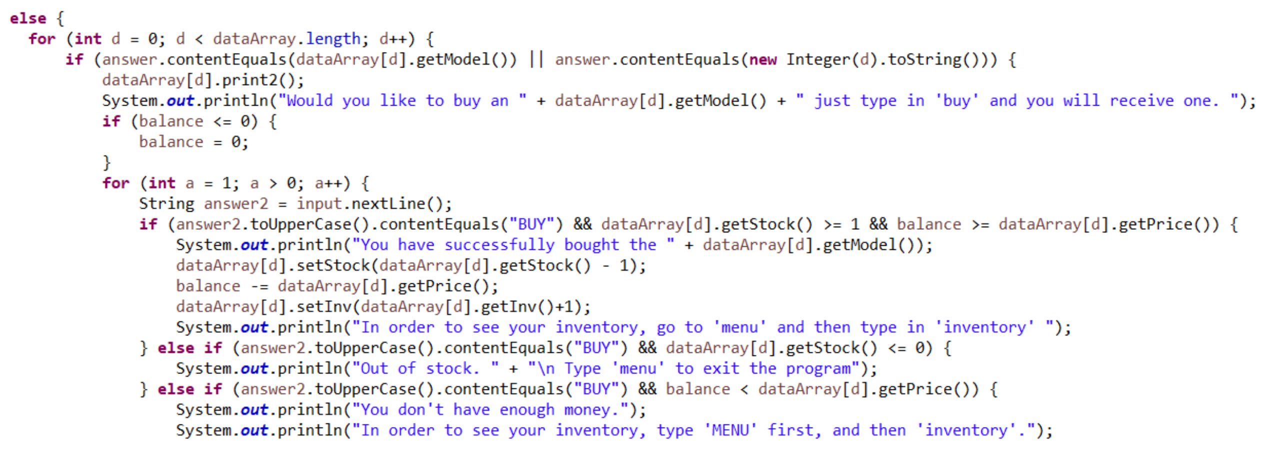
```
} else if (answer.toUpperCase().contentEquals("ADMIN")) {
    admin = true;
    System.out.println("Admin commands: \n Set Stock (Changing the items in stock) \n Set Price (Changing the price of items
    System.out.println("Usage: First you type in 'set stock' or 'set price' and then you choose the number you'd like ");
    System.out.println("In order to go back to the main menu, type in 'menu'");

} else if (answer.toUpperCase().contentEquals("MENU")) {
    System.out.println("!=!=!=!=! MENU !=!=!=!=!");
    for (int b = 0; b < dataArray.length; b++) {
        System.out.println(b + ". " +  dataArray[b].getModel());
    }
```

The next function is the **admin**. The coding of the “set stock” and “set price is located in the iPhone coding which I will later in this documentation explain. Once the code **admin** is inputted, the admin Boolean is set to true, which means that the admin commands are now available. If the user tries to “set stock” or “set price” without typing in **admin** in the menu first, the program won’t let that to happen. Once you type **admin** , in order to get back to the main menu, the user must input **menu**.



```java
if (answer.toUpperCase().contentEquals("EXIT PROGRAM")) {
    break;
}
```

In order to exit the program, the user needs to type in **exit program** and the main loop breaks, which causes the program to terminate.

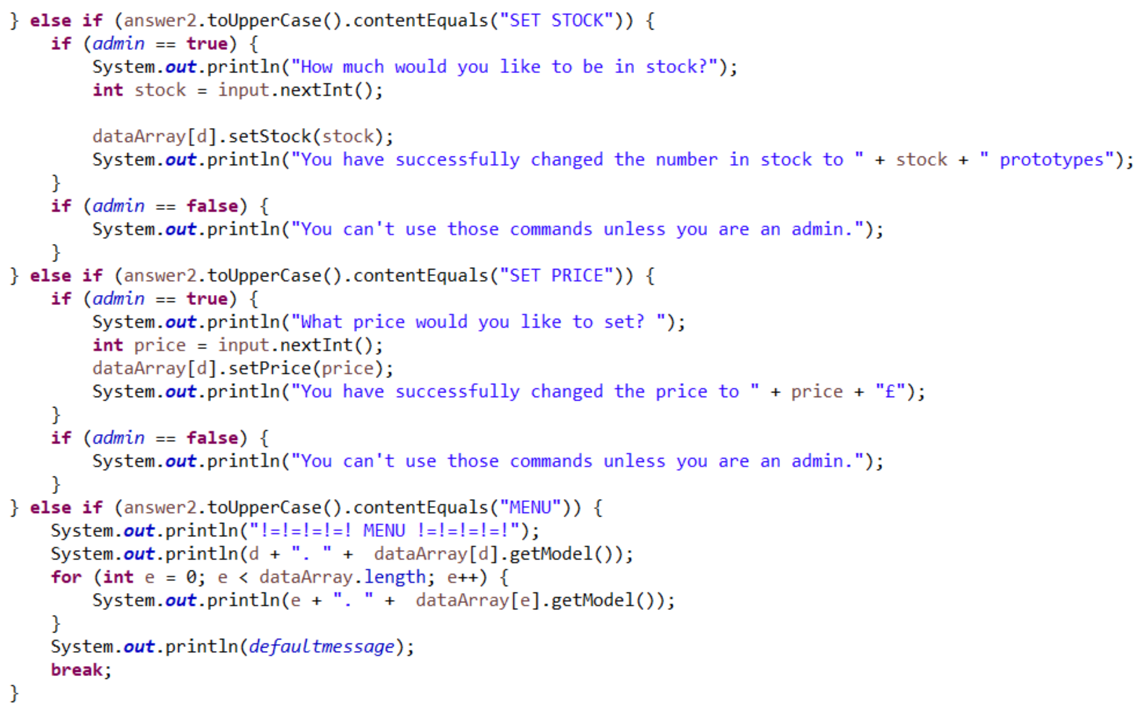


```java
else {
  for (int d = 0; d < dataArray.length; d++) {
      if (answer.contentEquals(dataArray[d].getModel()) || answer.contentEquals(new Integer(d).toString())) {
          dataArray[d].print2();
          System.out.println("Would you like to buy an " + dataArray[d].getModel() + " just type in 'buy' and you will receive one. ");
          if (balance <= 0) {
              balance = 0;
          }
          for (int a = 1; a > 0; a++) {
              String answer2 = input.nextLine();
              if (answer2.toUpperCase().contentEquals("BUY") && dataArray[d].getStock() >= 1 && balance >= dataArray[d].getPrice()) {
                  System.out.println("You have successfully bought the " + dataArray[d].getModel());
                  dataArray[d].setStock(dataArray[d].getStock() - 1);
                  balance -= dataArray[d].getPrice();
                  dataArray[d].setInv(dataArray[d].getInv()+1);
                  System.out.println("In order to see your inventory, go to 'menu' and then type in 'inventory' ");
              } else if (answer2.toUpperCase().contentEquals("BUY") && dataArray[d].getStock() <= 0) {
                  System.out.println("Out of stock. " + "\n Type 'menu' to exit the program");
              } else if (answer2.toUpperCase().contentEquals("BUY") && balance < dataArray[d].getPrice()) {
                  System.out.println("You don't have enough money.");
                  System.out.println("In order to see your inventory, type 'MENU' first, and then 'inventory'.");
```

Now I am going to talk about the iPhone menu, which is also located in a for loop. If the user inputs the model of the iPhone or the number in the list, the program will use the second print function and would ask the question if the user wants to **buy** the iPhone.

In order for the balance to not go to a negative amount, I have typed in that if the balance is lower than zero, the balance is automatically set to zero.

When the user inputs **buy** there's an if statement that states that if the stock of the item is higher or equal than one and the price is lower than or equal to the balance of the user, you have successfully purchased this iPhone. After you have purchased it, one stock is taken away from the **Stock** integer and the balance is reduced by the price amount of the current item. The program also adds one to the **inventory** function which I haven't explained yet, but later in the documentation will. There are 2 else if statements. One of them states that if the products doesn't have any items in stock, it prints that the item is out of stock. The other if statement states that if the user's balance is lower than the amount of the iPhone, the program prints out that the user doesn’t have enough money to purchase that item.

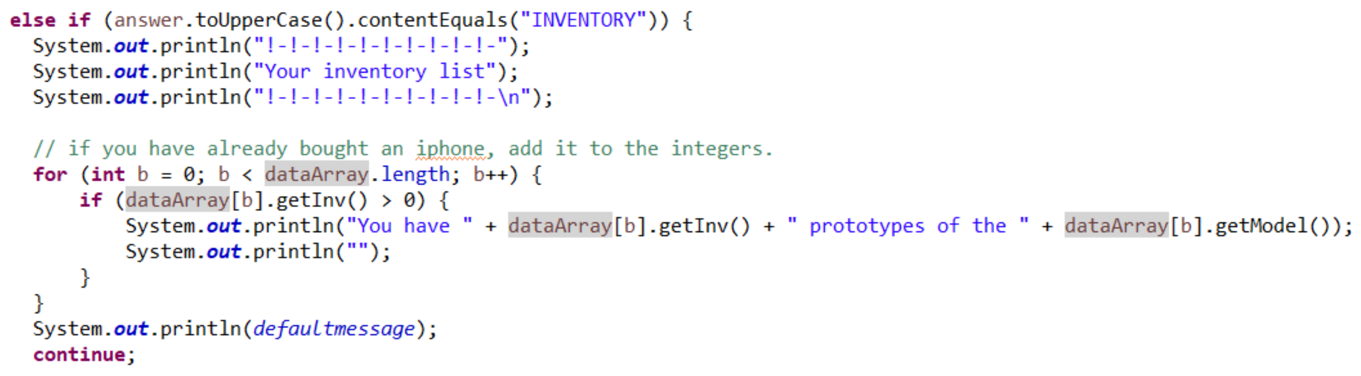


```
} else if (answer2.toUpperCase().contentEquals("SET STOCK")) {
    if (admin == true) {
        System.out.println("How much would you like to be in stock?");
        int stock = input.nextInt();

        dataArray[d].setStock(stock);
        System.out.println("You have successfully changed the number in stock to " + stock + " prototypes");
    }
    if (admin == false) {
        System.out.println("You can't use those commands unless you are an admin.");
    }
} else if (answer2.toUpperCase().contentEquals("SET PRICE")) {
    if (admin == true) {
        System.out.println("What price would you like to set? ");
        int price = input.nextInt();
        dataArray[d].setPrice(price);
        System.out.println("You have successfully changed the price to " + price + "£");
    }
    if (admin == false) {
        System.out.println("You can't use those commands unless you are an admin.");
    }
} else if (answer2.toUpperCase().contentEquals("MENU")) {
    System.out.println("!=!=!=!=! MENU !=!=!=!=!");
    System.out.println(d + ". " +  dataArray[d].getModel());
    for (int e = 0; e < dataArray.length; e++) {
        System.out.println(e + ". " +  dataArray[e].getModel());
    }
    System.out.println(defaultmessage);
    break;
}
```

Firstly we have the **set stock** function. If the admin Boolean is true, the program will ask how much you would like to be in stock and the program will set the stock to the current item with the inputted value thanks to the use of the set method. Although if the admin Boolean is false, the program will print out that the user is not authorized to use those commands.

The **set price** command is used on the same principle, but instead of changing the stock, the program is changing the price of the product.

In order to break out of the loop, we need to input **menu** which will send us to the main menu.



```
else if (answer.toUpperCase().contentEquals("INVENTORY")) {
  System.out.println("!-!-!-!-!-!-!-!-!-!-");
  System.out.println("Your inventory list");
  System.out.println("!-!-!-!-!-!-!-!-!-!-\n");

  // if you have already bought an iphone, add it to the integers.
  for (int b = 0; b < dataArray.length; b++) {
      if (dataArray[b].getInv() > 0) {
          System.out.println("You have " + dataArray[b].getInv() + " prototypes of the " + dataArray[b].getModel());
          System.out.println("");
      }
  }
  System.out.println(defaultmessage);
  continue;
```

This is the **inventory** function. After the user has bought any model of iPhone, the user can see their purchases by inputting **inventory** in the menu. This option is also controlled by a for loop that states that if the inv integer is a higher number than zero, the program prints out the number and the model. I have also added the defaultmessage to appear, so that the user would know their next step. After you enter the inventory, the program automatically transfers to the main menu of the program.