**CSC 3020 – Java Programming**

**Homework 5 – [Sayem Chowdhury]**

**25 points – Due April 4, 10am**

**Late deadline is April 6, 11:59pm, but 20% off**

//Sayem Chowdhury

HW#5

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

You’ve been hired by *Camping Critters* to write a Java console application that manages their product inventory. The application has the following two classes:

**Product.java**

Each object created from this class represents one product in their inventory and includes the following fields and methods:

**Fields**

● (static) productCount – count of all distinct products; initialize to 0 in declaration.

● (static) inventoryValue – total inventory value of all products; initialize to 0 in declaration.

● (static) inventoryCount – total count of all products in inventory; initialize to 0 in declaration.

● code – two-digit product code of the product.

● name – name of the product.

● cost – cost of the product in dollars.

● count – current count of the product in stock.

**Methods**

● A constructor with no parameters that sets the fields, respectively, to these values:

productCount = productCount + 1

code = -1

name = "(not set)"

cost = -1

count = -1

● A constructor with four parameters that sets the fields, respectively, to these values:

productCount = productCount + 1

inventoryValue = inventoryValue + (cost \* count)

inventoryCount = inventoryCount + count

code – set from parameter

name – set from parameter

cost – set from parameter

count – set from parameter

● Getter methods for each field (declare the getters for productCount, inventoryValue, and inventoryCount static).

● Setter methods for each field (declare the setters for productCount, inventoryValue, and inventoryCount static).

● *equals* method that compares the product codes from two objects for equality.

● *toString* method that returns instance variable values only.

**HW5.java**

This class contains the main method and uses the Product class to store product data. Create text file *ProductInventoryIn.txt*, paste the following data into it, and place the file in your project folder. It has the following file specification:

|  |  |  |
| --- | --- | --- |
| Field | Type | Start-End |
| Product code | integer | 1-7 |
| Product name | string | 8-31 |
| Product cost | real | 32-41 |
| Product quantity | integer | 42-51 |
| The file does not contain a header row. | | |

**ProductInventoryIn.txt**

**80 Daypack 110.00 50**

**81 Duffel Bag 35.00 60**

**82 Hammock 70.00 70**

**83 Cot 155.00 80**

**84 Tent 430.00 90**

**85 Stove 100.00 40**

**86 Cooler 350.00 30**

**87 Sleeping Bag 320.00 20**

**88 Blanket 140.00 10**

**89 Camp Chair 120.00 100**

Read the data from file *ProductInventoryIn.txt* into an array of Product objects called **products**. Note that to call a method for any array element object, you use, for example:

products[i].getCount()

Present the following menu to the user:

Camping Critters Menu

1 – Sell product

2 – Order product

3 – List product inventory

4 – Exit

Enter an option:

Here are what the options do:

● **Sell product –** use a validation loop to prompt for and get from the user the code of the product to be sold (it has to be a valid code). Then use a validation loop to prompt for and get from the user the quantity of the product to be sold. Insure that the quantity is not greater than the current inventory for that product. Update the following fields:

✓ (static) inventoryValue

✓ (static) inventoryCount

✓ count for the product

Print a "product sold" message that includes:

✓ Code

✓ Quantity

✓ Revenue from the sale

Format the message in two columns with the first column containing a label and the second column containing a value. Format real numbers to two decimal places.

● **Order Product –** use a validation loop to prompt for and get from the user the code of the product to be ordered (it has to be a valid code). Then use a validation loop to prompt for and get from the user the quantity of the product to be ordered. Insure that the quantity is greater than zero. Update the following fields:

✓ (static) inventoryValue

✓ (static) inventoryCount

✓ count for the product

Print a "product ordered" message that includes:

✓ Code

✓ Quantity ordered

✓ Cost of the order

Format the message in two columns with the first column containing a label and the second column containing a value. Format real numbers to two decimal places.

● **List product inventory** shows all product data in formatted columns. It then lists the product count, inventory value, and inventory count.

● **Exit** closes the menu.

Java doesn't handle multiple keyboard objects gracefully so declare one keyboard object as a field (global) and close it at the end of method *main*. Continue to process menu options until the user enter 4. Then write the data to file *ProductInventoryOut.txt* in the same layout as the input file. Use these menu options and inputs for your last run:

**Option Code Quantity**

**3**

**1 80 10**

**3**

**2 81 40**

**3**

**1 82 20**

**3**

**2 83 25**

**3**

**1 10,84 100,10**

**3**

**4**

**Product.java**

*[your Product.java class here*

*//---------------------------------------*

//======================================================================

// Class: <Product > //Product.java

// Description:

// <This class represents one product in the inventory and includes the product informations code,

// Name, cost, count, Also setter and getter methods to set and get information of the class objects>

//

//======================================================================

//Package

**package** HW5;

//java implemented class product for class HW5.java

**public** **class** Product {

//Fields / Instant Variables

//----------------------------------

//private static variables of the class

**private** **static** **int** *productCount*=0;

**private** **static** **double** *inventoryValue*=0;

**private** **static** **int** *inventoryCount*=0;

//private instant variables of the class

**private** **int** code;

**private** String name;

**private** **double** cost;

**private** **int** count;

//default constructor

**public** Product()

{

*productCount* = *productCount* + 1;

code = -1;

name = "(not set)";

cost = -1;

count = -1;

}

//constructor with parameters/arguments

**public** Product(**int** code,String name, **double** cost,**int** count) //:code (code),name (name),cost (cost),count(count)

{

//initializing static variables

*productCount* = *productCount* + 1;

*inventoryValue* = *inventoryValue* + (cost \* count);

*inventoryCount* = *inventoryCount* + count;

//----------------------------

//initializing instant variables

**this**.code = code;

**this**.name = name;

**this**.cost = cost;

**this**.count = count;

}

//Methods

//all setter methods

//==============================================================================================================

//for static variables

**public** **void** setProductCount(**int** PdC)

{

*productCount*=PdC;

}

**public** **void** setInventoryValue(**double** InV)

{

*inventoryValue*=InV;

}

**public** **void** setInventoryCount(**int** InC)

{

*inventoryCount*=InC;

}

//-----------------------------------------------

//for instant variables

**public** **void** setCode(**int** code)

{

**this**.code=code;

}

**public** **void** setName(String name)

{

**this**.name=name;

}

**public** **void** setCost(**double** cost)

{

**this**.cost=cost;

}

**public** **void** setCount(**int** count)

{

**this**.count=count;

}

//=================================================================================

//all getter methods

// for Static variables

**public** **int** getProductCount()

{

**return** *productCount*;

}

**public** **double** getInventoryValue()

{

**return** *inventoryValue*;

}

**public** **int** setInventoryCount()

{

**return** *inventoryCount*;

}

//--------------------------------

//for instant variables

**public** **int** getCode()

{

**return** code;

}

**public** String getName()

{

**return** name;

}

**public** **double** getCost()

{

**return** cost;

}

**public** **int** getCount()

{

**return** count;

}

//=============================================================================

//------------------------------------------------------------------

//equals and toString methods

//------------------------------------------------------------------

**public** **boolean** equals (Product b)

{

**if** (**this**.code==b.code && **this**.name.equalsIgnoreCase(b.name) && **this**.cost == b.cost && **this**.count==b.count)

**return** **true**;

**else**

**return** **false**;

}

**public** String toString ()

{

**return** "Code: " + code + "\nName: " + name + "\nCost: " + cost +"\nCount: " + count;

}

//---------------------------------------------------------------------------------------

//Method that used in HW5 class to update the sale after make a product sale

//--------------------------------------------------------------------------------------

**public** **void** updateSale(**int** q1)

{

//(static) inventoryValue

//(static) inventoryCount

//count for the product

*inventoryValue*=*inventoryValue*-(cost\*q1);

*inventoryCount*=*inventoryCount*-q1;

count=count-q1;

}

//----------------------------------------------------------------------------------------------

//Method that used in HW5 class to update the ordered product info after make a product ordered

//-----------------------------------------------------------------------------------------------

**public** **void** updateordered(**int** q1)

{

*inventoryValue*=*inventoryValue*+(cost\*q1);

*inventoryCount*=*inventoryCount*+q1;

count=count+q1;

}

//

//---------------------------------------------------------------------------------------

//Method that used in HW5 class to print the Total product count, inventory value, and inventory count.

//--------------------------------------------------------------------------------------

**public** **void** print\_l()

{

//System.out.println("\nProduct Inventory Total Product count and Value :");

System.***out***.printf("%n%s%d%n%s%.2f%n%s%d%n", "Product Count:", *productCount*,

"Inventory Value:$",*inventoryValue*, "Inventory Count:", *inventoryCount*);

}

// Function increment the product counted

**public** **void** incrementProductCount()

{

*productCount*=*productCount*+1;

}

}

//-----------------------------------------------------------------------------------------------

*]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**HW5.java**

*[your HW5.java class here*

*//---------------------------------------*

*//======================================================================*

*//*

*// Title: <HW5.java>*

*// Course: CSC 3020*

*// Homework: <#5>*

*// Author: <Sayem Chowdhury>*

*// Date: <3/31/2011>*

*// Description:*

*// <A Java console application that manages product inventory. >*

*//*

*//======================================================================*

*//Package Name*

*package HW5;*

*//import Classes*

*import java.io.FileInputStream;*

*import java.io.FileNotFoundException;*

*import java.io.\*;*

*import java.util.Scanner;*

*//Class Hw5.java*

*public class HW5 {*

*//declare variables*

*public static String output\_file = "ProductInventoryOut.txt";*

*static int dataCount =0;*

*public static Scanner KBD=new Scanner(System.in);*

*//--------------------------------------------------------*

*// function that sell the product*

*public static void sellproduct(Product [] pd)*

*{*

*// validate loop*

*int pcode;*

*int q;*

*int pn = 0;*

*System.out.println("\nPlease Enter the product code you want to sale:");*

*pcode=KBD.nextInt();*

*//String garbage=KBD.nextLine();*

*boolean b=false;*

*for (int i=0;i<pd.length;i++)*

*{*

*if (pd[i].getCode()==pcode)*

*{*

*pn=i; // saving index of the product found in variable pn*

*b=true;*

*}*

*}*

*if (b==true)*

*{*

*System.out.println("\nPlease enter the quantity of product you want to sell:");*

*//taking input from the user the number of product to sale*

*q=KBD.nextInt();*

*//validating quantity*

*if(q!=0 && q<=pd[pn].getCount())*

*{*

*pd[pn].updateSale(q);// call function that declare in class product to sale the product*

*//Print a "product sold" message that includes: Code ,Quantity, Revenue from the sale,*

*//(static) inventoryValue,(static) inventoryCount and count for the product*

*System.out.println("\nProduct Sold");*

*System.out.printf("%s%d%n%s%d%n%s%.2f%n", "Code:", pcode, "Quantity:",q,*

*"Revenue from the Sale:$", (pd[pn].getCost()\*q));*

*}*

*else*

*{*

*System.out.println("Invalid Quantity or Number of Quantity is out of Stack");*

*}*

*}*

*else*

*{*

*System.out.println("Sorry! Product code not Found");*

*}*

*}*

*//---------------------------------------------------------------------------------------------*

*//Function that take ordered and update the ordered*

*public static void orderproduct(Product [] pd)*

*{*

*//declare local variables*

*int pcode;*

*int q;*

*int pn = 0;*

*System.out.println("Please Enter the product code you want to order: ");*

*pcode=KBD.nextInt(); // user input*

*//String garbage=KBD.nextLine();*

*boolean b=false;*

*for (int i=0;i<pd.length;i++)*

*{*

*if (pd[i].getCode()==pcode)*

*{*

*pn=i;*

*b=true;*

*}*

*}*

*if (b==true) // product code matched to ordered*

*{*

*System.out.println("Please enter the quantity of product you want to Ordered: ");*

*q=KBD.nextInt();*

*if(q!=0 )*

*{*

*pd[pn].updateordered(q);*

*System.out.println("\nProduct ordered");*

*System.out.printf("%s%d%n%s%d%n%s%.2f%n", "Code:", pcode, "Quantity Ordered:",q,*

*"Cost Of the Ordered:$", (q\*pd[pn].getCost()));*

*}*

*else*

*{*

*System.out.println("Sorry! Ordered not Received, Invalid Quantity you Entered");*

*}*

*}*

*else*

*{*

*System.out.println("Sorry! Product code not found");*

*}*

*}*

*//Function that,List product inventory / shows all product data in formatted columns.*

*public static void Print\_p\_data(Product[] pdts)*

*{*

*System.out.println("\*\*\*Product Inventory\*\*\*");*

*System.out.println("-----------------------------------------------------------------------------");*

*System.out.printf("%-18s %-15s %15s %10s", "<Product Code> ||","<Product Name> ||",*

*" <Product Cost> ||"," <Product Quantity>\n");*

*System.out.println("=============================================================================");*

*for(int i=0;i<pdts.length;i++)*

*{*

*System.out.printf("%-18d %-14s%15.2f %15d",pdts[i].getCode(),pdts[i].getName(),*

*pdts[i].getCost(),pdts[i].getCount());*

*System.out.println("");*

*//System.out.println(pdts[i].getCode()+"\t"+pdts[i].getName()+"\t"+pdts[i].getCost()+"\t"+*

*//pdts[i].getCount());*

*}*

*}*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*//Function that Write Inventory Data into file*

*//---------------------------------------------------------------*

*public static void Wrt\_Str\_UBW(Product[]pdts) //write string using buffer writer*

*throws IOException {*

*BufferedWriter writer = new BufferedWriter(new FileWriter(output\_file));*

*for(int i=0;i<pdts.length;i++)*

*{*

*writer.write(String.format("%-18d %-14s %15.2f %15d%n",*

*pdts[i].getCode(),pdts[i].getName(),pdts[i].getCost(),pdts[i].getCount()));*

*}*

*writer.flush();*

*writer.close();*

*}*

*//-----------------------------------------------------------------------------------------------*

*//Function will display the menu for user*

*//---------------------------------------------------------------------------------------------------*

*public static void Show\_Menu(Product [] pr) throws IOException*

*{*

*while(true)*

*{*

*System.out.println("\*\*\*Camping Critters Menu\*\*\*");*

*System.out.println("---------------------------");*

*System.out.println("1-> Sell Product");*

*System.out.println("2-> Ordered Product");*

*System.out.println("3-> List Product Inventory");*

*System.out.println("4-> Exit \n");*

*//Scanner KBD =new Scanner(System.in);*

*int choice;*

*System.out.println("Enter an option: ");*

*choice=KBD.nextInt();*

*if(choice>=1 && choice<=3)*

*{*

*if (choice==1)*

*{*

*sellproduct(pr); //function call to sale product*

*System.out.println(" ");*

*Wrt\_Str\_UBW(pr);*

*}*

*else if (choice==2)*

*{*

*orderproduct(pr); //function call to ordered product*

*System.out.println(" ");*

*Wrt\_Str\_UBW(pr);*

*}*

*else if(choice==3)*

*{*

*Print\_p\_data(pr);//function call to print inventory*

*pr[1].print\_l();*

*System.out.println(" ");*

*Wrt\_Str\_UBW(pr);*

*}*

*}*

*else if (choice==4)*

*{*

*KBD.close(); //close keyboard*

*Wrt\_Str\_UBW(pr);// function call to Write in file*

*break;*

*}*

*else*

*{*

*System.out.println("sorry Invalid Entry");*

*}*

*}*

*}*

*//----------------------------------------------------------*

*//----------[Main Function]------------------------------*

*//-------------------------------------------------------------*

*public static void main(String[] args) throws IOException {*

*// array of the products / object type*

*Product [] products = new Product [10];*

*//file reading*

*// Declaring variables*

*Scanner fileIn = null;*

*String line;*

*// Attempt to open input file*

*//try catch Block*

*try//Exception handler*

*{*

*// Assign external file to file handle*

*fileIn = new Scanner(new FileInputStream("ProductInventoryIn.txt"));*

*// Loop to read data*

*//dataCount = 0;*

*while (fileIn.hasNextLine())*

*{*

*// Read each line*

*line = fileIn.nextLine();*

*{*

*products[dataCount]=new Product (Integer.parseInt(line.substring(0,5).trim()),*

*(line.substring(7,20).trim()),Double.parseDouble(line.substring(35,42).trim()),*

*Integer.parseInt(line.substring(48,51).trim()));*

*dataCount = dataCount + 1;//updating index*

*//products[0].incrementProductCount();*

*}*

*}*

*fileIn.close(); //close the file*

*}*

*// Handle file error*

*catch (FileNotFoundException e) //catch Block*

*{*

*System.out.println("Error: file '" + "ProductInventoryIn.txt" +*

*"' not found.");*

*System.out.println("Default folder: " +*

*System.getProperty("user.dir"));*

*System.out.println("Error message:\n" + e.getMessage());*

*}*

*// writing file before update information*

*Wrt\_Str\_UBW(products);*

*Show\_Menu(products); // function call to show menu*

*Wrt\_Str\_UBW(products);*

*}*

*}*

*// -------------------------------------------------------------------------------------------*

*]\**

**If possible, format your code like this:**

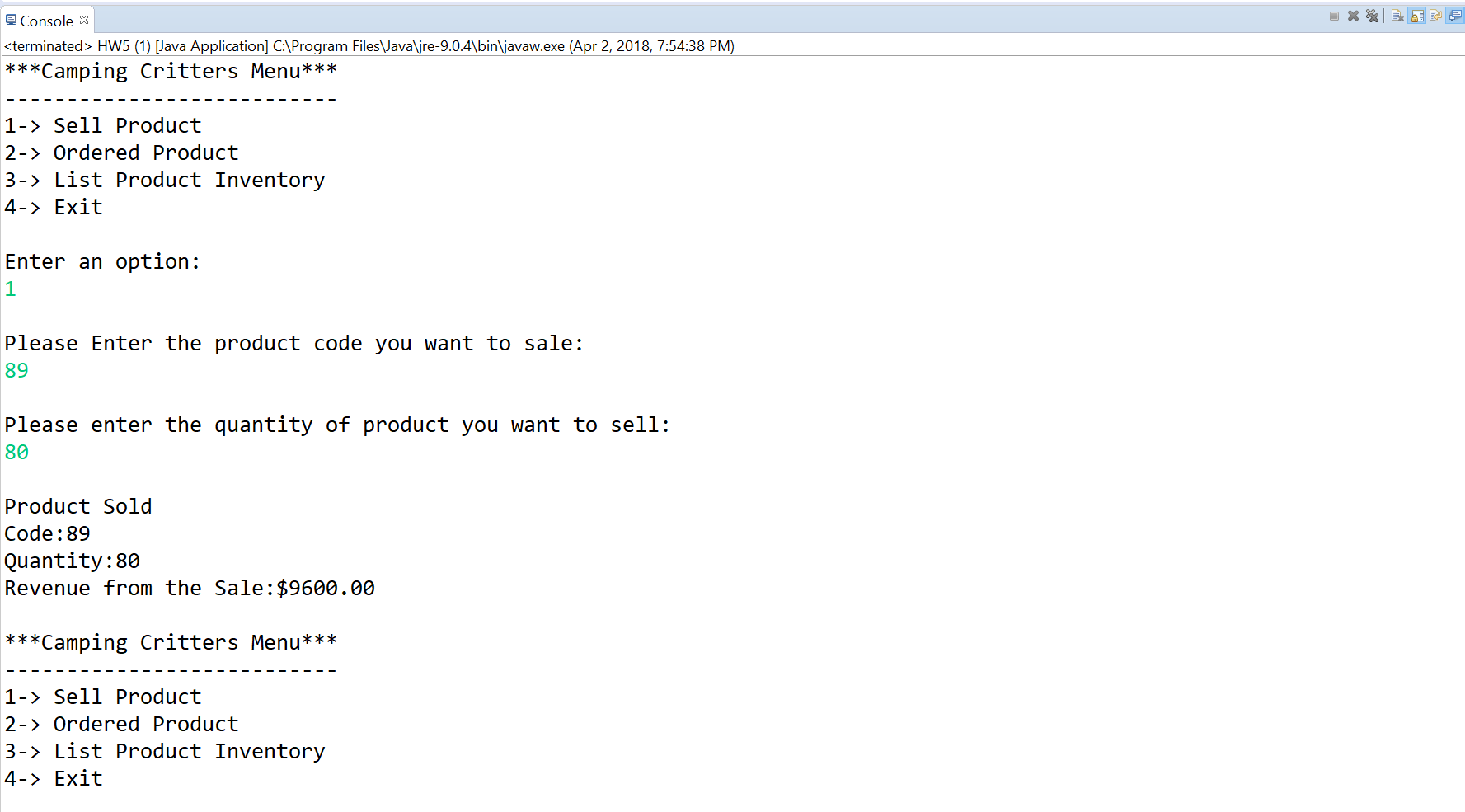
**Font “Courier New”**

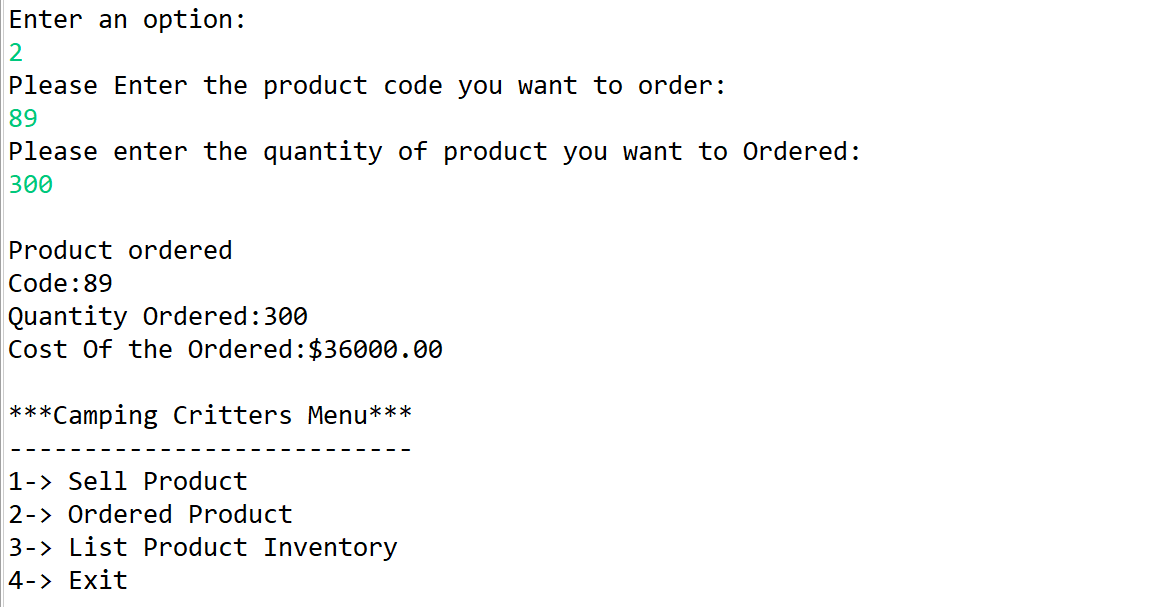
**Font size “9”**

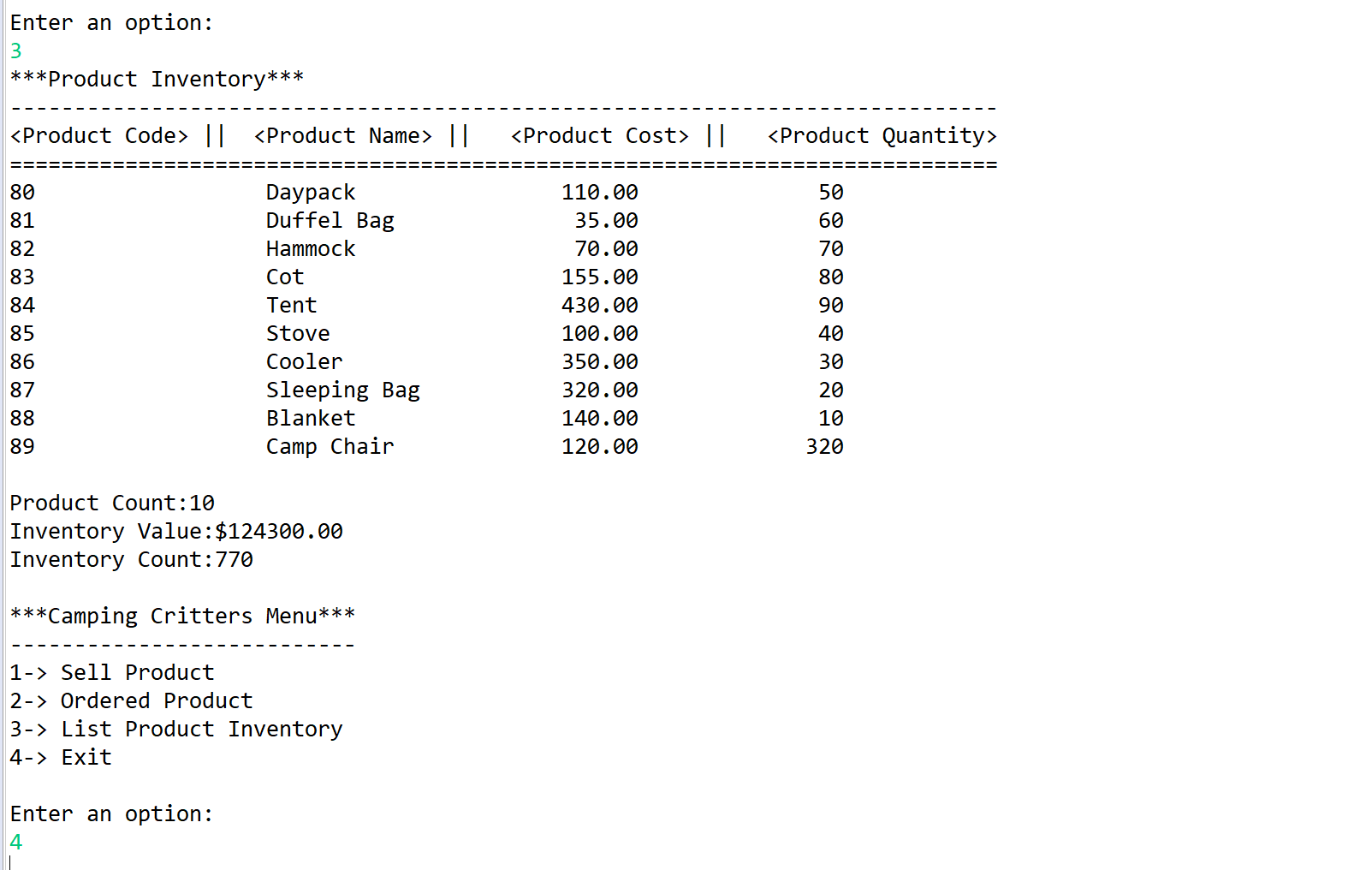
**Bold**

**Program output**

*[your program output here (just the last two screens)*



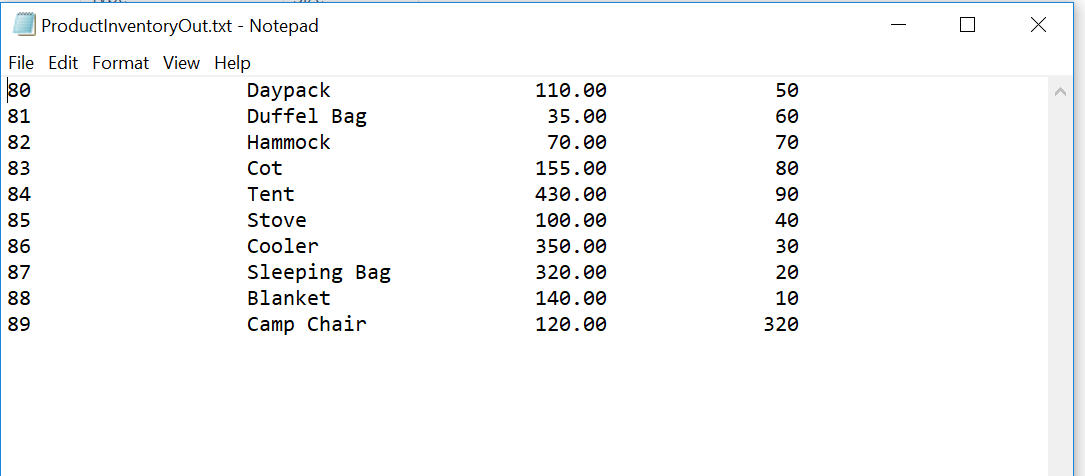




*]\*\**

**ProductInventoryOut.txt**

*[your ProductInventoryOut.txt file here*



*]\*\*\**

\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.

\*\*\* **Copying-and-pasting text file contents to a Word document**

1) From the text editor screen, maximize the window.

2) From the window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.