**CSC 3020 – Java Programming**

**Homework 5 – [your name]**

**25 points – Due April 4, 10am**

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

**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**

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

**//**

**// Class: Product**

**// Description:**

**// This class represents a product.**

**//**

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

**package wsu.HW05\_01;**

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

**// class Product**

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

**public class Product**

**{**

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

**// Fields**

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

**private static int productCount = 0;**

**private static double inventoryValue = 0;**

**private static int inventoryCount = 0;**

**private int code;**

**private String name;**

**private double cost;**

**private int count;**

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

**// Constructors**

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

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

**// No-parameter constructor**

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

**public Product()**

**{**

**productCount = productCount + 1;**

**code = -1;**

**name = "(not set)";**

**cost = -1;**

**count = -1;**

**}**

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

**// Four-parameter constructor**

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

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

**{**

**productCount = productCount + 1;**

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

**inventoryCount = inventoryCount + count;**

**this.code = code;**

**this.name = name;**

**this.cost = cost;**

**this.count = count;**

**}**

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

**// Methods**

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

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

**// get methods**

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

**public static int getProductCount()**

**{**

**return productCount;**

**}**

**public static double getInventoryValue()**

**{**

**return inventoryValue;**

**}**

**public static int getInventoryCount()**

**{**

**return inventoryCount;**

**}**

**public int getCode()**

**{**

**return code;**

**}**

**public String getName()**

**{**

**return name;**

**}**

**public double getCost()**

**{**

**return cost;**

**}**

**public int getCount()**

**{**

**return count;**

**}**

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

**// set methods**

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

**public static void setProductCount(int productCount)**

**{**

**Product.productCount = productCount;**

**}**

**public static void setInventoryValue(double inventoryValue)**

**{**

**Product.inventoryValue = inventoryValue;**

**}**

**public static void setInventoryCount(int inventoryCount)**

**{**

**Product.inventoryCount = inventoryCount;**

**}**

**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;**

**}**

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

**// equals and toString methods**

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

**public boolean equals(Product b)**

**{**

**if (this.code == b.code)**

**return true;**

**else**

**return false;**

**}**

**public String toString ()**

**{**

**return "Code: " + code +**

**"\nName: " + name +**

**"\nCost ($): " + String.format("%,1.2f", cost) +**

**"\nCount: " + count;**

**}**

**}**

**HW5.java**

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

**//**

**// Title: Camping Critters**

**// Course: CSC 3020**

**// Homework: 5-1**

**// Author: Dan Ouellette**

**// Date: 4 April 2018**

**// Description:**

**// This Java application manages product inventory for Camping**

**// Critters. It holds the following information for each product:**

**// -Product code**

**// -Product name**

**// -Product cost**

**// -Product count**

**// It also holds the following summary information:**

**// -Total product count**

**// -Total inventory value**

**// -Total inventory count**

**// The application reads information from text file**

**// ProductInventoryIn.txt. It presents 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' prompts for and gets from the user the product code**

**// and quantity of the product to be sold. It then updates the**

**// product inventory and summary information.**

**// -'Order product' prompts for and gets from the user the product**

**// code and quantity of the product to be ordered. It then updates**

**// the product inventory and summary information.**

**// -'List product inventory' lists all product inventory data.**

**// -'Exit' ends the application.**

**// The application then writes the data to file ProductInventoryIn.txt.**

**//**

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

**package wsu.HW05\_01;**

**// Import classes**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.PrintWriter;**

**import java.util.Scanner;**

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

**// class HW05\_01**

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

**public class HW05\_01**

**{**

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

**// Constants**

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

**// Declare file constants**

**public static final String FILE\_NAME\_IN = "ProductInventoryIn.txt";**

**public static final String FILE\_NAME\_OUT = "ProductInventoryOut.txt";**

**public static final int PRODUCTS = 10;**

**public static final int COL1\_START = 0;**

**public static final int COL1\_END = 6;**

**public static final int COL2\_START = 7;**

**public static final int COL2\_END = 30;**

**public static final int COL3\_START = 32;**

**public static final int COL3\_END = 41;**

**public static final int COL4\_START = 42;**

**public static final int COL4\_END = 51;**

**// Declare formatting constants**

**public static final String COLFMT1 = "%-" +**

**(COL1\_END - COL1\_START + 1) + "d";**

**public static final String COLFMT2 = "%-" +**

**(COL2\_END - COL2\_START + 1) + "s";**

**public static final String COLFMT3 = "%" +**

**(COL3\_END - COL3\_START + 1) + ".2f";**

**public static final String COLFMT4 = "%" +**

**(COL4\_END - COL4\_START + 1) + "d";**

**public static final String COLFMTS1 = "%-" + (COL1\_END + 1) + "s";**

**public static final String COLFMTS2 = "%-" +**

**(COL2\_END - COL2\_START + 1) + "s";**

**public static final String COLFMTS3 = "%" +**

**(COL3\_END - COL3\_START + 1) + "s";**

**public static final String COLFMTS4 = "%" +**

**(COL4\_END - COL4\_START + 1) + "s";**

**public static final String COLFMTCS = "%-24s";**

**public static final String COLFMTCD = "%,10d";**

**public static final String COLFMTCF = "%,10.2f";**

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

**// Variables**

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

**// Declare variables**

**private static Scanner keyboard = new Scanner(System.in);**

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

**// readTextFile**

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

**public static void readTextFile(Product[] products)**

**{**

**// Declare variables**

**Scanner fileIn = null;**

**String line;**

**int lineCount;**

**int code;**

**String name;**

**double cost;**

**int count;**

**// Attempt to open input file**

**try**

**{**

**// Assign external file to file handle**

**fileIn = new Scanner(new FileInputStream(FILE\_NAME\_IN));**

**// Loop to read data**

**lineCount = 0;**

**while (fileIn.hasNextLine())**

**{**

**// Read line and test if header row**

**line = fileIn.nextLine();**

**// Parse input line and store tokens**

**code = Integer.parseInt(line.substring(COL1\_START, COL1\_END).trim());**

**name = line.substring(COL2\_START, COL2\_END).trim();**

**cost = Double.parseDouble(line.substring(COL3\_START, COL3\_END).trim());**

**count = Integer.parseInt(line.substring(COL4\_START, COL4\_END).trim());**

**// Create product object and store in array**

**products[lineCount] = new Product(code, name, cost, count);**

**// Increment line counter**

**lineCount = lineCount + 1;**

**}**

**// Show number of lines read**

**System.out.println(lineCount + " line(s) read from file '" +**

**FILE\_NAME\_IN + "'.");**

**// Close input file**

**fileIn.close();**

**}**

**// Handle file error**

**catch (FileNotFoundException e)**

**{**

**System.out.println("Error: file '" + FILE\_NAME\_IN +**

**"' not found.");**

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

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

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

**}**

**}**

**//---------------==-------------------------------------------------**

**// menuOption**

**//-----------==-----------------------------------------------------**

**private static int menuOption()**

**{**

**// Show menu and get option**

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

**System.out.println("1–Sell product");**

**System.out.println("2–Order product");**

**System.out.println("3–List product inventory");**

**System.out.println("4–Exit");**

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

**return keyboard.nextInt();**

**}**

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

**// codeIndex**

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

**private static int codeIndex(Product[] products, int code)**

**{**

**// Declare variables**

**int i = 0;**

**// Loop to search for code**

**while (i < products.length && code != products[i].getCode())**

**i = i + 1;**

**if (i == products.length)**

**return -1;**

**else**

**return i;**

**}**

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

**// writeTextFile**

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

**public static void writeTextFile(Product[] products)**

**{**

**// Declare variables**

**PrintWriter fileOut = null;**

**int lineCount;**

**// Attempt to open output file**

**try**

**{**

**// Assign external file to file handle**

**fileOut =**

**new PrintWriter(new FileOutputStream(FILE\_NAME\_OUT));**

**// Loop to write data**

**lineCount = 0;**

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

**{**

**// Write tokens**

**fileOut.printf(**

**COLFMT1 + COLFMT2 + COLFMT3 + COLFMT4 + "%n",**

**products[lineCount].getCode(),**

**products[lineCount].getName(),**

**products[lineCount].getCost(),**

**products[lineCount].getCount());**

**// Increment line counter**

**lineCount = lineCount + 1;**

**}**

**// Show number of lines written**

**System.out.println("\n" + lineCount + " data line(s) " +**

**"written to file '" + FILE\_NAME\_OUT + "'.");**

**// Close output file**

**fileOut.close();**

**}**

**// Handle file error**

**catch (FileNotFoundException e)**

**{**

**System.out.println("Error: file '" + FILE\_NAME\_OUT +**

**"' cannot be created or opened.");**

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

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

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

**}**

**}**

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

**// main**

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

**public static void main (String[] args)**

**{**

**// Declare variables**

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

**int option;**

**int code;**

**int codeIndex;**

**int quantity;**

**// Show application header**

**System.out.println("Welcome to Camping Critters");**

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

**// Load data from text file**

**readTextFile(products);**

**// Loop to process options**

**option = menuOption();**

**while (option != 4)**

**{**

**// Handle option**

**switch (option)**

**{**

**// Sell product**

**case 1:**

**// Loop to prompt for and get product code**

**System.out.println("\nSell product");**

**System.out.print("Enter the product code: ");**

**code = keyboard.nextInt();**

**codeIndex = codeIndex(products, code);**

**while (codeIndex == -1)**

**{**

**System.out.println("Error: product code '" +**

**code + "' is not found.");**

**System.out.print("Enter the product code: ");**

**code = keyboard.nextInt();**

**codeIndex = codeIndex(products, code);**

**}**

**// Loop to prompt for and get product quantity**

**System.out.print("Enter the product quantity (<=" +**

**products[codeIndex].getCount() + "): ");**

**quantity = keyboard.nextInt();**

**while (quantity > products[codeIndex].getCount())**

**{**

**System.out.println("Error: product quantity '" +**

**quantity +**

**"' is greater than the inventory amount.");**

**System.out.print("Enter the product quantity (<=" +**

**products[codeIndex].getCount() + "): ");**

**quantity = keyboard.nextInt();**

**}**

**// Update product quantity and inventory value and quantity**

**products[codeIndex].setCount(products[codeIndex].getCount() - quantity);**

**Product.setInventoryValue(Product.getInventoryValue() -**

**(quantity \* products[codeIndex].getCost()));**

**Product.setInventoryCount(Product.getInventoryCount() - quantity);**

**// Show product sale result**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Code:", products[codeIndex].getCode());**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Quantity:", quantity);**

**System.out.printf(COLFMTCS + COLFMTCF + "%n",**

**"Sale revenue ($):", (quantity \* products[codeIndex].getCost()));**

**break;**

**// Order product**

**case 2:**

**// Loop to prompt for and get product code**

**System.out.println("\nOrder product");**

**System.out.print("Enter the product code: ");**

**code = keyboard.nextInt();**

**codeIndex = codeIndex(products, code);**

**while (codeIndex == -1)**

**{**

**System.out.println("Error: product code '" +**

**code + "' not found.");**

**System.out.print("Enter the product code: ");**

**code = keyboard.nextInt();**

**codeIndex = codeIndex(products, code);**

**}**

**// Loop to prompt for and get product quantity**

**System.out.print("Enter the product quantity: ");**

**quantity = keyboard.nextInt();**

**while (quantity <= 0)**

**{**

**System.out.println("Error: product quantity '" +**

**quantity +**

**"' must greater than zero.");**

**System.out.print("Enter the product quantity: ");**

**quantity = keyboard.nextInt();**

**}**

**// Update product quantity and inventory value and quantity**

**products[codeIndex].setCount(products[codeIndex].getCount() + quantity);**

**Product.setInventoryValue(Product.getInventoryValue() +**

**(quantity \* products[codeIndex].getCost()));**

**Product.setInventoryCount(Product.getInventoryCount() + quantity);**

**// Show product order result**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Code:", products[codeIndex].getCode());**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Quantity:", quantity);**

**System.out.printf(COLFMTCS + COLFMTCF + "%n",**

**"Order cost ($):", (quantity \* products[codeIndex].getCost()));**

**break;**

**// List products**

**case 3:**

**// Loop to list products**

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

**System.out.printf(COLFMTS1 + COLFMTS2 + COLFMTS3 +**

**COLFMTS4 + "%n", "Code", "Product", "Cost ($)",**

**"Count");**

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

**System.out.printf(COLFMT1 + COLFMT2 +**

**COLFMT3 + COLFMT4 + "%n",**

**products[i].getCode(),**

**products[i].getName(),**

**products[i].getCost(),**

**products[i].getCount());**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Product count:", Product.getProductCount());**

**System.out.printf(COLFMTCS + COLFMTCF + "%n",**

**"Inventory value ($):", Product.getInventoryValue());**

**System.out.printf(COLFMTCS + COLFMTCD + "%n",**

**"Inventory count:", Product.getInventoryCount());**

**break;**

**// Handle invalid option**

**default:**

**System.out.println("Error: unknown menu option '" +**

**option + "'");**

**}**

**// Get next option**

**option = menuOption();**

**}**

**// Close keyboard**

**keyboard.close();**

**// Write data to text file**

**writeTextFile(products);**

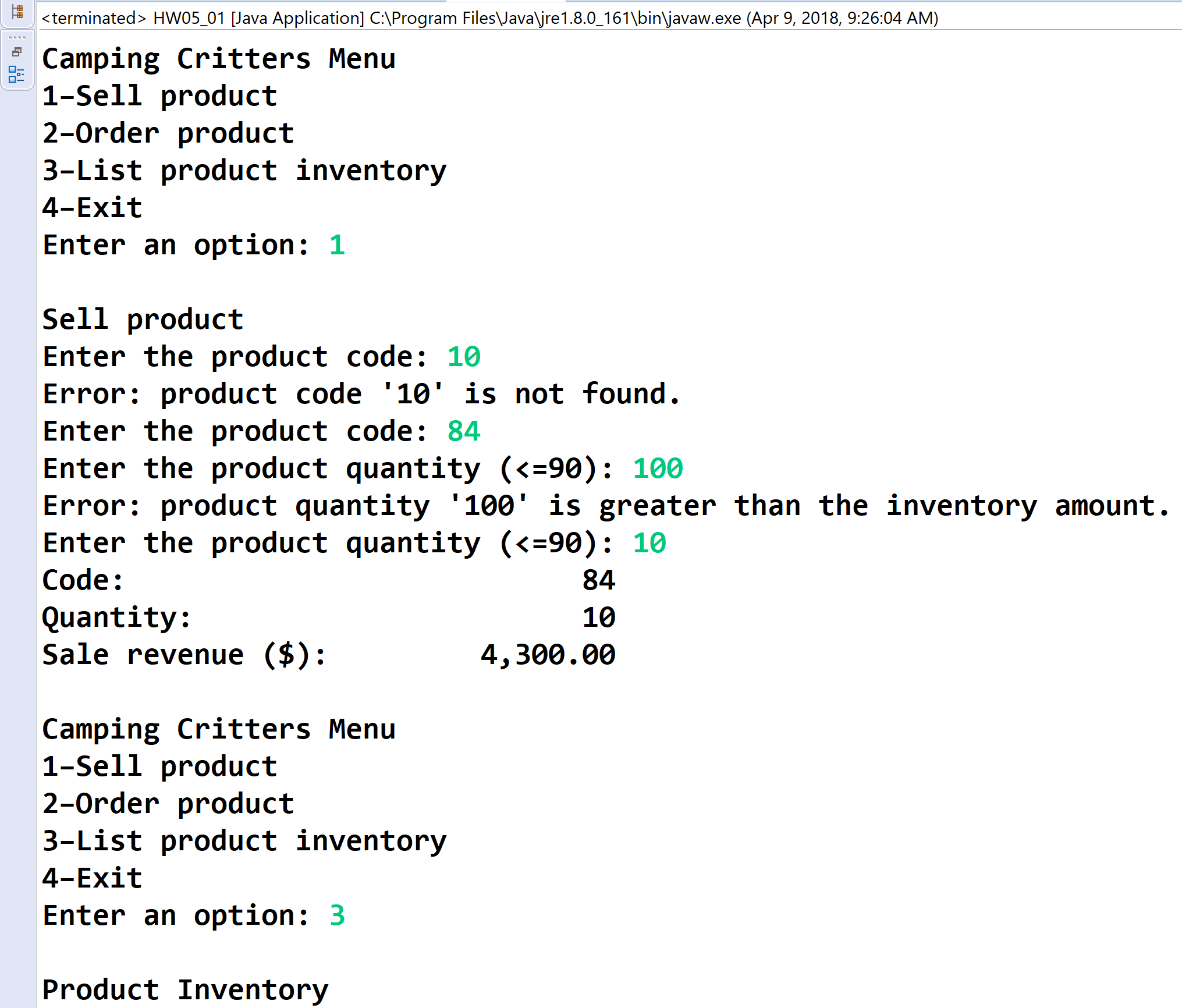
**// Show application close**

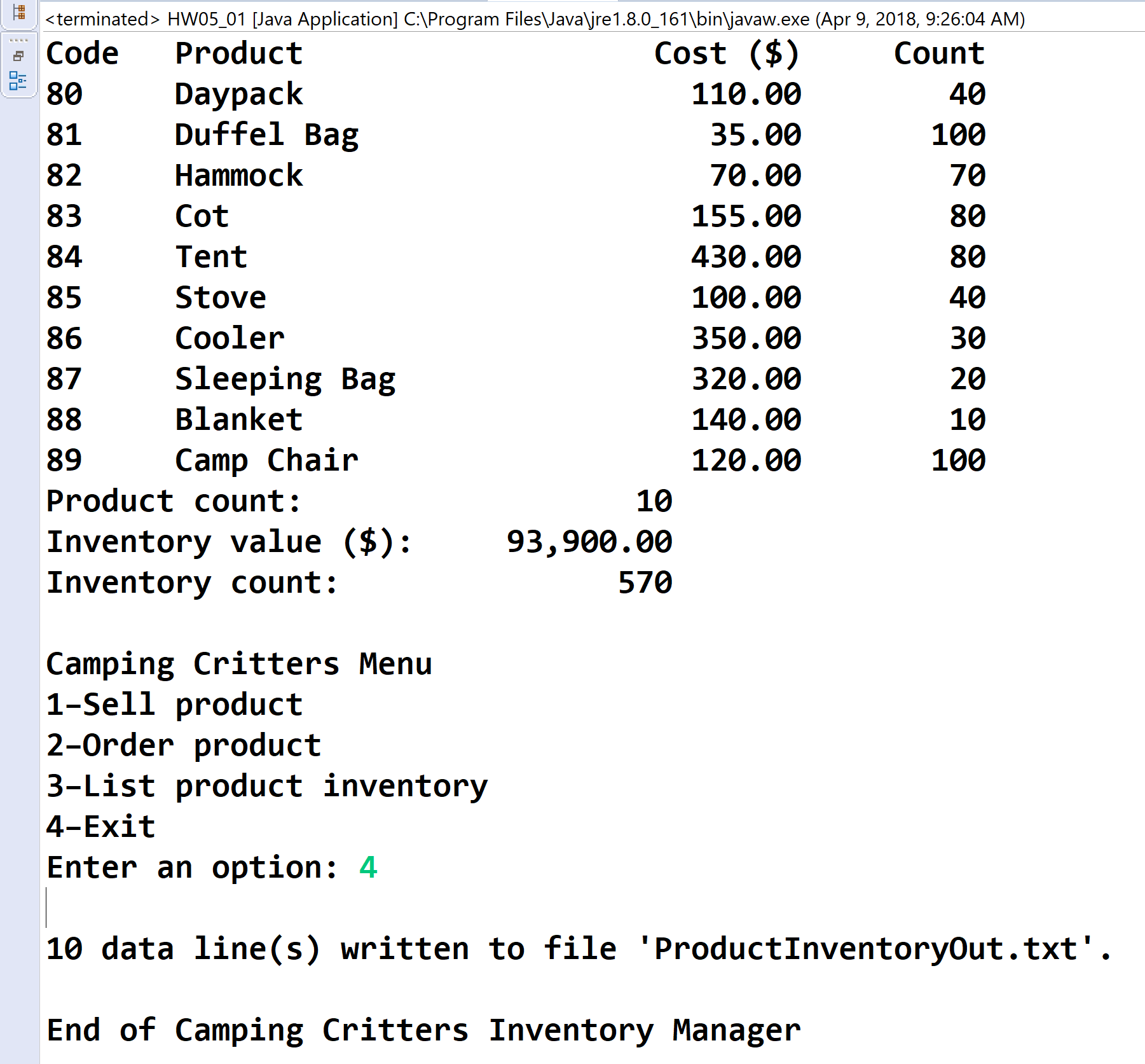
**System.out.println("\nEnd of Camping Critters Inventory Manager");**

**}**

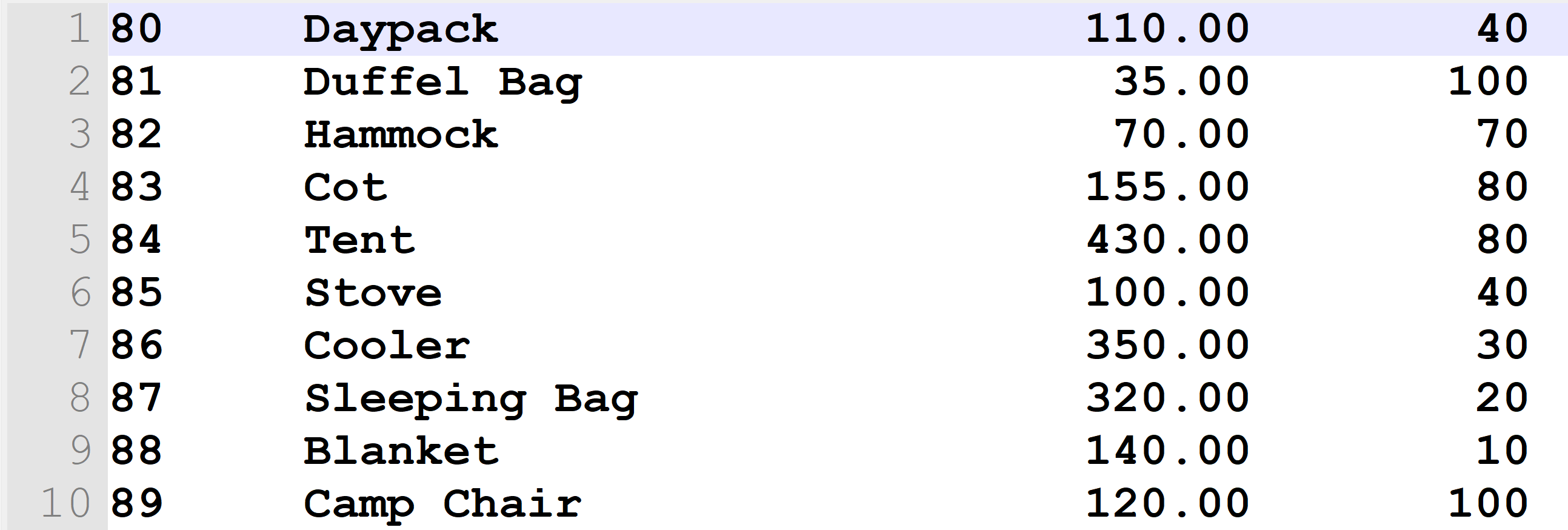
**}**

**Program output**





**ProductInventoryOut.txt**



\* **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**.