```java
package Assign_4;


import BasicIO.*;

/** This class creates a BasicForm driven application which reads a datafile of products and adds them
  * to a list. The user enters a target amount they would like to "purchase" and the program recursively searches
  * the list until a solution is found. It then adds the found items to a second list and displays that list
  * to the user.
  *
  * @author S. Fenwick
  *
  * @version 1.0 (March. 2017)
*/

public class Knapsack{

  public ASCIIDataFile file;
  private BasicForm form;
  private Node proList; //list of products
  private Node selList; //list of products that have been selected
  private int total; //the total of all product prices
  private int finalTotal; //if request is greater than this total, there is not enough product to buy
  private int request; //the number submitted by the user
  private int search; //the result of subtracting the request from the price of the product
  private int selListTotal; //the total value of the selected items


  /* This constructor creates a BasicForm and reads in an ASCIIDataFile. It recursively searches the list
   * looking for values that add up to what the user has requested.  */
  public Knapsack(){

    int button; //button pressed

    form = new BasicForm("Browse", "Buy", "Quit");
    proList = null;
    selList = null;

    loadProducts();
    setUpForm();


    for( ; ; ){
      button = form.accept();
      if(button == 2) break; //Quit
      switch(button) {
        case 0:{ //Browse
          browse();
          break;
        }
        case 1: { //Buy

          Node p;
          p = proList;

          request = form.readInt("target");

          if(request == 0){
            form.writeString("display", "Nothing here is free.");
            form.newLine("display");
            form.writeString("display", "-------");
            form.newLine("display");
            form.writeInt("target", finalTotal);
            break;
          }

          while( p != null ){

            selList = buy(p, request);
            if(selListTotal == request){
              break;
            }
            selListTotal = 0;
            selList = null;
            p = p.next;

          }

          if(selList == null){
            form.writeString("display", "No product selection to purchse.");
            form.newLine("display");
            form.writeString("display", "-------");
            form.newLine("display");
            form.writeInt("target", finalTotal);
            break;
          }

          form.writeString("display", "Products Selected: ");
          form.newLine("display");
          print();
          selListTotal = 0;
          selList = null;
          break;
        }

      }

    }

    form.close();

  } //constructor

  /* This method takes in the list from the constructor and searches through it looking for
   * values that add up to the requested amount by the user.
   *
   * @param aNode the node being looked at
   * @param find the value of (requeset - price of aNode) or what the program is searching for */

  public Node buy(Node aNode, int find){

    if( aNode == null ){ //Kick
      return null;
    }
    else if ((find - aNode.item.getPrice()) < 0){ //Deeper
      buy(aNode.next, search);
      return null;
    }
    else if (finalTotal < request){ //Kick
      form.writeString("display", "There is not enough product to match that request.");
      form.newLine("display");
      form.writeString("display", "--------");
      form.newLine("display");
      return null;
    }
    else if ((find - aNode.item.getPrice()) == 0){ //Kick
      selListTotal = selListTotal + aNode.item.getPrice();
      return new Node (aNode.item, null);
    }
    else{ //Deeper
      search = find - aNode.item.getPrice();
```

```java
      selListTotal = selListTotal + aNode.item.getPrice();
      return new Node (aNode.item , buy(aNode.next, search));
    }

  } //buy


  /* This method browses the 'store' (datafile) and writes the file to the text area on the
   * BasicForm. It calculates the total cost of all the products. */

  public void browse(){

    Node  p;

    p = proList;

    while ( p != null ) {

      form.writeString("display", p.item.getName());
      form.writeInt("display", p.item.getPrice());
      form.newLine("display");
      total = total + p.item.getPrice();
      p = p.next;

    };

    form.writeInt("target", total);
    form.writeString("display", "--------");
    form.newLine("display");
    finalTotal = total;
    total = 0; //resetting total so total does not continually add

  } //browse


  /* This method prints the list of selected items onto the text area of the BasicForm. */

  private void print(){

    Node p;

    p = selList;

    while ( p != null ){

      form.writeString("display", p.item.getName());
      form.writeInt("display", p.item.getPrice());
      form.newLine("display");
      p = p.next;
    }

    form.writeString("display", "--------");
    form.newLine("display");
    form.writeInt("target", finalTotal);

  } //print


  /* This method adds a canvas and text field to the BasicForm 'form'. */

  public void setUpForm(){

    form.addTextArea("display", "Status", 25, 75 );
    form.addTextField("target", "Target", 5, 10, 490);

  } //setUpForm

  /* This methods loads the products from the datafile. */

  public void loadProducts(){

    ASCIIDataFile file; //file of product info
    Product aProduct;

    file = new ASCIIDataFile();
    for( ; ; ){
      aProduct = new Product(file);
      if(file.isEOF() ) break;
      addList(aProduct);
    }

  } //loadProducts


  /* This method adds the products to a linked list.
   *
   * @param aProduct the product being added  */

  private void addList(Product aProduct){

    Node p;
    Node q;

    q = null;
    p = proList;
    while( p != null ) {
      q = p;
      p = p.next;
    };
    if( q ==null ){
      proList = new Node(aProduct, null);
    }
    else {
      q.next = new Node(aProduct, null);
    };
  } //addList


  public static void main (String args[]){ Knapsack k = new Knapsack();}

} //Knapsack
```