**// GetData.java**  
import javax.swing.JOptionPane;

public class GetData  
{  
   public static double getDouble(String s)  
   {  
       return Double.parseDouble(getWord(s));  
   }

   public static int getInt(String s)  
   {  
       return Integer.parseInt(getWord(s));  
   }

   public static String getWord(String s)  
   {  
       return JOptionPane.showInputDialog(s);  
   }

   public static String getString(String s)  
   {  
       return JOptionPane.showInputDialog(s);  
   }  
}

**// Address.java**  
public class Address  
{  
   // instance variables  
   private String street, city, state, zip;

   // parameterized constructor  
   public Address(String str, String city, String st, String zip)  
   {  
       street = str;  
       this.city = city;  
       state = st;  
       this.zip = zip;  
   }

   // to get the street name  
   public String getStreet()  
   {  
       return street;  
   }

   // to get the city name  
   public String getCity()  
   {  
       return city;  
   }

   // to get the state name  
   public String getState()  
   {  
       return state;  
   }

   // to get the zip code  
   public String getZip()  
   {  
       return zip;  
   }  
}

**// Manufacturer.java**  
public class Manufacturer  
{  
   // instance variables  
   private String companyName;  
   private Address companyAddress;

   // default constructor  
   public Manufacturer()  
   {  
       this.companyName = "";  
       this.companyAddress = null;  
   }

   // parameterized constructor  
   public Manufacturer(String compName, Address address)  
   {  
       this.companyName = compName;  
       this.companyAddress = address;  
   }

   // to get the name of the manufacturer company  
   public String getCompanyName()  
   {  
       return companyName;  
   }

   // to set the name of the manufacturer company  
   public void setCompanyName(String companyName)  
   {  
       this.companyName = companyName;  
   }

   // to get the Address object  
   public Address getCompanyAddress()  
   {  
       return companyAddress;  
   }

   // to set the Address object  
   public void setCompanyAddress(Address address)  
   {  
       this.companyAddress = address;  
   }  
}

**// Product.java**  
import java.text.SimpleDateFormat;  
import java.util.Date;

public class Product  
{  
   // instance variables  
   SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");  
    
   // to hold the Manufacturer object  
   Manufacturer manufacture;  
    
   // to hold product name  
   String productName;  
    
   // to hold quantity  
   int quantity;  
    
   // to hold the unit price  
   double unitPrice;  
    
   // to hold the product date  
   Date productCreated;  
    
   // default constructor  
   public Product()  
   {  
       this.productName = "";  
       this.quantity = 0;  
       this.unitPrice = 0.0;  
       this.productCreated = null;  
       this.manufacture = null;  
   }  
    
   // parameterized constructor  
   public Product(String prodName, int quantity, double unitPrice,  
                         Date productCreated, Manufacturer manufact)  
   {  
       this.productName = prodName;  
       this.quantity = quantity;  
       this.unitPrice = unitPrice;  
       this.productCreated = productCreated;  
       this.manufacture = manufact;  
   }  
    
   // to get the date of the product manufactured  
   public Date getProductCreated()  
   {  
       return productCreated;  
   }  
    
   // to set the date of the product manufactured  
   public void setProductCreated(Date productCreated)  
   {  
       this.productCreated = productCreated;  
   }  
    
   // to get the Manufacturer object  
   public Manufacturer getManufacture()  
   {  
       return manufacture;  
   }  
    
   // to set the Manufacturer object  
   public void setManufacture(Manufacturer manufacture)  
   {  
       this.manufacture = manufacture;  
   }  
    
   // to get the name of the product  
   public String getProductName()  
   {  
       return productName;  
   }  
    
   // to set the name of the product  
   public void setProductName(String prodName)  
   {  
       this.productName = prodName;  
   }  
    
   // to get the quantity of the product  
   public int getQuantity()  
   {  
       return quantity;  
   }  
    
   // to set the quantity of the product  
   public void setQuantity(int quantity)  
   {  
       this.quantity = quantity;  
   }  
    
   // to get the unit price of the product  
   public double getUnitPrice()  
   {  
       return unitPrice;  
   }  
    
   // to set the unit price of product  
   public void setUnitPrice(double unitPrice)  
   {  
       this.unitPrice = unitPrice;  
   }  
    
   // to update the quantity of the product  
   public void upDateQuantity(int quantity\_upDate)  
   {  
       quantity += quantity\_upDate;  
   }  
    
   // to update the price of the product  
   public void upDatePrice(double price\_upDate)  
   {  
       this.unitPrice = price\_upDate;  
   }  
    
   // to get the product information  
   public String getProductInfomation()  
   {  
       String result = "";  
       result += String.format("%-30s", productName);  
       String dateForm = sdf.format(productCreated);  
       result += String.format("\t %s", dateForm);  
       result += String.format("%10d", quantity);  
       result += String.format("\t%15.2f", unitPrice);  
       result += String.format("\t%15s",  
                           manufacture.getCompanyName());  
       result += String.format("\t%20s",  
                  manufacture.getCompanyAddress().getState());  
       return result;  
   }    
}

**// Database.java**  
import java.util.ArrayList;

public class Database  
{  
   // instance variables  
   private ArrayList<Product> list;

   private Product prod;  
   private int index;  
   private boolean found;

   // default constructor  
   public Database()  
   {  
       list = new ArrayList<Product>();  
       prod = null;  
       index = 0;  
       found = false;  
   }

   // search method  
   public void search(String key)  
   {  
       found = false;  
       int i = 0;

       while (!found && i < list.size())  
       {  
           Product b = list.get(i);  
           if (b.getProductName().equalsIgnoreCase(key))  
           {  
               prod = b;  
               found = true;  
               index = i;  
           }  
           else  
               i++;  
       }  
   }

   // add the product to the inventory list  
   public void add(Product newProduct)  
   {  
       list.add(newProduct);  
   }

   // remove the product from the inventory list  
   public Product delete(int i)  
   {  
       return list.remove(i);  
   }

   // to get index of the current product  
   // from the list  
   public int getIndex()  
   {  
       return index;  
   }

   // to get the boolean value which specifies  
   // whether the item is found in the list or not  
   public boolean inList()  
   {  
       return found;  
   }

   // to the current product  
   public Product getProduct()  
   {  
       return prod;  
   }

   // to get the size of the inventory list  
   public int size()  
   {  
       return list.size();  
   }

   // to intimate whether the list is empty or not  
   public boolean isEmpty()  
   {  
       return list.isEmpty();  
   }

   // to get the inventory list  
   public ArrayList<Product> getList()  
   {  
       return list;  
   }  
}

**// TestProduct.java**  
import java.text.\*;  
import javax.swing.\*;  
import java.util.\*;

public class TestProduct  
{  
   // to set or to retrieve the date  
   static SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

   // main() method  
   public static void main(String args[]) throws ParseException  
   {  
       // declare the required variables  
       String street, city, state, zip;  
       String companyName;  
       String productName;  
       int quantity;  
       String dateManu;  
       double unitPrice;  
       Product product = new Product();  
       Manufacturer manufacture = new Manufacturer();  
       Address address = null;  
       Database productDB = new Database();  
       Database deletedProdDB = new Database();  
       Date present = new Date();  
       boolean isDone = false;  
       int subMenu = 0;

       // loop until, the program is done  
       while (!isDone)  
       {  
           // prompt the user for the menu  
           int selectedMenu = GetData.getInt(  
                   "Welcome to Product Inventory\n\t1. Create Products\n"  
                   + "\t2. Update Product (Quantity/Price)\n\t3. Delete Product\n"  
                   + "\t4. To locate single product\n\t5. Inventory Report\n"  
                   + "\t6. List of deleted Products\n\t7. Quit");

           // use the conditional structured statement  
           // to switch to the selected menu  
           switch (selectedMenu)  
           {

           // Case to add products to the inventory list  
           case 1:

               // prompt the user and get the information regarding  
               // product, manufacturer and address  
               productName = GetData.getString("Enter product name: ");  
               quantity = GetData.getInt("Enter the quantity of product: ");  
               unitPrice = GetData  
                       .getDouble("Enter the unit price of product(in $): ");  
               companyName = GetData.getString(  
                       "Enter the name of the manufacturing product: ");  
               street = GetData.getString("Enter street address of company: ");  
               city = GetData.getString("Enter city address of company: ");  
               state = GetData.getString("Enter state address of company: ");  
               zip = GetData.getString("Enter ZIP code of company: ");

               dateManu = GetData.getString(  
                       "Enter date of manufacturing product in the form of(MM/dd/yyyy): ");  
               address = new Address(street, city, state, zip);  
               manufacture = new Manufacturer(companyName, address);

               present = sdf.parse(dateManu);

               // create the Product object  
               product = new Product(productName, quantity, unitPrice, present,  
                       manufacture);

               // add the product to the data base  
               productDB.add(product);  
               break;

           // case to update the product's price or quantity information  
           // as per the user choice  
           case 2:

               // prompt the user for the product name of which  
               // they want to update  
               productName = GetData  
                       .getString("Enter product name to update: ");

               // get the option chosen by the user  
               subMenu = GetData.getInt(  
                       "Select option to update: \n1. To update quantity\n2. To update price\n");

               // search for the product in data base  
               productDB.search(productName);

               // condition to check whether the product is not in  
               // the data base  
               if (!productDB.inList())  
               {  
                   // if the condition is true intimate the user  
                   // about the respective product is not found in the  
                   // database  
                   JOptionPane.showMessageDialog(null, "Product not found.");  
               }

               // otherwise  
               else  
               {  
                   // depending on the user wants to update the information  
                   // switch to the respective case  
                   switch (subMenu)  
                   {

                   // if user selects 1, then the user wants to update  
                   // about the quantity  
                   case 1:

                       // prompt the user whether user wants to add the  
                       // quantity of the  
                       // product or deduct the quantity of the product  
                       int menuOpt = GetData.getInt(  
                               "Select option to update: \n1. To add quantity\n2. To deduct quantity\n");  
                       int addRDeductQuant = 0;

                       // condition to check for the valid input  
                       if (menuOpt < 1 || menuOpt > 2)  
                       {  
                           JOptionPane.showMessageDialog(null,  
                                   "Please chose the correct choice");  
                       }  
                       else  
                       {  
                           // depending on the user choice switch to the  
                           // add or deduct quantity  
                           switch (menuOpt)  
                           {  
                           // case 1 to add quantity  
                           case 1:

                               // prompt for the amount of quantity  
                               addRDeductQuant = GetData.getInt(  
                                       "Enter the amount of quantity to add: ");

                               // get the product from the database  
                               product = productDB.getProduct();

                               // then update the quantity  
                               product.upDateQuantity(addRDeductQuant);  
                               break;

                           // case 2 to deduct quantity  
                           case 2:

                               // prompt for the amount of quantity  
                               addRDeductQuant = GetData.getInt(  
                                       "Enter the amount of quantity to deduct: ");

                               // get the product from the database  
                               product = productDB.getProduct();

                               // then update the quantity  
                               product.upDateQuantity(addRDeductQuant \* -1);  
                               break;  
                           }

                           JOptionPane.showMessageDialog(null,  
                                   "Quantity of the product is uupdated successfully.");  
                       }  
                       break;

                   // case to update the price  
                   case 2:

                       // prompt the user for the new price  
                       double priceToUpdate = GetData.getDouble(  
                               "Enter the unit price of product to update: ");  
                        
                       // validation of the price  
                       if (priceToUpdate < 0)  
                       {  
                           JOptionPane.showMessageDialog(null,  
                                   "Price of the product is not updated.");  
                       }  
                       else  
                       {  
                           // get the product from the database  
                           product = productDB.getProduct();

                           // update the price of the respective product  
                           product.upDatePrice(priceToUpdate);

                           // display the information to the user  
                           JOptionPane.showMessageDialog(null,  
                                   "Price of the product is updated successfully.");  
                       }  
                       break;  
                   default:  
                       JOptionPane.showMessageDialog(null,  
                               "Unable to process the option selected from the update list.");  
                   }  
               }  
               break;  
                
               // case to delete the product from the database if exists.  
           case 3:  
                
               // get the name of the product  
               productName = GetData  
                       .getString("Enter product name to update: ");  
                
               // search for the product  
               productDB.search(productName);  
                
               // check whether the product exists in the database  
               if (productDB.inList())  
               {  
                   // if exists, get the index  
                   int index = productDB.getIndex();  
                    
                   // add the product to the deleted product database  
                   deletedProdDB.add(productDB.getProduct());  
                    
                   // delete the product from the product database  
                   productDB.delete(index);  
                    
                   // display the information of the status of the  
                   // deletion  
                   JOptionPane.showMessageDialog(null, "The \"" + productName  
                           + "\" product is deleted successfully.");  
               }  
                
               // otherwise display the information of the status of the  
               // deletion  
               else  
               {  
                   JOptionPane.showMessageDialog(null, "Product not found.");  
               }  
               break;  
                
               // case to display the single product's information  
           case 4:  
                
               // prompt the user for the product name of which  
               // they want to update  
               productName = GetData  
                       .getString("Enter name of the product to display about: ");  
                
               // search for the product  
               productDB.search(productName);  
                
               // check whether the product exists in the database  
               if (productDB.inList())  
               {  
                   // display the information about the single product  
                   displaySingleProduct(productDB.getProduct(),  
                           JOptionPane.INFORMATION\_MESSAGE);  
               }  
                
               // otherwise, display the error message  
               else  
               {  
                   JOptionPane.showMessageDialog(null, "Product not found.");  
               }  
               break;  
                
               // case to display the inventory information  
           case 5:  
               // check whether the inventory in the data base  
               // is not null  
               if (productDB.getList() != null)  
               {  
                   // if not null, display the inventory information  
                   displayInventory(productDB,  
                           JOptionPane.INFORMATION\_MESSAGE);  
               }  
                
               // otherwise, display the error message  
               else  
               {  
                   JOptionPane.showMessageDialog(null,  
                           "There are no products present in the inventory.");  
               }  
               break;  
                
               // case to display the deleted inventory information  
           case 6:  
               // check whether the deleted inventory data base  
               // is not null  
               if (deletedProdDB.getList() != null)  
               {  
                   displayInventory(deletedProdDB,  
                           JOptionPane.INFORMATION\_MESSAGE);  
               }  
                
               // otherwise, display the error message  
               else  
               {  
                   JOptionPane.showMessageDialog(null,  
                           "There are no products present in the inventory.");  
               }  
               break;  
                
               // if the user chose to quit set the isDone to true  
           case 7:  
               isDone = true;  
               break;  
                
               // if user chooses other than 1 - 7 options, display  
               // error message  
           default:  
               JOptionPane.showMessageDialog(null,  
                       "Unable to process the option selected from the menu list.");  
           }  
       }  
   }

   // public method to get the product information in a format  
   public static String getFormatedProductInfo(Product info)  
   {  
       String result = String.format("%30s", info.getProductName());  
       result += String.format("%30s", sdf.format(info.getProductCreated()));  
       result += String.format("%30s", info.getManufacture().getCompanyName());  
       return result;  
   }

   // public method to display the deleted inventory list  
   public static void displayDeletedInventory(Database productDB,  
           int Type\_Message)  
   {  
       String inventResult = "";  
       ArrayList<Product> prodList = productDB.getList();  
       inventResult += String.format("%30s %30s %30s", "Product",  
               "Purchase Date", "Manufacturer");  
       for (int i = 0; i < productDB.size(); i++)  
       {  
           inventResult += getFormatedProductInfo(prodList.get(i)) + "\n";  
       }  
       JTextArea text = new JTextArea(inventResult, 10, 50);

       JScrollPane pane = new JScrollPane(text);

       JOptionPane.showMessageDialog(null, pane, "Deleted Inventory Details",  
               Type\_Message);  
   }

   // public method to display the inventory list  
   public static void displayInventory(Database productDB, int Type\_Message)  
   {  
       String inventResult = "";  
       ArrayList<Product> prodList = productDB.getList();  
       inventResult += String.format("%-30s \t%s %10s %15s %20s %15s\n",  
               "Product", "Purchase Date", "Quantity", "Price($)",  
               "Manufacturer", "State");  
       for (int i = 0; i < productDB.size(); i++)  
       {  
           inventResult += prodList.get(i).getProductInfomation() + "\n";  
       }  
       JTextArea text = new JTextArea(inventResult, 10, 60);

       JScrollPane pane = new JScrollPane(text);

       JOptionPane.showMessageDialog(null, pane, "Inventory Details",  
               Type\_Message);  
   }

   // public method to display the single product  
   public static void displaySingleProduct(Product product, int Type\_Message)  
   {  
       String productInfo = "Product Name: " + product.getProductName() + "\n";  
       productInfo += String.format("Product's Unit Price: $%.2f",  
               product.getUnitPrice()) + "\n";  
       productInfo += "Quantity of product: " + product.getQuantity() + "\n";  
       JTextArea text = new JTextArea(productInfo, 10, 30);

       JScrollPane pane = new JScrollPane(text);

       JOptionPane.showMessageDialog(null, pane,  
               product.getProductName() + " Details", Type\_Message);  
   }  
}