```java
package com.CMP_4008Y;

import java.util.ArrayList;
import java.util.Collections;

public class Inventory {

    // Empty constructor.
    Inventory() {}

    // Create new private instance of ArrayList called 'list' (global).
    ArrayList<StockItem> list = new ArrayList<>();

    // Add item (from StockProgram class) to 'list'.
    private void add(StockItem item) { list.add(item); }

    // How many instances of 'StockItem' are in inventory.
    int size() { return list.size(); }

    // Retrieve the nth 'StockItem' instance.
    StockItem get(int index) { return list.get(index); }

    /*
    Checks every field named 'price' in stockItem, if it's > than 'bound',
    then add to new inventory object (called 'price').
    */
    Inventory queryPrice(int bound, Inventory original) {
        System.out.println("Components above " + bound + "p:");

        Inventory price = new Inventory();
        for (int i = 0; i < original.size(); i++) {
            double priceField = original.list.get(i).price;
            if (priceField > bound) price.add(original.get(i));
        }
        System.out.println("Total items above " + bound + "p: " + price.size());
        System.out.println(String.format(
                "%-15s %-15s %-15s %-15s %-15s", "Name", "Code",
                "Quantity", "Price", "Description"
        ));
        return price;
    }

    /*
    finds largest int in 'quantity' field of all StockItems.
    Output StockItems (to string) that contain largest quantity value.
    */
    StringBuilder biggestQuantity(Inventory original) {
        int biggest = 0;
        StringBuilder str = new StringBuilder();

        System.out.println("The component we have most in stock is:");
        System.out.println(String.format(
                "%-15s %-15s %-15s %-15s %-15s", "Name", "Code",
                "Quantity", "Price", "Description"
        ));

        for (int i = 0; i < original.size(); i++ )
            if(original.get(i).quantity >= biggest)
                biggest = original.get(i).quantity;

        for (int i = 0; i < original.size(); i++)
            if (original.list.get(i).quantity == biggest)
                str.append(original.list.get(i).toString());
        return str;
    }

    // Count number of NPN transistors and display them.
    String npnTransistors(Inventory original) {
        System.out.println("NPN transistors:");
        int totalNPN = 0;

        System.out.println(String.format(
                "%-15s %-15s %-15s %-15s %-15s", "Name", "Code",
                "Quantity", "Price", "Description"
        ));

        for (int i = 0; i < original.size(); i++ )

            if (original.list.get(i).name.equals("transistor") &&
                    original.list.get(i).description.equals("NPN")) {

                totalNPN += original.list.get(i).quantity;
                System.out.println(original.list.get(i));
            }

        return "Total NPN resistors available: " + totalNPN + "\n\n";
    }

    // Sorts list in reverse (ascending) order.
    void sortByPrice() {
        System.out.println("inventory filtered by price (Ascending):");
        System.out.println(String.format(
                "%-15s %-15s %-15s %-15s %-15s", "Name", "Code",
                "Quantity", "Price", "Description"
        ));

        this.list.sort(Collections.reverseOrder());
    }

    /*
    Method checks if each object in list has a name 'resistor'.
    If it does, add description value to totalResistance value.
    Output totalResistance when loop is complete.
    */
    String resistanceSum(Inventory original) {
        double totalResistance = 0;
        for (int i = 0; i < original.size(); i++)
            if (original.list.get(i).name.equals("resistor"))
                totalResistance += Double.parseDouble(
                        original.list.get(i).description) *
                        original.list.get(i).quantity;
        return String.format("%f", totalResistance);
    }

    // instead of returning object type and storage location,
    // converts each object in the list into String and outputs the result.
    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        for (StockItem stockItem : list)
            str.append(stockItem.toString()).append("\n");
        return str.toString();
    }
}
```

```java
package com.CMP_4008Y;

public class StockItem implements Comparable<StockItem> {

    String name;
    private String code;
    int quantity;
    double price;
    String description;

    // Make public constructor (accessible even in other packages).
    public StockItem(String[] line) {
        name = line[0];
        code = line[1];
        quantity = Integer.parseInt(line[2]);
        if (Double.parseDouble(line[3]) > 0 ) {
            price = Double.parseDouble(line[3]);
        }
        else throw new IllegalArgumentException
                ("price for items cannot be below or equal to zero!");

        // Check if length of array is greater than 4, if so,
        // generate the fifth field to store defined 'description'.
        if (line.length > 4) description = line[4];
    }

    @Override // Over ride toString() in object class and define manually (below).
    public String toString() {
        if (description == null) description = "";
        return String.format(
                "%-15s %-15s %-15s %-15s %-15s",
                name, code, quantity, "£" + price/100, description
        );
    }

    // method compares instance of price with field price in 'stockItem'
    @Override
    public int compareTo(StockItem stockItem) {
        return Double.compare(stockItem.price, this.price);
    }
}
```

```java
/*

By: Callum Clegg [100279967]
Last updated: 17/01/2020

 */

package com.CMP_4008Y;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class StockProgram {

    public static void main(String[] args) throws FileNotFoundException {
        // Calling 'stockItems()' method to be compiled.
        stockItems();
    }

    private static void stockItems() throws FileNotFoundException {

        // Generate a new instance of class 'inventory'.
        Inventory inventory = new Inventory();

        // Assign 'inventory.txt' to File dataType.
        File inventoryContent = new File("inventory.txt");

        // Create new scanner object to receive input from .txt file above.
        Scanner scan = new Scanner(inventoryContent);

        /*
        While scan object contains a line,
        declare line as String dataType
        and then split line into String[] with regex ', '.
        Create a new instance of class StockItem (called 'item')
        and use data stored in 'fields' to produce instance.
        Add 'item' (object) to the generated inventory class instance.
        Repeat process till all 'items' are stored in a list.
        */
        while (scan.hasNextLine()) {
            String line = scan.nextLine();
            String[] fields = line.split(", ");
            StockItem item  = new StockItem(fields);
            inventory.list.add(item);
        }

        // Print out all StockItems in inventory incrementally.
        System.out.println("Inventory:");
        System.out.println(String.format(
                "%-15s %-15s %-15s %-15s %-15s", "Name", "Code",
                "Quantity", "Price", "Description"
        ));

        for (int i = 0; i < inventory.size(); i++)
            System.out.println(inventory.get(i));
        System.out.println("\n");

        // Calling methods in 'inventory.java' to answer queries.
        System.out.println(inventory.resistanceSum(inventory) + " \u2126 (Ohms) \n\n");
        System.out.println(inventory.queryPrice(10, inventory) + "\n");
        System.out.println(inventory.npnTransistors(inventory));
        inventory.sortByPrice();
        System.out.println(inventory + "\n");
        System.out.println(inventory.biggestQuantity(inventory));
    }
}
```