**Part -1:**

**Java Program:**

public class InventoryManagementSystem {

    private static final int MAX\_ITEMS = 100; // Maximum number of items allowed in inventory

    private Item[] items; // Array to store inventory items

    private int numItems; // Current number of items in inventory

    public InventoryManagementSystem() {

        items = new Item[MAX\_ITEMS];

        numItems = 0;

    }

    public void addItem(String name, double price, int quantity) {

        if (numItems >= MAX\_ITEMS) {

            System.out.println("Inventory full! Cannot add more items.");

            return;

        }

        items[numItems] = new Item(name, price, quantity);

        numItems++;

        System.out.println("Item added successfully!");

    }

    public void removeItem(String name, int quantity) {

        int foundIndex = findItemByName(name);

        if (foundIndex == -1) {

            System.out.println("Item not found in inventory.");

            return;

        }

        if (items[foundIndex].getQuantity() < quantity) {

            System.out.println("Insufficient quantity available for " + name + ".");

            return;

        }

        items[foundIndex].removeQuantity(quantity);

        numItems--;

        System.out.println(quantity + " units of " + name + " removed from inventory.");

    }

    public void updateItemPrice(String name, double newPrice) {

        int foundIndex = findItemByName(name);

        if (foundIndex == -1) {

            System.out.println("Item not found in inventory.");

            return;

        }

        items[foundIndex].setPrice(newPrice);

        System.out.println("Price of " + name + " updated successfully.");

    }

    public void searchItemByName(String name) {

        int foundIndex = findItemByName(name);

        if (foundIndex == -1) {

            System.out.println("Item not found in inventory.");

            return;

        }

        System.out.println(items[foundIndex].toString());

    }

    public void displayInventory() {

        if (numItems == 0) {

            System.out.println("Inventory is empty.");

            return;

        }

        System.out.println("Inventory List:");

        for (int i = 0; i < numItems; i++) {

            System.out.println(items[i].toString());

        }

    }

    private int findItemByName(String name) {

        for (int i = 0; i < numItems; i++) {

            if (items[i].getName().equals(name)) {

                return i;

            }

        }

        return -1;

    }

    public static void main(String[] args) {

        InventoryManagementSystem inventory = new InventoryManagementSystem();

        inventory.addItem("Shirt", 19.99, 10);

        inventory.addItem("Pants", 29.95, 5);

        inventory.addItem("Hat", 14.50, 20);

        inventory.searchItemByName("Shirt");

        inventory.updateItemPrice("Pants", 32.00);

        inventory.removeItem("Hat", 15);

        inventory.displayInventory();

    }

}

class Item {

    private String name;

    private double price;

    private int quantity;

    public Item(String name, double price, int quantity) {

        this.name = name;

        this.price = price;

        this.quantity = quantity;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

    public int getQuantity() {

        return quantity;

    }

    public void setPrice(double newPrice) {

        this.price = newPrice;

    }

    public void removeQuantity(int amount) {

        this.quantity -= amount;

    }

    @Override

    public String toString() {

        return "Name: " + name + ", Price: $" + price + ", Quantity: " + quantity;

    }

}

**RSM Tool Report:**

***Report Banner - Edit rsm.cfg File***

**Resource Standard Metrics™ for C, C++, C# and Java**

Version 7.75 - [mSquaredTechnologies.com](http://mSquaredTechnologies.com)

**License Type**: **Shareware Evaluation License**

**Licensed To** : **Shareware End User - Distribute Freely**

**License No**. : **SW1380 License Date**: Dec 05, 1998

**Build Date** : **Sep 2 2009 Run Date**: Apr 22, 2024

**©1996-2009 M Squared Technologies LLC™**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**License File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.lic**

**Config. File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.cfg**

**Command Line: -H -OC:\Users\prash\M2 RSM Wizard\output\output.htm -c -FC**

**:\Users\prash\M2 RSM Wizard\input\rsm\_file\_list.lst**

**~~ Function Metrics ~~**

**~~ Complexity Analysis ~~**

**File: [D:\ASU\Classes\SPPQM CSE566\Assignments 566\A5\A5 Java files\Inven](file:///D:\\ASU\\Classes\\SPPQM%20CSE566\\Assignments%20566\\A5\\A5%20Java%20files\\InventoryManagementSystem.java" \t "_blank)**

**[toryManagementSystem.java](file:///D:\\ASU\\Classes\\SPPQM%20CSE566\\Assignments%20566\\A5\\A5%20Java%20files\\InventoryManagementSystem.java" \t "_blank)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.InventoryManagementSystem**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.addItem**

**Parameters: (String name, double price, int quantity)**

**Complexity Param 3 Return 1 Cyclo Vg 2 Total 6**

**LOC 9 eLOC 7 lLOC 5 Comment 0 Lines 10**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.removeItem**

**Parameters: (String name, int quantity)**

**Complexity Param 2 Return 2 Cyclo Vg 3 Total 7**

**LOC 14 eLOC 11 lLOC 8 Comment 0 Lines 16**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.updateItemPrice**

**Parameters: (String name, double newPrice)**

**Complexity Param 2 Return 1 Cyclo Vg 2 Total 5**

**LOC 9 eLOC 7 lLOC 5 Comment 0 Lines 10**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.searchItemByName**

**Parameters: (String name)**

**Complexity Param 1 Return 1 Cyclo Vg 2 Total 4**

**LOC 8 eLOC 6 lLOC 4 Comment 0 Lines 9**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.displayInventory**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 3 Total 4**

**LOC 10 eLOC 7 lLOC 5 Comment 0 Lines 11**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.findItemByName**

**Parameters: (String name)**

**Complexity Param 1 Return 2 Cyclo Vg 3 Total 6**

**LOC 8 eLOC 5 lLOC 3 Comment 0 Lines 8**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.main**

**Parameters: (String[] args)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 10 eLOC 9 lLOC 8 Comment 0 Lines 13**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.Item**

**Parameters: (String name, double price, int quantity)**

**Complexity Param 3 Return 1 Cyclo Vg 1 Total 5**

**LOC 5 eLOC 4 lLOC 3 Comment 0 Lines 5**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getName**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getPrice**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getQuantity**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.setPrice**

**Parameters: (double newPrice)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.removeQuantity**

**Parameters: (int amount)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.toString**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

**------------------------------------------------------------------------**

**~~ Total File Summary ~~**

**LOC 106 eLOC 80 lLOC 55 Comment 3 Lines 131**

**------------------------------------------------------------------------**

**~~ File Functional Summary ~~**

**File Function Count....: 15**

**Total Function LOC.....: 95 Total Function Pts LOC : 2.0**

**Total Function eLOC....: 71 Total Function Pts eLOC: 1.5**

**Total Function lLOC....: 49 Total Function Pts lLOC: 1.0**

**Total Function Params .: 15 Total Function Return .: 17**

**Total Cyclo Complexity : 24 Total Function Complex.: 56**

**------ ----- ----- ------ ------ -----**

**Max Function LOC ......: 14 Average Function LOC ..: 6.33**

**Max Function eLOC .....: 11 Average Function eLOC .: 4.73**

**Max Function lLOC .....: 8 Average Function lLOC .: 3.27**

**------ ----- ----- ------ ------ -----**

**Max Function Parameters: 3 Avg Function Parameters: 1.00**

**Max Function Returns ..: 2 Avg Function Returns ..: 1.13**

**Max Interface Complex. : 5 Avg Interface Complex. : 2.13**

**Max Cyclomatic Complex.: 3 Avg Cyclomatic Complex.: 1.60**

**Max Total Complexity ..: 7 Avg Total Complexity ..: 3.73**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File: [D:\ASU\Classes\SPPQM CSE566\Assignments 566\A5\A5 Java file](file:///D:\\ASU\\Classes\\SPPQM%20CSE566\\Assignments%20566\\A5\\A5%20Java%20files\\InventoryManagementSystem.java" \t "_blank)**

**[s\InventoryManagementSystem.java](file:///D:\\ASU\\Classes\\SPPQM%20CSE566\\Assignments%20566\\A5\\A5%20Java%20files\\InventoryManagementSystem.java" \t "_blank)**

**------------------------------------------------------------------------**

**~~ Total Metrics For 1 Files ~~**

**------------------------------------------------------------------------**

**~~ Total Project Summary ~~**

**LOC 106 eLOC 80 lLOC 55 Comment 3 Lines 131**

**Average per File, metric/1 files**

**LOC 106 eLOC 80 lLOC 55 Comment 3 Lines 131**

**------------------------------------------------------------------------**

**~~ Project Functional Metrics ~~**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.InventoryManagementSystem**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.addItem**

**Parameters: (String name, double price, int quantity)**

**Complexity Param 3 Return 1 Cyclo Vg 2 Total 6**

**LOC 9 eLOC 7 lLOC 5 Comment 0 Lines 10**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.removeItem**

**Parameters: (String name, int quantity)**

**Complexity Param 2 Return 2 Cyclo Vg 3 Total 7**

**LOC 14 eLOC 11 lLOC 8 Comment 0 Lines 16**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.updateItemPrice**

**Parameters: (String name, double newPrice)**

**Complexity Param 2 Return 1 Cyclo Vg 2 Total 5**

**LOC 9 eLOC 7 lLOC 5 Comment 0 Lines 10**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.searchItemByName**

**Parameters: (String name)**

**Complexity Param 1 Return 1 Cyclo Vg 2 Total 4**

**LOC 8 eLOC 6 lLOC 4 Comment 0 Lines 9**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.displayInventory**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 3 Total 4**

**LOC 10 eLOC 7 lLOC 5 Comment 0 Lines 11**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.findItemByName**

**Parameters: (String name)**

**Complexity Param 1 Return 2 Cyclo Vg 3 Total 6**

**LOC 8 eLOC 5 lLOC 3 Comment 0 Lines 8**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: InventoryManagementSystem.main**

**Parameters: (String[] args)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 10 eLOC 9 lLOC 8 Comment 0 Lines 13**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.Item**

**Parameters: (String name, double price, int quantity)**

**Complexity Param 3 Return 1 Cyclo Vg 1 Total 5**

**LOC 5 eLOC 4 lLOC 3 Comment 0 Lines 5**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getName**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getPrice**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.getQuantity**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.setPrice**

**Parameters: (double newPrice)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.removeQuantity**

**Parameters: (int amount)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

[**Function**](file:///D:\ASU\Classes\SPPQM%20CSE566\Assignments%20566\A5\A5%20Java%20files\InventoryManagementSystem.java)**: Item.toString**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 3 eLOC 2 lLOC 1 Comment 0 Lines 3**

**Total: Functions**

**LOC 95 eLOC 71 lLOC 49 InCmp 32 CycloCmp 24**

**Function Points FP(LOC) 1.8 FP(eLOC) 1.3 FP(lLOC) 0.9**

**------------------------------------------------------------------------**

**~~ Project Functional Analysis ~~**

**Total Functions .......: 15 Total Physical Lines ..: 104**

**Total LOC .............: 95 Total Function Pts LOC : 1.8**

**Total eLOC ............: 71 Total Function Pts eLOC: 1.3**

**Total lLOC.............: 49 Total Function Pts lLOC: 0.9**

**Total Cyclomatic Comp. : 24 Total Interface Comp. .: 32**

**Total Parameters ......: 15 Total Return Points ...: 17**

**Total Comment Lines ...: 0 Total Blank Lines .....: 9**

**------ ----- ----- ------ ------ -----**

**Avg Physical Lines ....: 6.93**

**Avg LOC ...............: 6.33 Avg eLOC ..............: 4.73**

**Avg lLOC ..............: 3.27 Avg Cyclomatic Comp. ..: 1.60**

**Avg Interface Comp. ...: 2.13 Avg Parameters ........: 1.00**

**Avg Return Points .....: 1.13 Avg Comment Lines .....: 0.00**

**------ ----- ----- ------ ------ -----**

**Max LOC ...............: 14**

**Max eLOC ..............: 11 Max lLOC ..............: 8**

**Max Cyclomatic Comp. ..: 3 Max Interface Comp. ...: 4**

**Max Parameters ........: 3 Max Return Points .....: 2**

**Max Comment Lines .....: 0 Max Total Lines .......: 16**

**------ ----- ----- ------ ------ -----**

**Min LOC ...............: 3**

**Min eLOC ..............: 2 Min lLOC ..............: 1**

**Min Cyclomatic Comp. ..: 1 Min Interface Comp. ...: 1**

**Min Parameters ........: 0 Min Return Points .....: 1**

**Min Comment Lines .....: 0 Min Total Lines .......: 3**

**------------------------------------------------------------------------**

**~~ File Summary ~~**

**C Source Files *\*.c* ....: 0 C/C++ Include Files *\*.h*: 0**

**C++ Source Files *\*.c\** .: 0 C++ Include Files *\*.h\** : 0**

**C# Source Files *\*.cs* ..: 0 Java Source File *\*.jav\**: 1**

**Other Source Files ....: 0**

**Total File Count ......: 1**

**Shareware evaluation licenses process only 20 files.**

**Paid licenses enable processing for an unlimited number of files.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Report Banner - Edit rsm.cfg File***

***Apart from the default metrics such as LOC (Lines of Code), eLOC (Executable Lines of Code), lLOC (Logical Lines of Code), Comment, Lines. Metrics for cyclomatic, interface and total complexity have been considered for additional metrics.***

**Analysis of InventoryManagementSystem.java:**

* Overall LOC: 106
* Average LOC per Function: 6.33
* Cyclomatic Complexity: Ranges from 1 to 3 (mostly low)
* Comment Lines: 0 (throughout the code)

**Interpretation:**

* The RSM metrics suggest that the code has relatively low inherent complexity. Most functions have a single decision point and a manageable size.
* However, the complete absence of comments is a significant concern. It reduces code readability and makes understanding the logic and purpose of functions challenging. This can hinder maintainability and future modifications.

**Recommendations:**

* Adding comments to explain the functionality of code sections, especially within the larger functions like addItem and removeItem, would significantly improve code clarity.
* Comments can describe the purpose of variables, logic behind conditional statements, and overall function intent.

Incorporating comments would enhance the overall quality and maintainability of the InventoryManagementSystem.java program.

**Part - 2: Code Portability Measures**

**Literature Review: Lines of Code (LOC) as a Portability Indicator**

While not a dedicated metric for portability, Lines of Code (LOC) can offer indirect insights. Generally, code with lower LOC tends to be more portable. This is because it suggests a focus on core functionality without unnecessary platform-specific elements or complex logic.

**Calculation:**

LOC is a simple count of the total number of lines in the source code. However, blank lines and comments are typically excluded.

**Relation to RSM Tool Metrics:**

RSM metrics like LOC and eLOC (Executable Lines of Code) can be correlated with portability. Lower LOC and a higher ratio of eLOC to LOC suggest a focus on essential functionality, potentially indicating better portability. However, RSM doesn't directly assess dependencies on specific libraries or APIs, which can significantly impact portability.

**Additional Considerations:**

* Code with higher code reuse through functions or well-designed classes might be more portable as it promotes modularity and reduces platform-specific dependencies.
* Portability analysis should ideally consider the target platforms and identify potential dependencies or limitations within the code.

**Conclusion:**

LOC can be a rudimentary indicator of code portability when considered alongside other factors. For a comprehensive evaluation, it's beneficial to combine these measures with manual code review to identify platform-specific dependencies and opportunities for improvement.

**References:**

1. <http://msquaredtechnologies.com/RSM-Help.html>