SHANMUGHA

ARTS, SCIENCE, TECHNOLOGY AND RESEARCH ACADEMY

(SASTRA) University

Tirumalaisamudram, Thanjavur-613 401

School of Computing

B.Tech Computer Science and Engineering Program

# CSE218-Software Engineering Lab

Manual Revised by

Dr. R.Kavitha, AP-III, CSE, SoC

On 04.01.2024

**SHANMUGHA**
**ARTS, SCIENCE, TECHNOLOGY AND RESEARCH ACADEMY**
**(SASTRA) University**
Tirumalaisamudram, Thanjavur-613 401 School of Computing
B.Tech Computer Science Program

# CSE218 Software Engineering Lab

## Detailed List of Experiments for Software Engineering Lab as per the syllabus

1. CASE TOOLS STUDY – Study of Rational unified Process.

2. Preparing an SRS using Rational Requisite Pro through Rational Administrator.

3. Designing the Use Case Diagram and Activity Diagram using Rational Rose.

4. Drawing a) Sequence Diagram
   b) Collaboration Diagram
   c) Class Diagram using Rational Rose

5. Demonstrate Forward and Reverse Engineering using Java.

6. Execute Memory leak testing with Rational Purify.

7. Perform Code Coverage testing with Rational Pure Coverage.

8. Managing Test plan, suites using Rational Test Manager for Manual Testing.

9. Develop a GUI Test Script using Rational Robot for Automated Testing.

10. Develop a Test Script for Object Insertions and Alphanumeric verification points.

11. a Develop a Test Script with an External Data Source using Rational Robot.
11. b Develop a Test Script with Data Pools using Rational Robot and Rational Test Manager.

12. Create a Test script for Website Testing using Rational Robot.

**Exercise No.1**

## Computer Aided Software Engineering (CASE) Tools – Overview (Study Experiment)

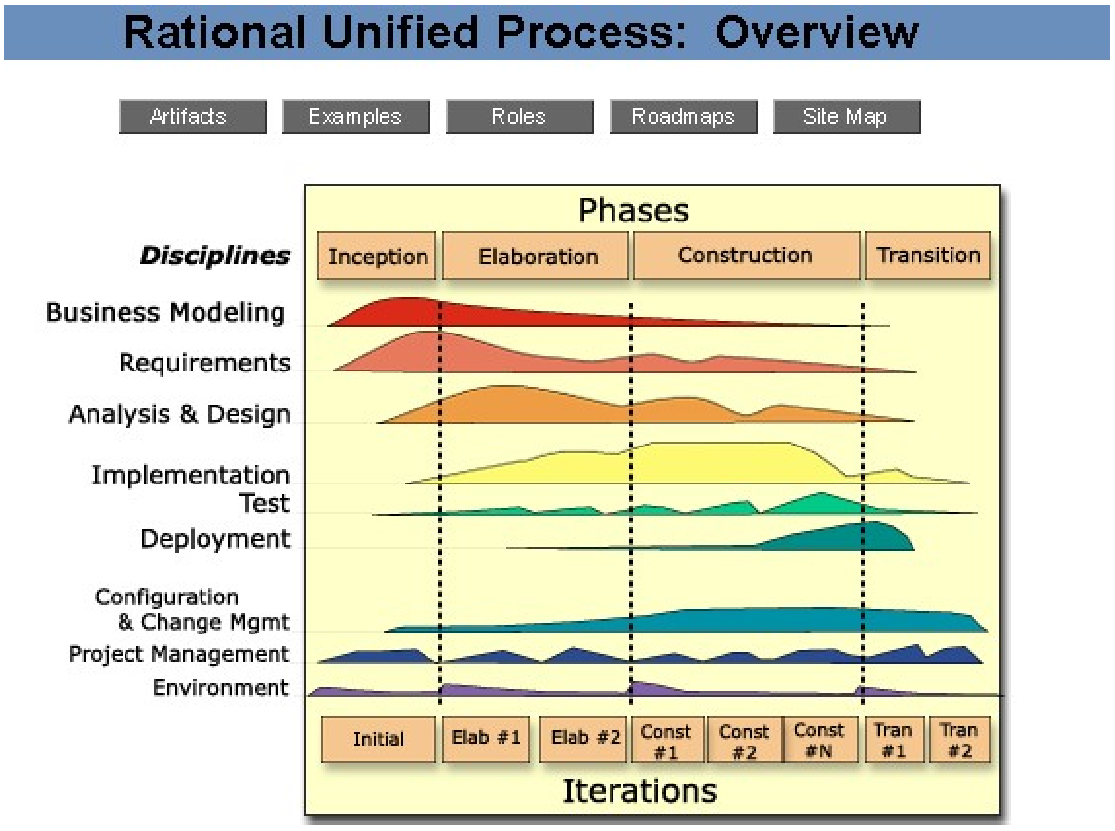
**OBJECTIVE:** The students will study about various tools in IBM – Rational Software Architect in RUP. The list of Software which the students would learn is:

1. Rational Administrator
2. Rational Requisite Pro
3. Rational Rose
4. Rational Purify
5. Rational Test Manager
6. Rational Robot
7. Rational Pure Coverage

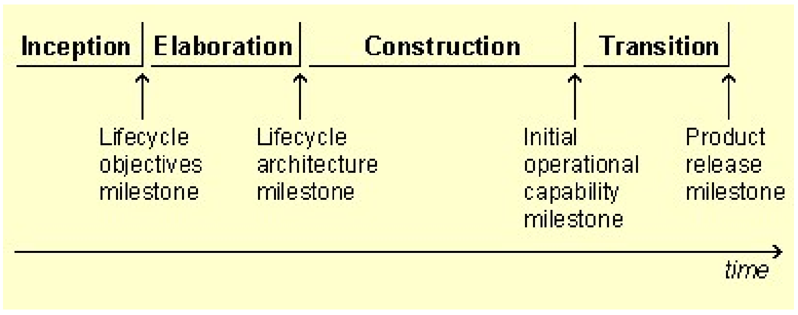
**PRE-REQUISITE:** Knowledge of RUP Process and Phases, Software Engineering development Cycle.

**Pre-Lab: Studying different process models of software project.**

Rational Unified Process: Overview



Rational Unified Process: **Phases**


Inception
Elaboration
Construction
Transition
Lifecycle objectives milestone
Lifecycle architecture milestone
Initial operational capability milestone
Product release milestone
time

**PROCEDURE**

**Step 1: Install RSA/RSE**

**Step 2: Select Rational Unified Process from All programs->RSA/RSE**

**Step 3: Read the details of RUP and draw diagram of RUP and its disciplines**

**Step 4: Select Tool mentors in the left panel.**

**Step 5: Read and Write the Purpose and tool Steps:**

**INPUT/OUTPUT**

**Not applicable (Study experiment)**

**CONTENT OF THE STUDY LAB EXPERIMENT**

**Rational Suite Tools**

**Rational Administrator**

The Rational Administrator centralizes the management of an Administrator project, and manages the association of Rational product datastores and the users and groups for Rational Test. You use Administrator to create a project, connect to a project, create a Test datastore, create an integration between test products and Rational RequisitePro, create an integrated Rational ClearQuest database, and convert previous versions of the repository to the new project.

**Rational RequisitePro**

Rational RequisitePro helps teams organize, prioritize, track, and control changing requirements of a system or application.

- Detailing a Business Use Case Using Rational RequisitePro
- Capturing a Common Vocabulary Using Rational RequisitePro
- Setting Up Rational RequisitePro for a Project
- Developing a Vision Using Rational RequisitePro
- Eliciting Stakeholder Requests Using Rational RequisitePro

- Detailing a Use Case Using Rational RequisitePro
- Managing Dependencies Using Rational RequisitePro
- Reviewing Requirements Using Rational RequisitePro
- Baselining a Rational RequisitePro Project
- Viewing Requirement History Using Rational RequisitePro
- Adding Templates to Your Rational RequisitePro Project
- Archiving Requirements Using Rational RequisitePro

**Rational Rose**

Rational Rose is a graphical component modeling and development tool that uses the industry-standard Unified Modeling Language (UML). Below is a list of tool mentors applicable if you do business modeling.

- Detailing Business Workers and Entities Using Rational Rose
- Finding Business Actors and Use Cases Using Rational Rose
- Detailing a Business Use Case Using Rational Rose
- Finding Business Workers and Entities Using Rational Rose
- Structuring the Business Use-Case Model Using Rational Rose
- Finding Actors and Use Cases Using Rational Rose
- Publishing Web-based Rational Rose Models Using Web Publisher

**Rational PureCoverage**

PureCoverage monitors your program as it runs and reports exactly what parts of your program have and have not been exercised. You can use PureCoverage on whatever scale you want: you can collect data from informal tests of specific program functionalities, or combine data from all the runs controlled by your test suite. Used with Purify, PureCoverage can tell you what parts of your program have not been checked for memory errors and leaks, which helps you significantly improve the quality of the code you ship to customers.

**Tool Steps**

To improve the effectiveness of your testing coverage:

1. Run a program using PureCoverage to collect coverage data
2. Use PureCoverage tools to analyze the coverage data
3. Use your analysis to retest your program more thoroughly

**Rational Purify**

Rational Purify automatically pinpoints hard-to-find, run-time errors in your applications.

Purify detects run-time errors, including memory leaks, both in your own code and in the components your software uses, even when you don't have the source code. It reports memory errors such as array bounds errors, access through dangling pointers, uninitialized memory reads, memory allocation errors, and memory leaks, so that you can resolve them before they do any damage. If you have Rational PureCoverage on your system, Purify can also report how thoroughly you have tested your code for errors.

**Tool Steps**

To test a program using Purify:

1. Run the program with Purify to collect error data and (optionally) coverage data
2. Analyze the Purify error data
3. Correct the errors and rerun the program

If you collected coverage data, take one more step to ensure that you've Purify'd your entire program:

4. Analyze the Purify coverage data to find untested code

**Rational TestManager**

Rational TestManager is the cornerstone of Rational's testing tools, controlling and managing all test activities.

- Performing Test Activities Using Rational TestManager
- Implementing an Automated Test Suite Using Rational TestManager
- Executing a Test Suite Using Rational TestManager
- Creating Performance Test Suites with Rational TestManager
- Creating a Test Case Using Rational TestManager
- Creating a Test Plan Using Rational TestManager

Rational TestManager is the one place to manage all testing activities—planning, design, implementation, execution, and analysis. TestManager ties together testing with the rest of the development effort, joining your testing assets and tools to provide a single point from which to understand the exact state of your project.

This tool mentor is applicable when running Windows 98/2000/NT 4.0.

**Tool Steps**

To use Rational TestManager:

1. Plan the tests
2. Design the tests
3. Implement the tests
4. Execute the tests
5. Evaluate the tests

### Rational Robot

Rational Robot lets you create, modify, and run automated functional tests on your applications.

- Executing Test Suites Using Rational Robot
- Implementing Test Scripts Using Rational Robot
- Setting Up the Test Environment in Rational Robot
- Creating an Automated Performance Test Script Using Rational Robot

For performance testing, scripts are often created by recording them in Robot. When you record a virtual user script, your interactions with the application under test cause protocol-specific communication between the client and the server. Robot records this communication and abstracts it into VU language test scripts after session recording is complete.

A performance test suite in TestManager then executes multiple instances of this script to apply a load to the system under test.

This tool mentor applies to Windows 98/ME/XP/2000 and NT 4.0 platforms.

### Tool Steps

To record a virtual user script using Robot:

1. Start recording the virtual user script
2. Insert timers, blocks, comments, and synchronization points
3. Split script or end session recording
4. Edit the virtual user test script if necessary

### Artifact set

Business Modeling Set

- Business Architecture Document
- Business Entity
- Business Object Model
    - Activity Diagram
    - Association
    - Aggregation
    - Generalization
    - Class Diagram
    - Collaboration Diagram

  Going from Business Models to Systems

    - Sequence Diagram
    - Statechart Diagram
- Business Rules
- Business Use Case
    - Activity Diagram
- Business Use-Case Model
    - Extend-Relationship
    - Include-Relationship
    - Use-Case-Generalization
    - Use-Case Diagram
- Use-Case Package
- Use-Case Storyboard
- User-Interface (General)

Analysis & Design Set

- Data Model
    - Design Class
    - Class Diagram
    - Forward-Engineering
    - Reverse-Engineering
    - Relational Databases
    - Building Web Applications with the UML

Test Set

- Test Plan
- Test Case
- Test Data
- Test Script
- Workload Analysis Model
- Unit Test

**Exercise No.2**

Preparing an SRS using Rational Requisite Pro

**OBJECTIVES**

- ❖ Understanding the IEEE standard of Software Requirement Specification
- ❖ Preparing Requirement Management Plan
- ❖ Preparing SRS document using Rational Requisite Pro for the sample "Student's Course Registration System" project (as per their team choice)
- ❖ Finding Features of the Software System
- ❖ Finding UseCase requirements
- ❖ Finding Stakeholders requirements
- ❖ Finding Supplementary requirements
- ❖ Mapping Features with other requirements
- ❖ Preparing Attribute Matrix
- ❖ Preparing Traceability Matrix

**Pre-Requisite: IEEE SRS Standards**

**Pre-Lab: Online CD purchasing system**

**A discussion of requirements management**

Requirements management is a systematic approach to finding, organizing, documenting and tracking the changing requirements of a software application or system. Requirements are capabilities and objectives to which the software or system being built must conform. Requirements are found in vision documents, problem reports, feature requests, other products, business practices, designs, specifications, quality assurance plans, test cases and prototypes.

The Standish Group reported forty percent of software projects fail. This failure is attributed to poor requirements management; incorrect definition of requirements from the start of the project and poor requirements management throughout the development lifecycle. One way to help ensure the success of a project is to implement an effective requirements management process. RequisitePro can help you effectively organize and manage requirements as well as trace the impact of changes on schedules, costs and resources.

Requirements management offers numerous benefits. These include improved predictability of a project's schedules and deliverables, reduced project costs and delays, improved software quality, improve team communication, improved compliance with standards and regulations (the Capability Maturity Model (**CMM**), the Department of Defense, ISO 9000, FDA).

---

**Implementing an effective requirements management process**

Here are five essential steps to help you manage your requirements and achieve your project goals.

1. Identify your requirements.
2. Organize and prioritize your requirements.
3. Analyze and understand the factors associated with your requirements.
4. Refine and expand your requirements.
5. Manage changes to your requirements.

*Identify your requirements.* It is very important to identify and manage requirements from the beginning of a project. Here are some factors to consider when setting priorities at the beginning of the requirements management process.

- How does the requirement add to product functionality, usability, reliability and performance?
- Will the requirement being considered be worth the effort, given your schedule constraints?
- Is the requirement feasible given risks associated with it?
- If the requirement is implemented, how will it impact your ability to maintain the product?

*Organize your requirements*. Look at your schedule. Do you have time to complete all of your high priorities? Setting expectations at this stage will help team stay on schedule and close to budget. Ask yourself these questions:

- What are the needs of my users?
- What are primary goals of this project?
- What key features are parts of my requirements list?
- What kinds of documents are needed to record and trace requirements from wish lists to test scripts?
- Determine which tasks are necessary to accomplish requirements, who is responsible for completing them and which requirements are parts of the project's critical path.

*Understand and analyze the factors associated with your requirements*. If you have a long list of requirements, classify them in categories that make sense to your team. Asking the following questions will help you analyze these factors more efficiently.

- How will each requirement benefit your customers?

- How much effort is necessary to accomplish the requirement?
- How will each requirement affect your budget and schedule?
- What are the risks associated with each requirement?

*Refine and expand your requirements*. Refining requirements early on in the process will help prevent requirement errors later. The time, money and effort required to fix errors at the end of the process could cost one hundred times more than the time spent planning now.

*Manage changes to your requirements*. Track the progress of your requirements and trace changes to them. Monitor changes, analyze the impact of change and communicate this to your team.

**SAMPLE INPUT**
**i) Course Registration System**

**Problem Statement**

You are tasked with developing a new student registration system. The college would like a new client-server system to replace its much older system developed around mainframe technology. The new system will allow students to register for courses and view report cards from personal computer attached to the campus LAN. Professors will be able to access the system to sign up to teach courses as well as record grades.

Due to a decrease in federal funding, the college cannot afford to replace the entire system at once. The college will keep the existing course catalog database where all course information is maintained. This database is an ingress relational database running on a DEC VAX. Fortunately the college has invested in an open SQL interface that allows access to this database from College’s Unix servers. The legacy system performance is rather poor, so the new system must ensure that access to the data on the legacy system occurs in timely manner. The new system will access course information from the legacy database but will access course information from the legacy database but will not update it. The registrar’s office will continue to maintain course information through another system.

At the beginning of each semester, students may request a course catalog containing a list of course offerings for the semester. Information about each course, such as professor, department, and prerequisites, will be included to help students make informed decisions.

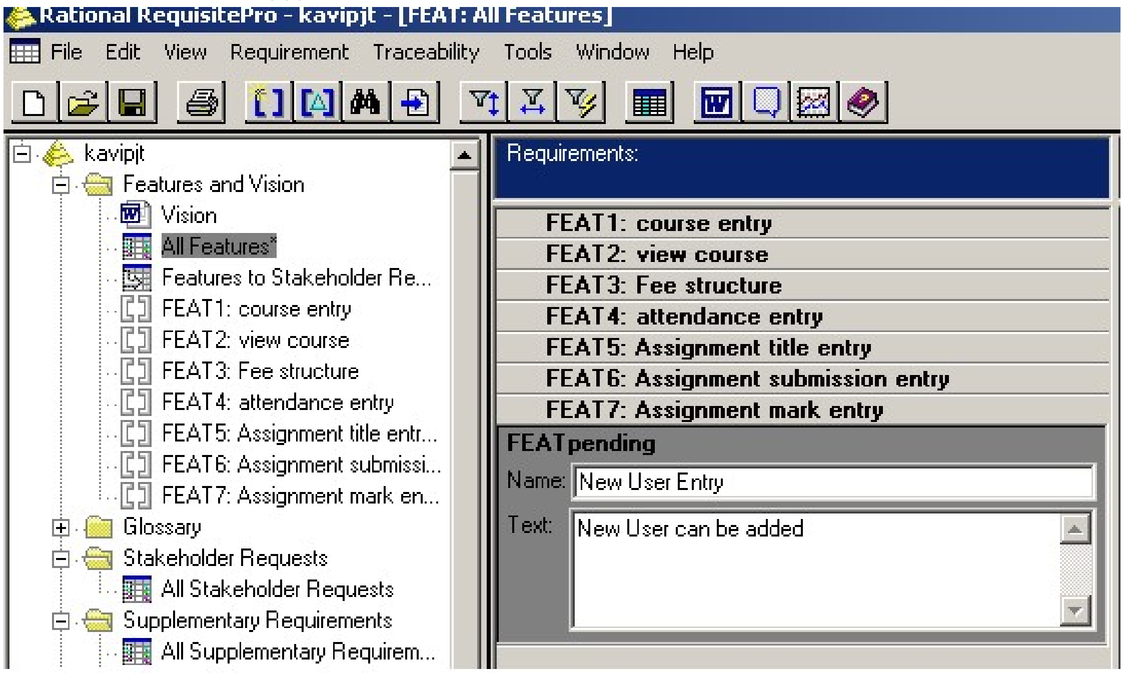
The new system will allow students to select four course offerings for the coming semester. In addition, each student will indicate two alternative choices incase the student cannot be assigned to a primary selection. Course offerings will have a maximum of ten students and a minimum of three students. A course offering with fewer than three students will cancelled. For each semester, there is a period of time that students can change their schedule. Students must be able to access the system during this time to add or drop courses. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the semester. If a course fills up during the actual registration process, the student must be notified of the change before submitting the schedule for processing.

At the end of the semester, the student will be able to access to view an electronic report card. Since student grades are sensitive information, the system must employ extra security measures to prevent unauthorized access.

Professors must be able to access the on-line system to indicate which courses they will be teaching. They will also need to see which students signed up for their course offerings. In addition, the professors will be able to record the grades for the students in each class.

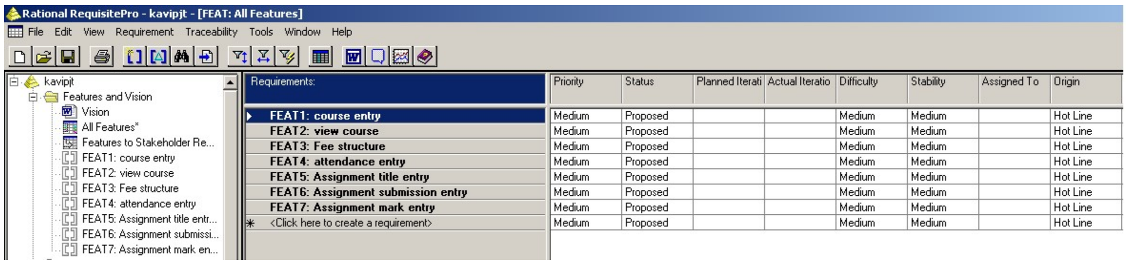
**System Features Entry:**

**Course Entry**
**View Course**
**Fee Structure**
**etc..**


Rational RequisitePro - kavipjt - [FEAT: All Features]
File Edit View Requirement Traceability Tools Window Help
kavipjt
Features and Vision
Vision
All Features*
Features to Stakeholder Re...
FEAT1: course entry
FEAT2: view course
FEAT3: Fee structure
FEAT4: attendance entry
FEAT5: Assignment title entr...
FEAT6: Assignment submissi...
FEAT7: Assignment mark en...
Glossary
Stakeholder Requests
All Stakeholder Requests
Supplementary Requirements
All Supplementary Requirem...
Requirements:

FEAT1: course entry
FEAT2: view course
FEAT3: Fee structure
FEAT4: attendance entry
FEAT5: Assignment title entry
FEAT6: Assignment submission entry
FEAT7: Assignment mark entry
FEATpending
Name: New User Entry
Text: New User can be added

**SAMPLE OUTPUTS**
**System Features Attributes**

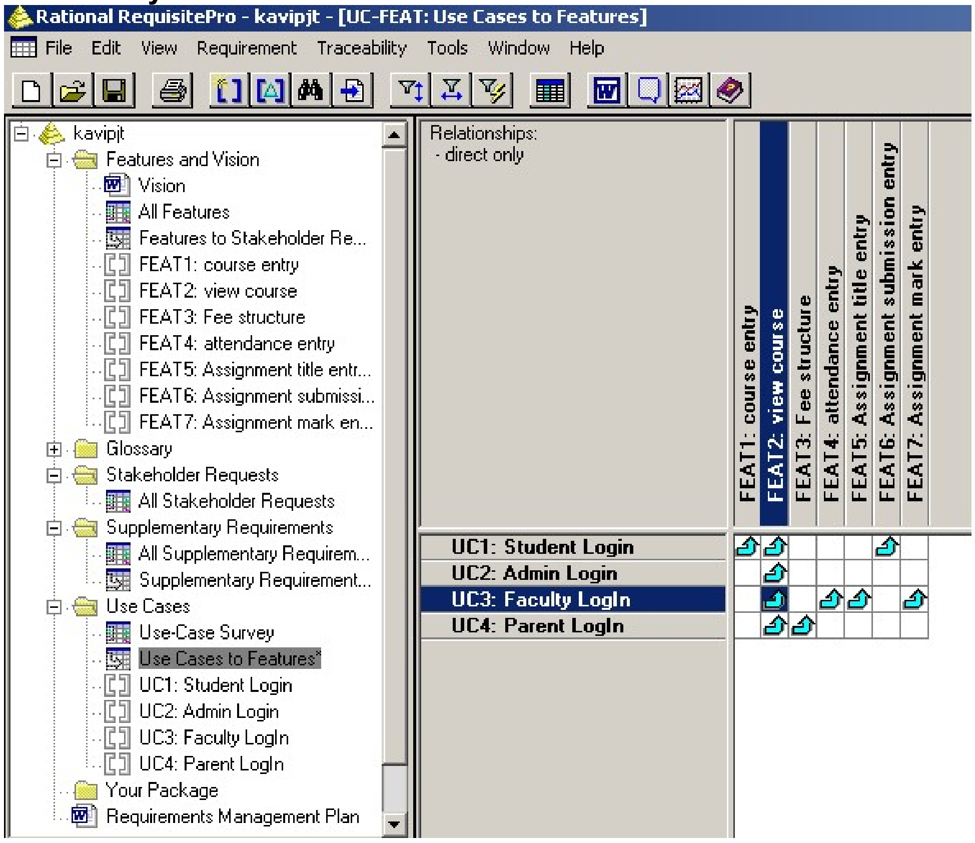


Rational RequisitePro - kavipjt - [FEAT: All Features]

- kavipjt
  - Features and Vision
    - Vision
    - All Features*
    - Features to Stakeholder Re...
    - FEAT1: course entry
    - FEAT2: view course
    - FEAT3: Fee structure
    - FEAT4: attendance entry
    - FEAT5: Assignment title entr...
    - FEAT6: Assignment submissi...
    - FEAT7: Assignment mark en...

| Requirements: | Priority | Status | Planned Iterati | Actual Iteratio | Difficulty | Stability | Assigned To | Origin |
|---|---|---|---|---|---|---|---|---|
| **FEAT1: course entry** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT2: view course** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT3: Fee structure** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT4: attendance entry** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT5: Assignment title entry** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT6: Assignment submission entry** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| **FEAT7: Assignment mark entry** | Medium | Proposed | | | Medium | Medium | | Hot Line |
| <Click here to create a requirement> | Medium | Proposed | | | Medium | Medium | | Hot Line |

## Traceability Matrix



Rational RequisitePro - kavipjt - [UC-FEAT: Use Cases to Features]

- kavipjt
  - Features and Vision
    - Vision
    - All Features
    - Features to Stakeholder Re...
    - FEAT1: course entry
    - FEAT2: view course
    - FEAT3: Fee structure
    - FEAT4: attendance entry
    - FEAT5: Assignment title entr...
    - FEAT6: Assignment submissi...
    - FEAT7: Assignment mark en...
  - Glossary
  - Stakeholder Requests
    - All Stakeholder Requests
  - Supplementary Requirements
    - All Supplementary Requirem...
    - Supplementary Requirement...
  - Use Cases
    - Use-Case Survey
    - Use Cases to Features*
    - UC1: Student Login
    - UC2: Admin Login
    - UC3: Faculty LogIn
    - UC4: Parent LogIn
  - Your Package
  - Requirements Management Plan

Relationships:
- direct only

| | FEAT1: course entry | FEAT2: view course | FEAT3: Fee structure | FEAT4: attendance entry | FEAT5: Assignment title entry | FEAT6: Assignment submission entry | FEAT7: Assignment mark entry |
|---|---|---|---|---|---|---|---|
| **UC1: Student Login** | ✓ | ✓ | | | | ✓ | |
| **UC2: Admin Login** | | ✓ | | | | | |
| **UC3: Faculty LogIn** | | ✓ | | ✓ | ✓ | | ✓ |
| **UC4: Parent LogIn** | | ✓ | ✓ | | | | |

Exercise No.3

## Use case and activity Diagram (Rational Rose)

## Objectives

- ❖ Understanding and deriving the use cases of given scenario
- ❖ Understanding objects and its actions of the software system
- ❖ Understanding the symbols of UML which are necessary to Use case and activity diagram
- ❖ Identifying actors, actions, prerequisite, post-requisite and constraints of each scenario
- ❖ Preparing Use case and activity diagram using Rational rose

**PRE-REQUISITE**

**Knowledge of UML notations and OOAD**

**Pre-Lab: Online CD purchasing system**

**SAMPLE INPUT**

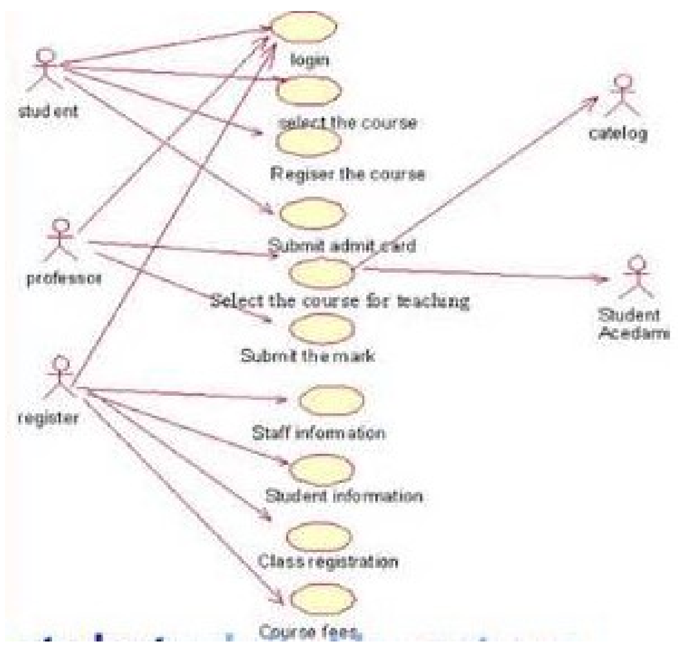
**Table of contents**

1. Course Registration System Use-Case Model Main Diagram
2. Close Registration
3. Login
4. Maintain Professor Information
5. Maintain Student Information
6. Register for Courses
7. Select course to teach
8. Submit grades
9. View report cards

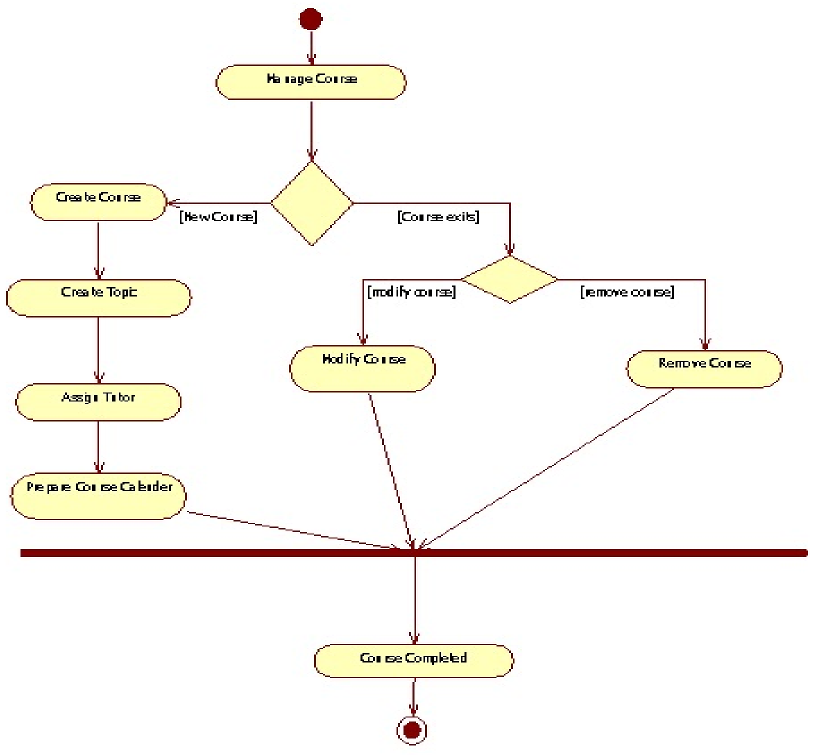
**SAMPLE OUTPUT**

**i) Course Registration System**

## A) USE CASE DIAGRAM



## B) ACTIVITY DIAGRAM



## Exercise No.4

**Creating a Sequence, Collaboration and Class Diagram.**

**Objectives**

- ❖ Understanding and deriving the sequences of given scenario
- ❖ Understanding objects and its classes of the software system
- ❖ Understanding the symbols of UMLwhich are necessary to Sequence, Collaboration and class diagram
- ❖ Preparing Sequence, Collaboration and class diagram using Rational rose

**Create a Sequence Diagram**

In this lab, you'll practice creating a sequence diagram from a use case's flow of events.

**Task 1: Add Sequence Diagram to the Browser.**
**Task 2: Add Actor and Classes**
**Task 3: Add Object Messages**
**Task 4: Add Responsibilities to Object Messages**

**Collaboration Diagram**

**Auto Generate Collaboration Diagram**
In this lab, you'll auto generate the collaboration diagram from the sequence diagram you've already created. You will also run the Show Unresolved Objects and Show Unresolved Messages reports.

1. Auto generate the Maintain Personal Planner – Basic Flow Collaboration diagram.
2. Run Reports.

**Lab Outcome**

- ➢ A collaboration diagram showing the maintain Personal Planner's basic flow.

**Task 1: Auto Generate the Collaboration Diagram**
**Task 2: Run reports**

**Create Class Diagram**

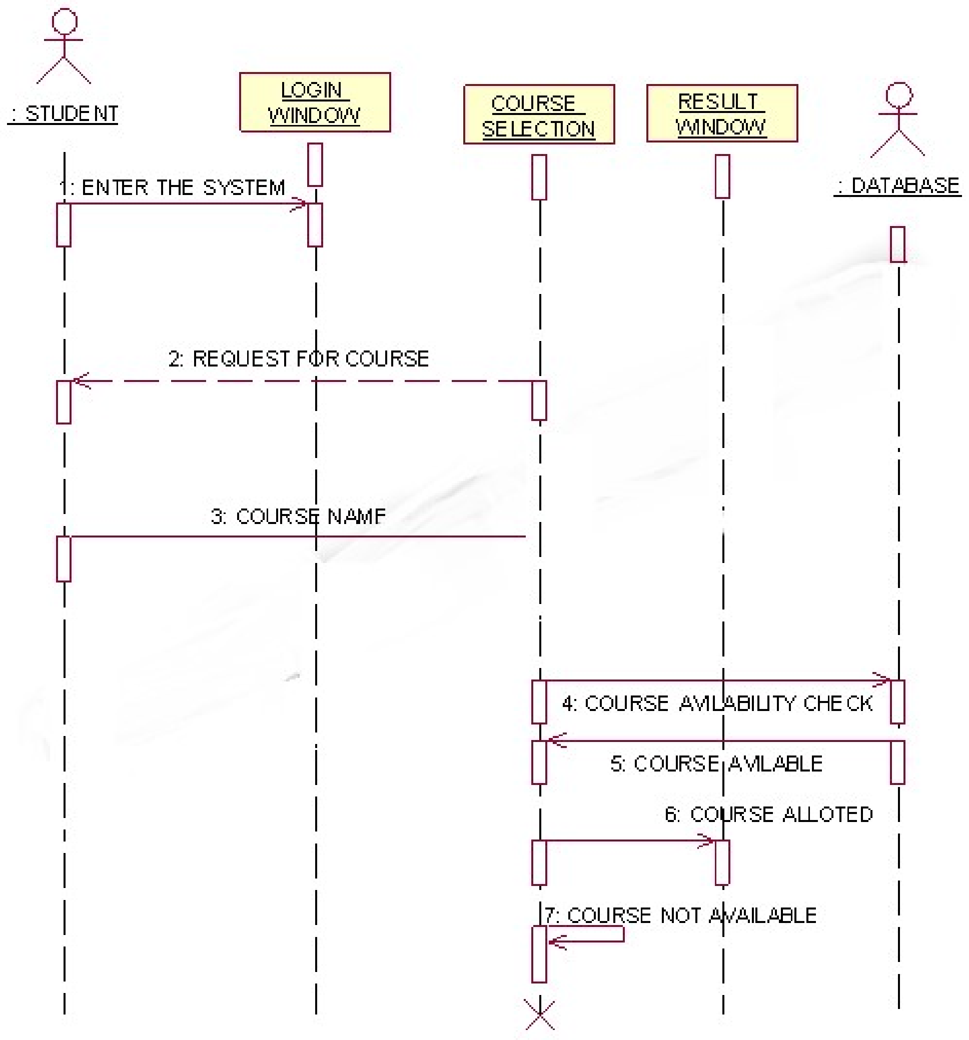
In this lab you will create a class diagram to show the classes and relationship that participates in the basic flow of the maintain personal profile use-case realization

Task 1: Add class diagram to the browser

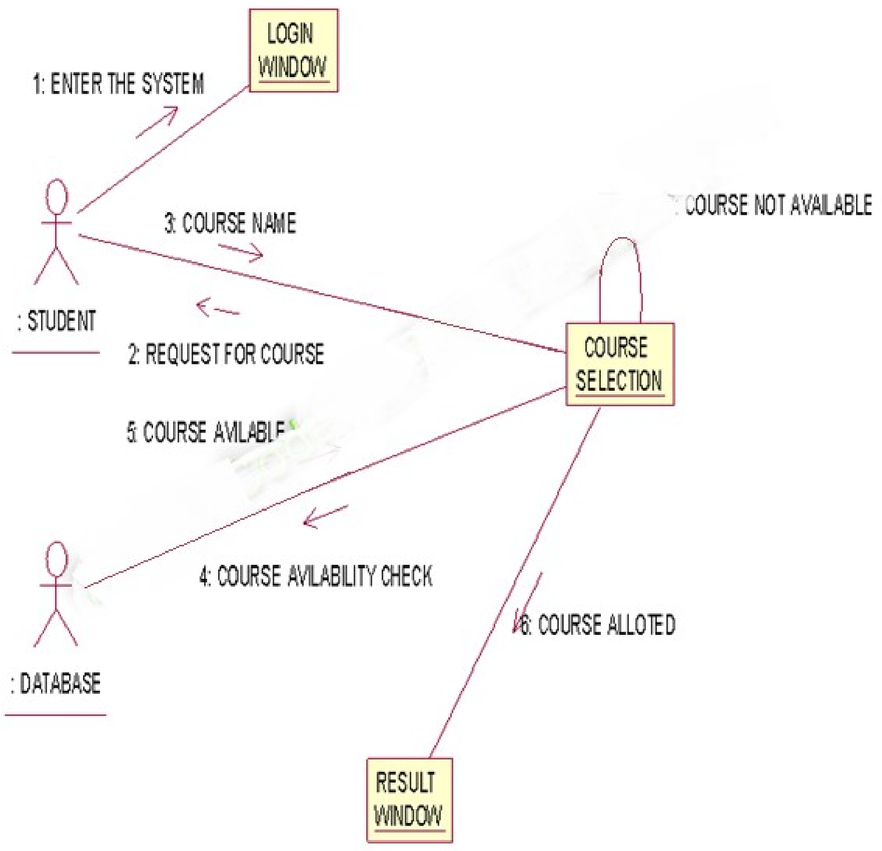
Task 2: Add Classes to the diagram
Task 3: Add Associations
Task 4: Add role names and multiplicity

**SAMPLE INPUT/OUTPUT**

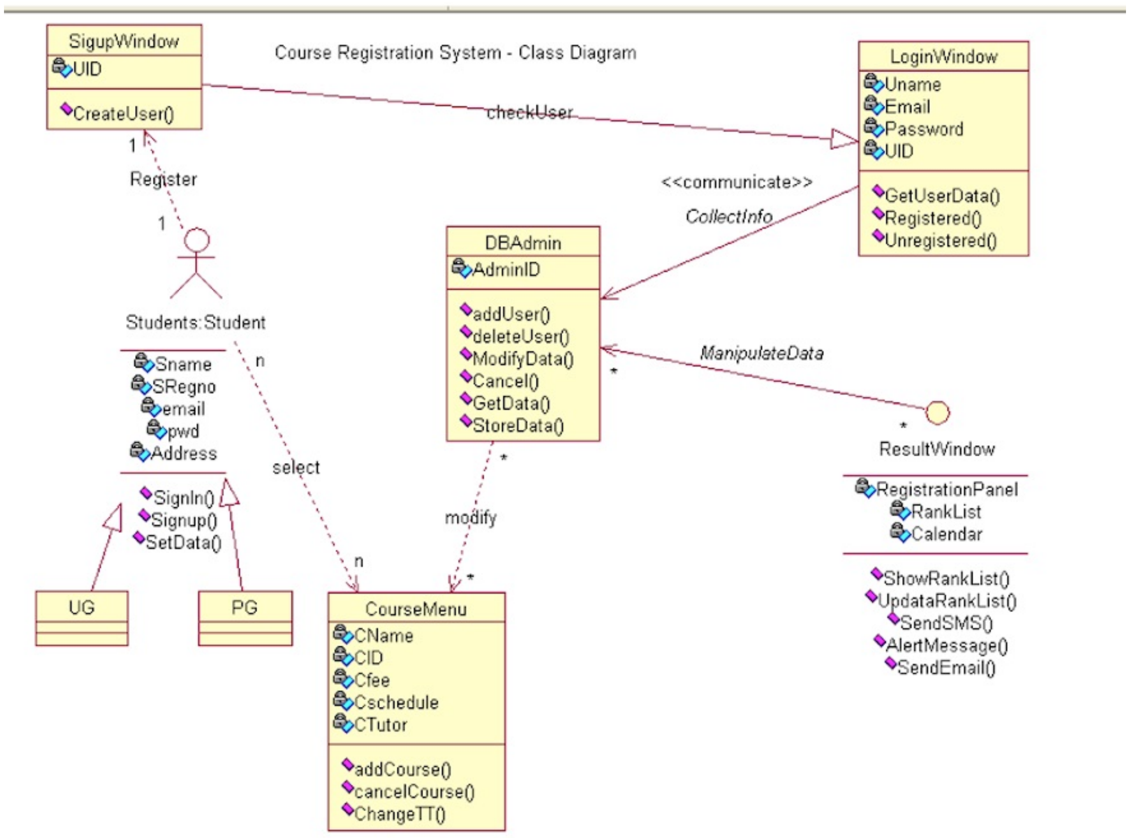
i. Course Registration System
a) Sample Sequence diagram



b) Sample Collaboration diagram

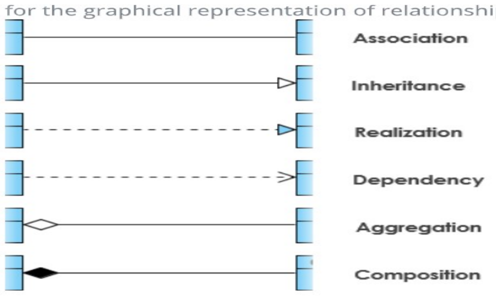


c) Sample Class diagram



### Class Relationships

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types: (Refer to the figure on the right for the graphical representation of relationships).



**Exercise No.5: Forward and Reverse Engineering**

**Objectives**

**Understanding and performing Forward and Reverse Engineering**

**Pre-Requisite:**

- **Knowledge of Java or Visual Basic**
- **Knowledge of Forward Engineering and Reverse Engineering**

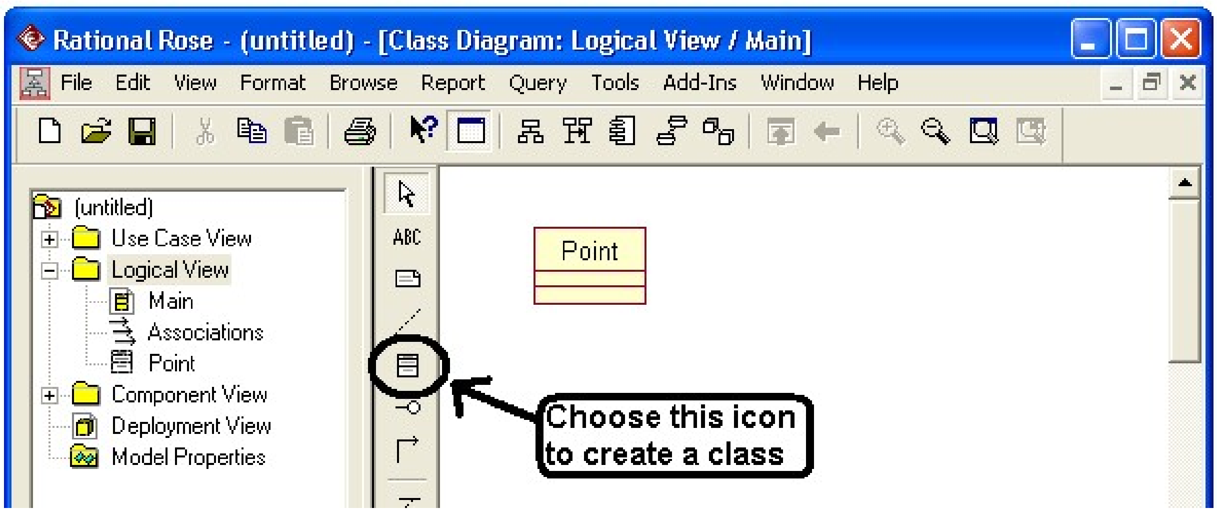
**Pre-Lab: Online CD purchasing system**

**SAMPLE INPUT/OUTPUT:**

**<u>Forward Engineering Example with Rational Rose:</u>**

## <u>Creating a class</u>

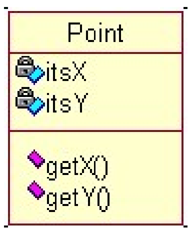
1. In the **logical view** of your new model create a class called **Point.**


Rational Rose - (untitled) - [Class Diagram: Logical View / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
(untitled)
Use Case View
Logical View
Main
Associations
Point
Component View
Deployment View
Model Properties
ABC
Point
Choose this icon to create a class

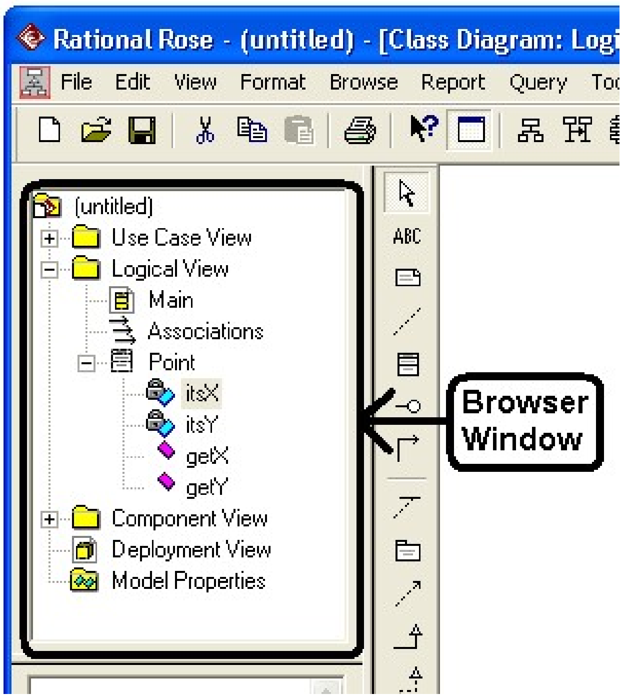
2. Right click the class Point and click **New Attribute.** Name the attribute **itsX** and **itsY**.

3. Right click again, click **New Operation.** Name the new operation **getX(),** and **getY().**

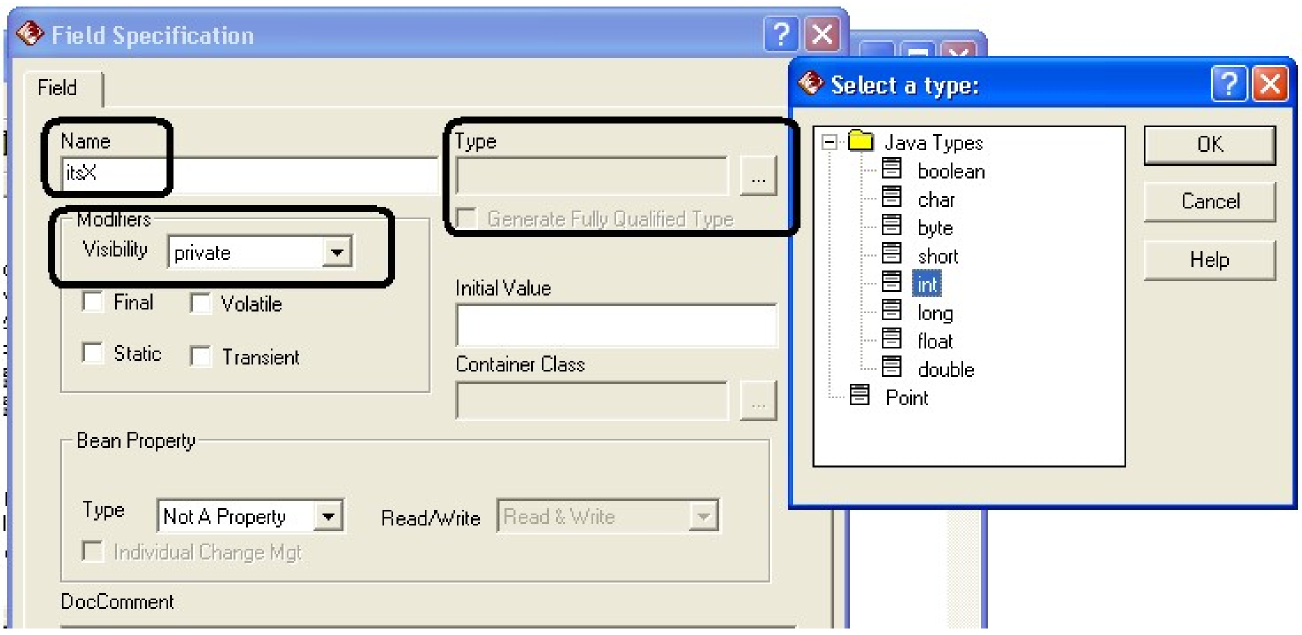
4. Your model should now look as follows:



5. In the browser window, click on **Logical View** and then expand on **Point.**



5. Double click **itsX** and set its type to int and also set its various modifiers [private, etc.], also for **itsY.**


Field Specification

Field

Name: itsX

Type:

Generate Fully Qualified Type

Modifiers

Visibility: private

Final

Volatile

Static

Transient

Initial Value

Container Class

Bean Property

Type: Not A Property

Read/Write: Read & Write

Individual Change Mgt

DocComment

Select a type:

Java Types
- boolean
- char
- byte
- short
- int
- long
- float
- double

Point

OK

Cancel

Help

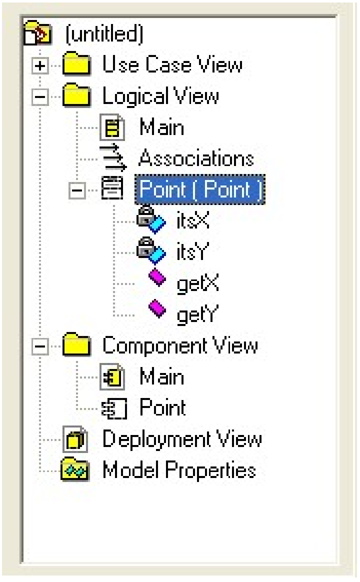
6. Similarly, double-click **getX()** and **getY()** in the browser and set its various attributes {return type, etc.}

7. In the browser, left-click the **Component View** to create a new component called **Point**.

8. In the browser, drag the Point class to the Point component to assign the class to the Component.

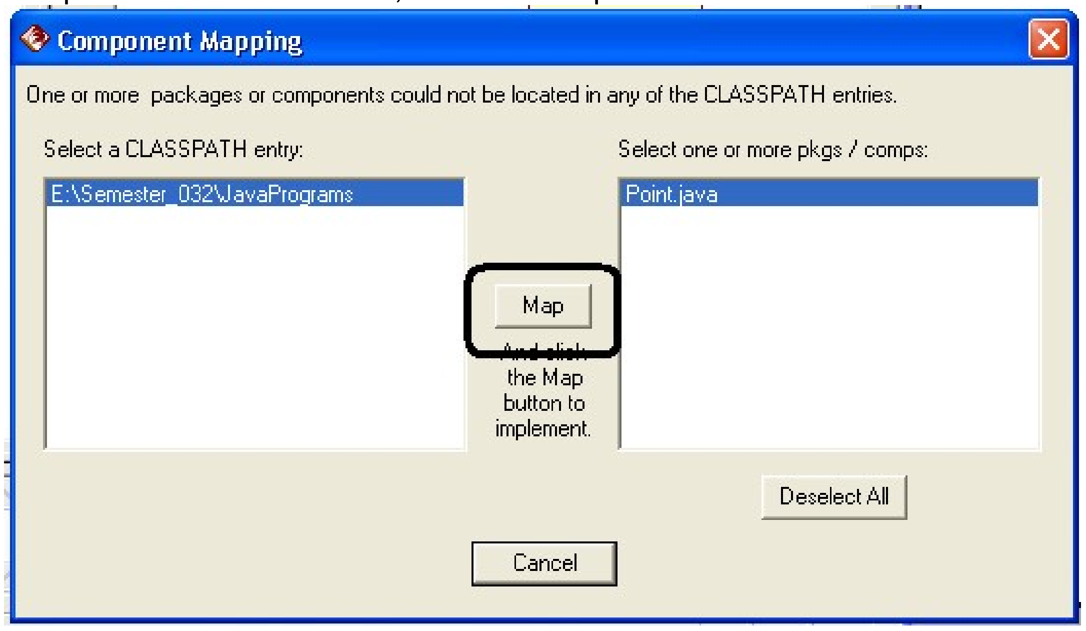
9. Check the browser to make sure the class now has the component name next to it in parenthesis.

10. The browser should now look as follows:


```
(untitled)
+ Use Case View
- Logical View
    Main
    Associations
  - Point ( Point )
      itsX
      itsY
      getX
      getY
- Component View
    Main
    Point
  Deployment View
  Model Properties
```

11. In the browser or in the diagram, right-click the Point class, click **Java > Generate Java.**

12. Since this is the first time you've generated code for this model, Rational Rose displays a mapping dialog prompting you to map the file it will generate to a directory in your class path. Click on the class path name to select it, click on the component name to select it, then click Map:



One or more packages or components could not be located in any of the CLASSPATH entries.

Select a CLASSPATH entry:

E:\Semester_032\JavaPrograms

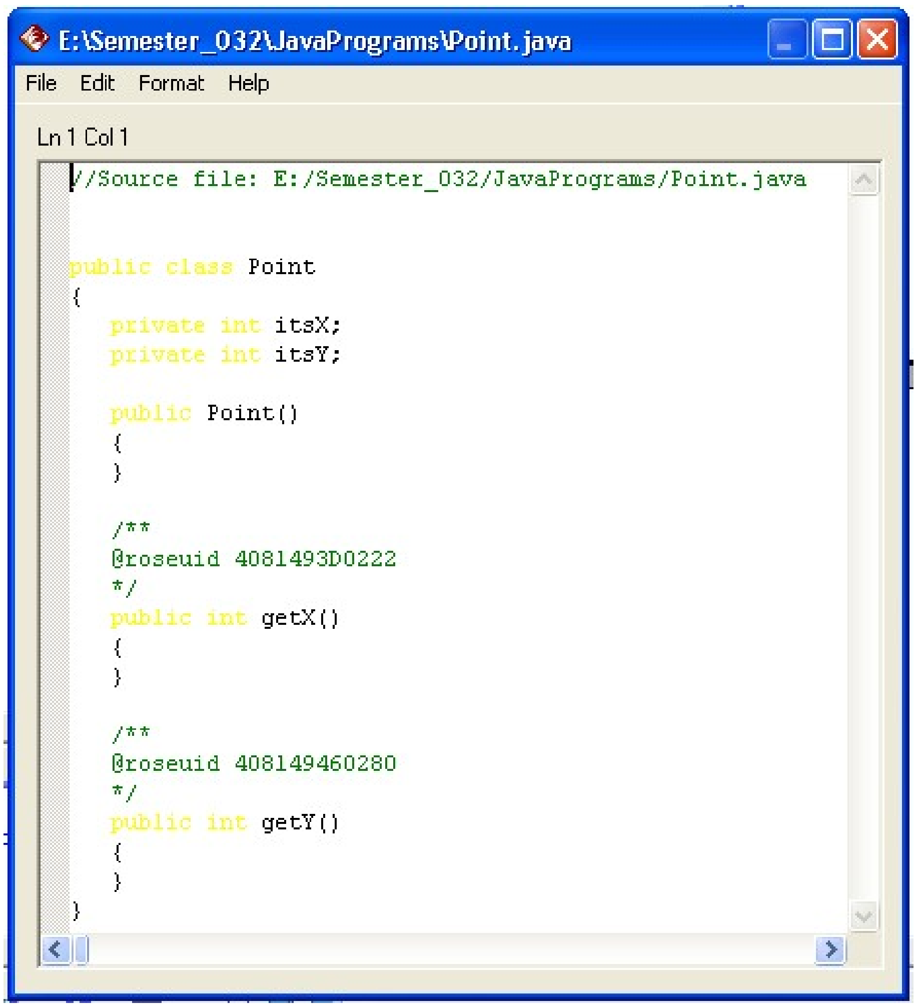
Select one or more pkgs / comps:

Point.java

And click the Map button to implement.

13. If code generation is successful, a "successful" dialog will be seen:

14. Click on **Java > Browse Java Source** to view the Java source:



```
//Source file: E:/Semester_032/JavaPrograms/Point.java


public class Point
{
   private int itsX;
   private int itsY;

   public Point()
   {
   }

   /**
   @roseuid 4081493D0222
   */
   public int getX()
   {
   }

   /**
   @roseuid 408149460280
   */
   public int getY()
   {
   }
}
```

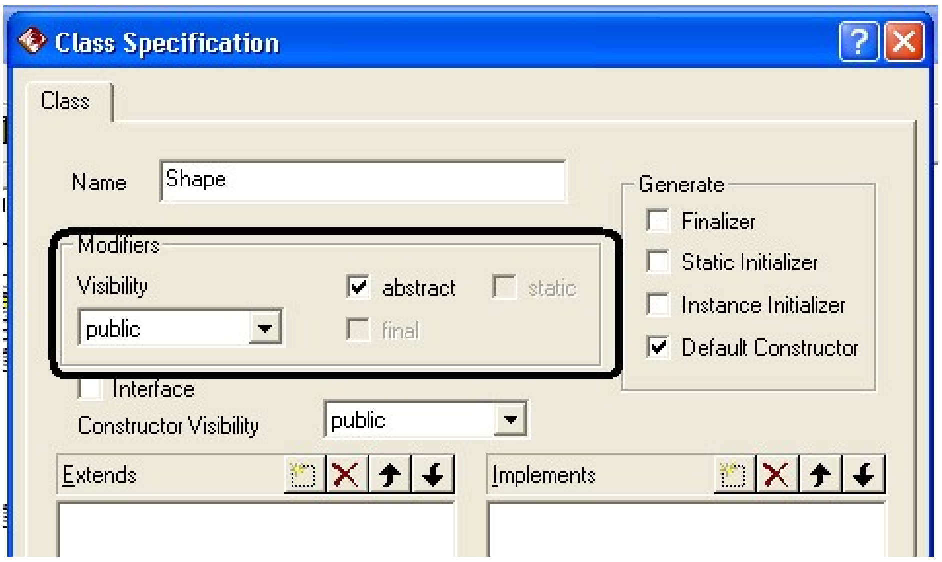
15. Complete your code.

16. Save your model as **Lab09.mdl** file.

## <u>Creating an abstract Class</u>

1. Use the same technique of creating a class, create an abstract class name **Shape**.

2. Double click on the class Shape to open Class specification and change its modifier into abstract.


Class Specification

Class

Name: Shape

Modifiers
Visibility: public
[✓ abstract] static
final

Generate
Finalizer
Static Initializer
Instance Initializer
[✓ Default Constructor]

Interface

Constructor Visibility: public

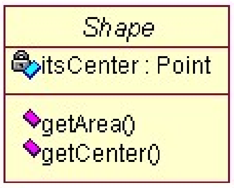
Extends

Implements

3. Add a member **itsCenter** with the type of **Point**.

4. Add a function **getCenter()** with return type is **Point**.

5. Add a function **getArea()** with return type is **double** and its modifier is **abstract**, your model should look like the following:

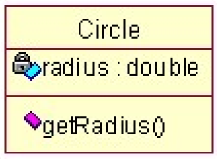


6. Do step 7 – 16 as describe above, to generate Java file of this model, and save the model.

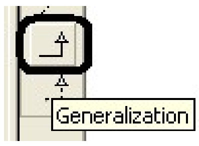
## Creating Relationship between Classes

### Inheritance

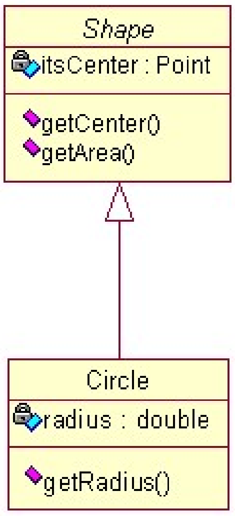
1. Create a new concrete class **Circle** with **radius** (double) as a member and **getRadius()** (returning double) as its operation. It should be look like :



2. Generate inheritance relationship between Circle (subclass) and Shape (super class). Click on generalization icon (see below figure) or select **Tool → Create → Generalization.**

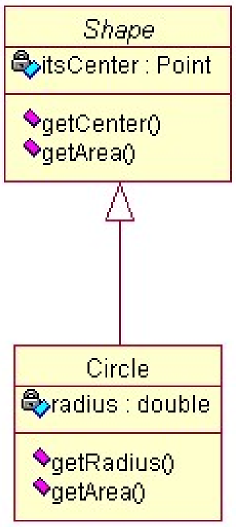


3. While the pointer is displaying up arrow (generalization mode), drag it from Circle class to Shape abstract class. Your model should look like :



4. Validate your model by double-clicking the class Circle, then verify that abstract class Shape has been added in extends fields.

5. Do step 7 – 16 as describe above, to generate Java file of this model, and save the model. If it is successful then the getArea() operation will automatically added to Circle class. As following figure:



## Association, Composition, Aggregation, Navigation and Multiplicity

**Association** relationships capture the static relationships between entities. These generally relate to one object having an instance of another as an attribute or being related in the sense of owning (but not being composed of). For example a Shape has an association to its “center” point.

**Aggregation** relationships define whole/part relationships. The weak form of aggregation is denoted with an open diamond. This relationship denotes that the aggregate class (the class with the white diamond touching it) is in some way the “whole”, and the other class in the relationship is somehow “part” of that whole.

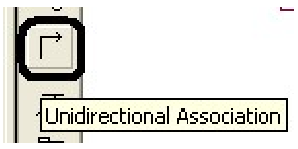
**Composition** relationships are a strong form of containment or aggregation. The black diamond represents composition.

**Multiplicity** defines how many objects participate in a relationship.

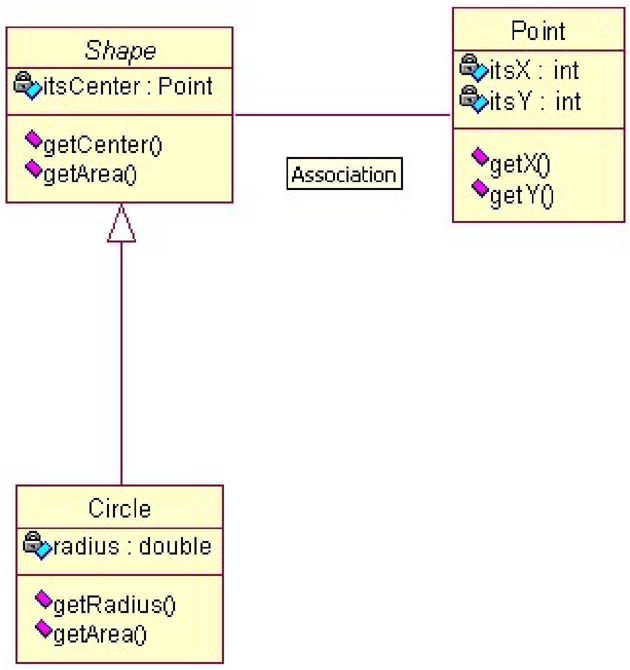
**Navigation** is the direction of the relationships between the classes.

Suppose that we would like to create the Association relationship between Shape and Point. These are steps:

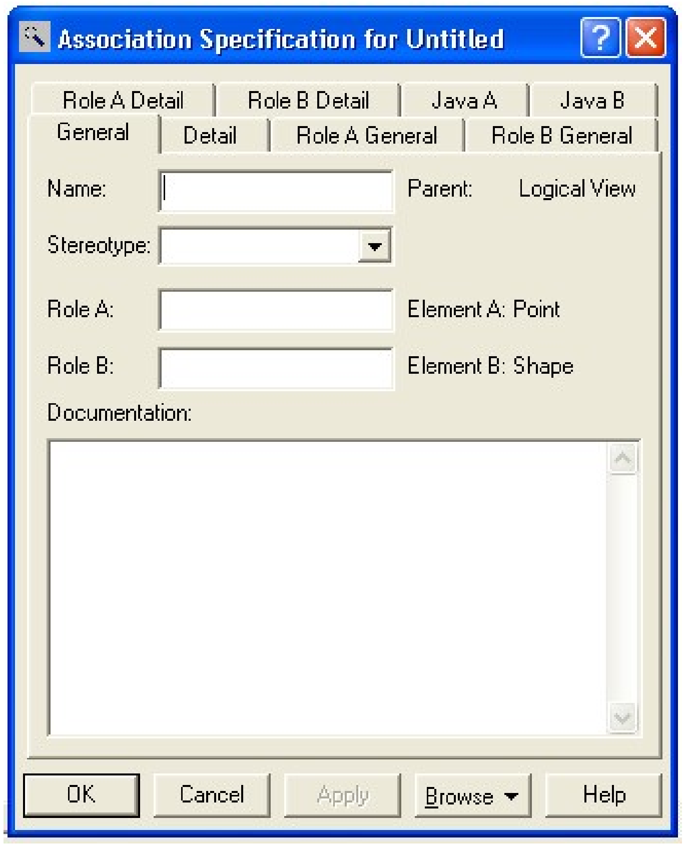
1. Select Association icon (as shown bellow) or select **Tool → Create → Association.**



2. While the pointer is displaying up arrow (Association mode), drag it from Shape class to Point class. Your model should look like:

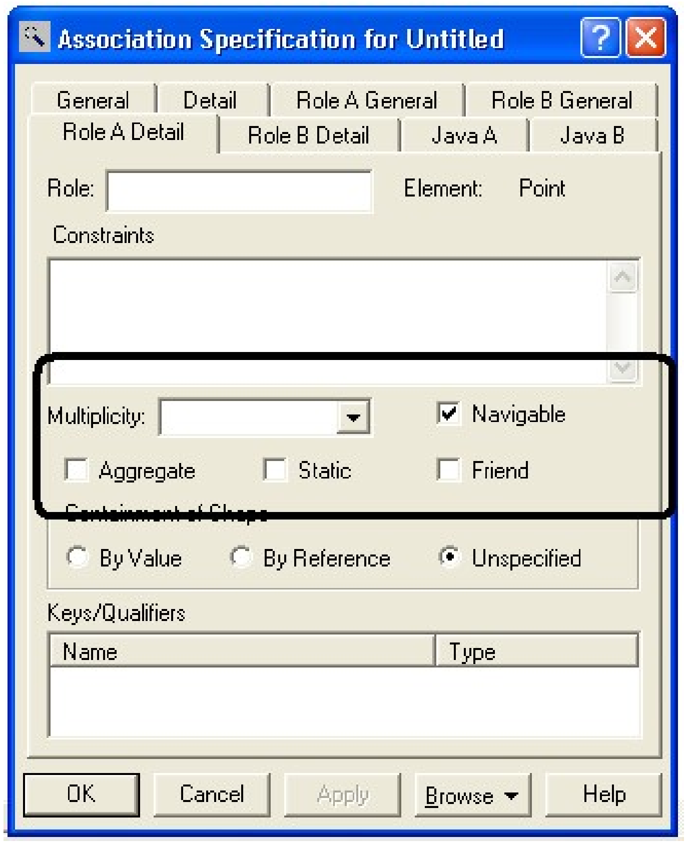


3. Double click on the line between Shape and Point to open Association specification, to update the properties such as Name, the Roles etc.



Name:
Stereotype:
Parent: Logical View
Role A:
Element A: Point
Role B:
Element B: Shape
Documentation:

4. To modify type of relationship to Aggregate / Composition, or to add navigations and multiplicity, select either Role A detail or Role B detail accordingly.



Role:

Element: Point

Multiplicity:

[✓ Navigable]

[ ] Aggregate [ ] Static [ ] Friend

( ) By Value ( ) By Reference [● Unspecified]

**Reverse Engineering:**

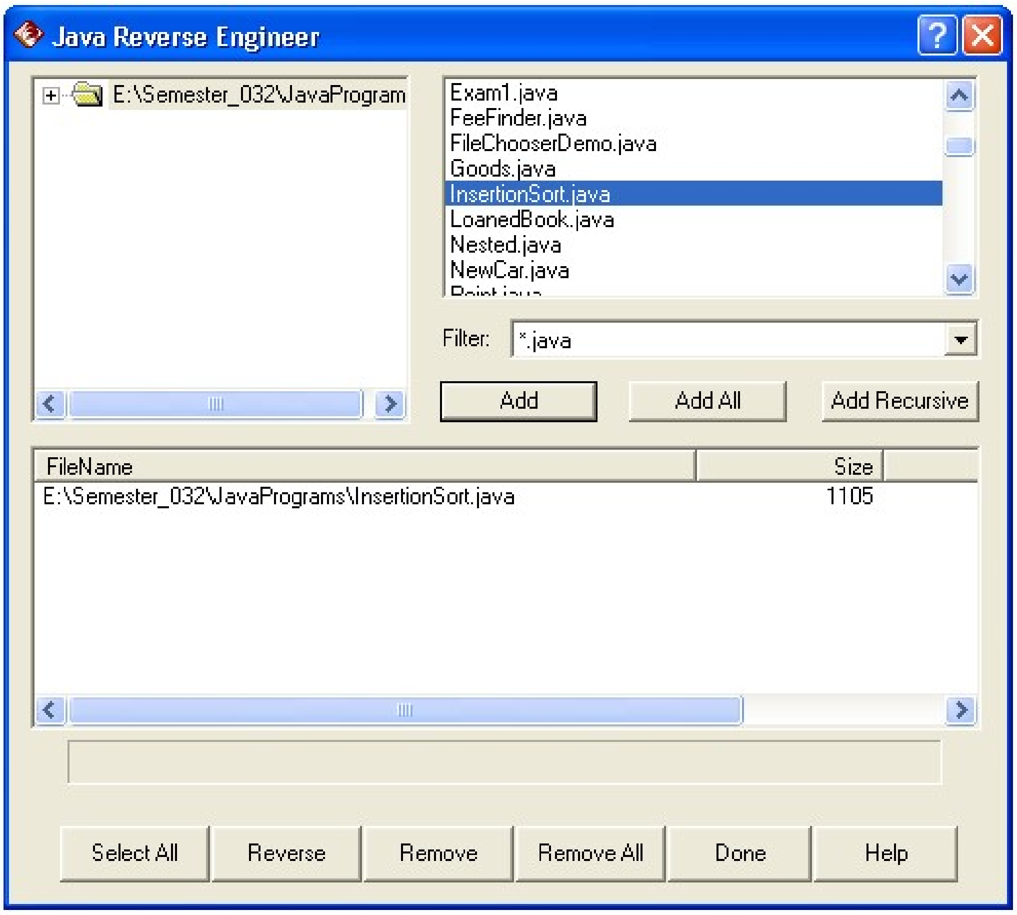
Consider the following file:

```
import java.util.Random;

class ArrayUtil {
   public static int[] randomIntArray(int length, int n) {
      int[] a = new int[length];
      Random generator = new Random();

      for (int i = 0; i < a.length; i++)
         a[i] = generator.nextInt(n);

      return a;
   }
       public static void swap(int[] a, int i, int j) {
      int temp = a[i];
      a[i] = a[j];
      a[j] = temp;
   }

   public static void print(int[] a) {
      for (int i = 0; i < a.length; i++)
         System.out.print(a[i] + " ");
      System.out.println();
   }
}

class InsertSort {
   public static void sort(int[] a) {
       int current, n, j;
       for (n = 1; n < a.length; n++) {
             current = a[n];
                  for (j = n; j>0 && current < a[j-1]; j--)
                          a[j] = a[j-1];
                 a[j] = current;
      }
   }}
public class InsertionSort {
   public static void main(String[] args) {
      int[] a = ArrayUtil.randomIntArray(20, 100);
      ArrayUtil.print(a);
           InsertSort.sort(a);
      ArrayUtil.print(a);
   }}
```

1. The following steps will be taken to reverse-engineer the model.

2. Check your class path by opening the Java Project Specification: click **Tools** > **Java** > **Project Specificatio**n. If needed, use **Directories** field of the Class Path tab to extend your classpath. Note that you must have a class path to the Java API library in order to reverse engineer.

3. In the browser or in a class or component diagram, right-click to display the shortcut menu. Click **Java** > **Reverse Engineer**.This displays the **Java Reverse Engineer** dialog:



| FileName | Size |
|---|---|
| E:\Semester_032\JavaPrograms\InsertionSort.java | 1105 |

4. Click the Java file whose code you want to reverse engineer.

5. Click a folder in the tree to display the list of files it contains. (Traverse the tree to find the folder or subfolder that contains the files to be reverse engineered.)

6. Do one of the following to place the Java files of the type you selected into the **Selected Files** list:

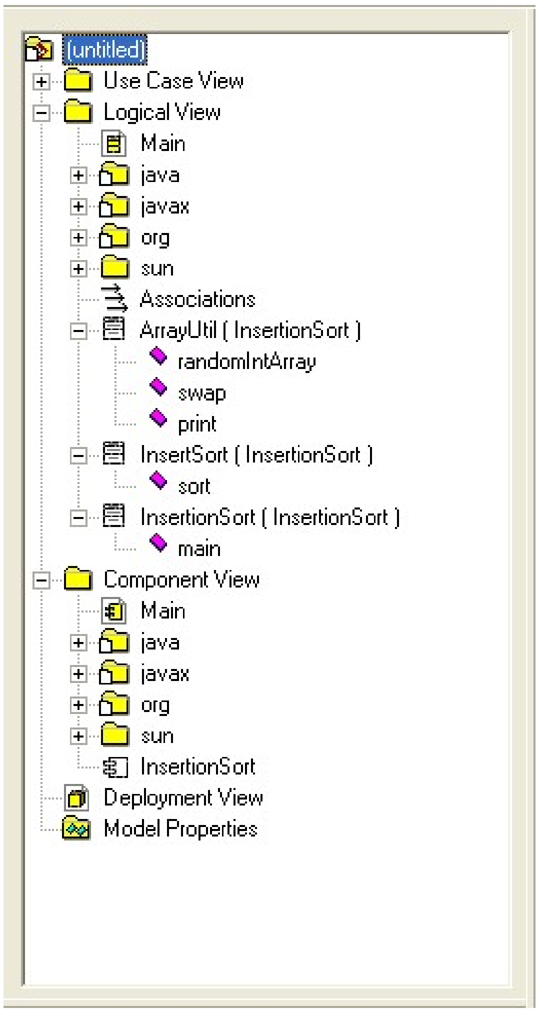
- In the list box, click on one or more individual files and click **Add**.
- Click **Add All** to add all of the files.
- Click **Add Recursive** to take all of the files of the selected file type that are contained in the currently selected folder and all of its subfolders and place them in the **Selected Files** list. (Use **Add Recursive** to select files in multiple folders without having to search for and select each one.)

7. Click on one or more files in the **Selected Files** box or click **Select All** to confirm the entire list of files to reverse engineer.

8. Click **Reverse** to create or update your model from the Java source you specified.

9. If Rose encounters an error, a message appears and you should check the Rose Log Window for a description of the error. Otherwise, click **Done** to close the dialog.

When reverse engineering is complete, you can view the new model elements in the browser, create your own logical views, and drag and drop the new elements into the views you create.



(untitled)
Use Case View
Logical View
Main
java
javax
org
sun
Associations
ArrayUtil ( InsertionSort )
randomIntArray
swap
print
InsertSort ( InsertionSort )
sort
InsertionSort ( InsertionSort )
main
Component View
Main
java
javax
org
sun
InsertionSort
Deployment View
Model Properties

**Exercise No.6: Memory leak testing with Rational Purify**

**Objectives**

- ❖ **Locate reading and writing beyond the memory bounds**
- ❖ **Using uninitialized memory**
- ❖ **Reading and writing freed memory**
- ❖ **Leaking memory and file descriptors**

**Pre-Requisite: Knowledge of memory leaking**

**Pre-Lab: Online CD purchasing system**

Rational Purify is the most comprehensive runtime error detection tool available. It checks all the code in your program, including any application, system, and third-party libraries. Purify works with complex software applications, including multi-threaded and multi-process applications.

**PROCEDURE:**

Write a C program with memory leaks (atleast 10 errors)

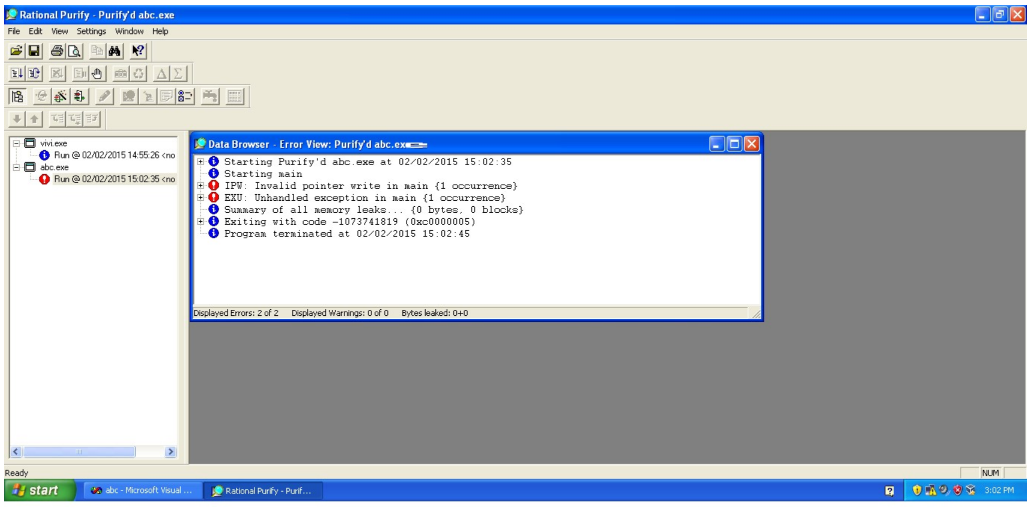
1. Open .exe of C program in rational purify and choose 'run your program from purify' option
2. Memory leak report is generated.
3. Analyze the memory leak report and corresponding error codes in detail.

**Sample cpp code:**

```
class student
{
int rno;
char *name;
public:
 student(int n, char *na)
{ rno=n;
name=new char[strlen(na)+1];
name=na;
}
void print()
{
cout<< "roll no"<<rno<<endl;
cout<< "Name"<<name<<endl;
}

~student()
{
delete name;
};

void main()
```

```
{student s1(1, "kavitha");
student s2(2,"abcdef");
s1.print();
s2.print();
}
```

## Sample Output


Rational Purify - Purify'd abc.exe
File Edit View Settings Window Help
vivi.exe
Run @ 02/02/2015 14:55:26 <no
abc.exe
Run @ 02/02/2015 15:02:35 <no
Data Browser - Error View: Purify'd abc.exe
Ready
NUM
start
abc - Microsoft Visual ...
Rational Purify - Purif...
3:02 PM

```
Starting Purify'd abc.exe at 02/02/2015 15:02:35
Starting main
IPW: Invalid pointer write in main {1 occurrence}
EXU: Unhandled exception in main {1 occurrence}
Summary of all memory leaks... {0 bytes, 0 blocks}
Exiting with code -1073741819 (0xc0000005)
Program terminated at 02/02/2015 15:02:45
```

Displayed Errors: 2 of 2    Displayed Warnings: 0 of 0    Bytes leaked: 0+0

## Exercise No. 7 Code Coverage Testing with Rational Pure Coverage:

**OBJECTIVE: Understanding code coverage testing.**

Pure Coverage monitors your program as it runs and reports exactly what parts of your program have and have not been exercised. You can use Pure Coverage on whatever scale you want: you can collect data from informal tests of specific program functionalities, or combine data from all the runs controlled by your test suite. Used with Purify, Pure Coverage can tell you what parts of your program have not been checked for memory errors and leaks, which help you significantly, improve the quality of the code you ship to customers.

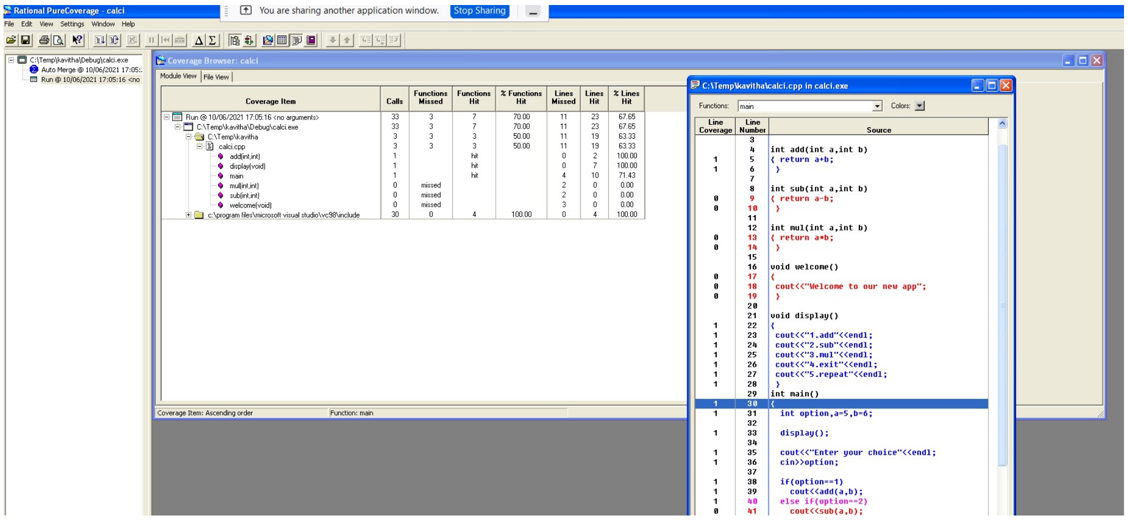
**Pre-Requisite: Writing code in C / C++ / Java**

**Pre-Lab: Online CD purchasing system**

**Procedure:**

1. Compile and link your program under Pure Coverage to Instrument the program with coverage monitoring instructions.
2. Run the program to collect coverage data.
3. Display and analyze the coverage data to determine which parts of your program were not tested.
4. Improve coverage for the program.
5. Modify your make files to use Pure Coverage throughout your development cycle.

**SAMPLE INPUT/OUTPUT**


Rational PureCoverage - calci
File Edit View Settings Window Help
You are sharing another application window.
Stop Sharing
C:\Temp\kavitha\Debug\calci.exe
Auto Merge @ 10/06/2021 17:05:
Run @ 10/06/2021 17:05:16 <no
Coverage Browser: calci
Module View
File View
Coverage Item: Ascending order
Function: main
C:\Temp\kavitha\calci.cpp in calci.exe
Functions: main
Colors:

| Coverage Item | Calls | Functions Missed | Functions Hit | % Functions Hit | Lines Missed | Lines Hit | % Lines Hit |
|---|---|---|---|---|---|---|---|
| Run @ 10/06/2021 17:05:16 <no arguments> | 33 | 3 | 7 | 70.00 | 11 | 23 | 67.65 |
| C:\Temp\kavitha\Debug\calci.exe | 33 | 3 | 7 | 70.00 | 11 | 23 | 67.65 |
| C:\Temp\kavitha | 3 | 3 | 3 | 50.00 | 11 | 19 | 63.33 |
| calci.cpp | 3 | 3 | 3 | 50.00 | 11 | 19 | 63.33 |
| add(int,int) | 1 | | hit | | 0 | 2 | 100.00 |
| display(void) | 1 | | hit | | 0 | 7 | 100.00 |
| main | 1 | | hit | | 4 | 10 | 71.43 |
| mul(int,int) | 0 | missed | | | 2 | 0 | 0.00 |
| sub(int,int) | 0 | missed | | | 2 | 0 | 0.00 |
| welcome(void) | 0 | missed | | | 3 | 0 | 0.00 |
| c:\program files\microsoft visual studio\vc98\include | 30 | 0 | 4 | 100.00 | 0 | 4 | 100.00 |

| Line Coverage | Line Number | Source |
|---|---|---|
| | 3 | |
| | 4 | `int add(int a,int b)` |
| 1 | 5 | `{ return a+b;` |
| 1 | 6 | `}` |
| | 7 | |
| | 8 | `int sub(int a,int b)` |
| 0 | 9 | `{ return a-b;` |
| 0 | 10 | `}` |
| | 11 | |
| | 12 | `int mul(int a,int b)` |
| 0 | 13 | `{ return a*b;` |
| 0 | 14 | `}` |
| | 15 | |
| | 16 | `void welcome()` |
| 0 | 17 | `{` |
| 0 | 18 | `cout<<"Welcome to our new app";` |
| 0 | 19 | `}` |
| | 20 | |
| | 21 | `void display()` |
| 1 | 22 | `{` |
| 1 | 23 | `cout<<"1.add"<<endl;` |
| 1 | 24 | `cout<<"2.sub"<<endl;` |
| 1 | 25 | `cout<<"3.mul"<<endl;` |
| 1 | 26 | `cout<<"4.exit"<<endl;` |
| 1 | 27 | `cout<<"5.repeat"<<endl;` |
| 1 | 28 | `}` |
| | 29 | `int main()` |
| 1 | 30 | `{` |
| 1 | 31 | `int option,a=5,b=6;` |
| | 32 | |
| 1 | 33 | `display();` |
| | 34 | |
| 1 | 35 | `cout<<"Enter your choice"<<endl;` |
| 1 | 36 | `cin>>option;` |
| | 37 | |
| 1 | 38 | `if(option==1)` |
| 1 | 39 | `cout<<add(a,b);` |
| 1 | 40 | `else if(option==2)` |
| 0 | 41 | `cout<<sub(a,b);` |

## Exercise No.8: Management of Test Plan, Test suite for Manual Testing using Rational Test Manager

**OBJECTIVE: Preparing Test Plans with Test cases, Test data**

**PRE-REQUISITE: Use cases and its equivalent test cases**

**Pre-Lab: Online CD purchasing system**

**PROCEDURE:**

### Managing test Planning

Rational Test Manager allows you to collect and organize your testing information, and locate it in a central place that all members of the team can access.

In this lab, you set up test manager with planning information that helps you run the right tests at the right time. You register an excel spreadsheet that contains use case flow details as a test input source. You also edit existing configurations and add a new one.

**Steps:**

- Connect to a Rational Project
- Navigate the Test manager main window
- Register an MS Excel spreadsheet as a test input source
- Manage configurations and configuration attributes

### Start Test manager and Connect to a project

Steps:

1. Start Rational Test Manager
2. Type admin as the User Name, if not already there, and leave the password box blank
3. Since you connected to the TT Project in the last lab, TT project appears in the project box, and the its path C:\TS Training\TT Project.srp appears in the path box.
4. Click OK.

### Explore the Test Manager Main Window

Steps:

1. In the Test Asset Workspace, select the planning Lab.
2. Expand the Test Plan folder to see the test plans in the project.

3. Expand the Iterations folder to see iterations available in your project.
4. Explore the Execution, Results, and Analysis tabs in the test Asset workspace.

**Register a Test Input Source**

In this section of the lab, you register an Excel spreadsheet as a test input source. The spreadsheet in the example outlines scenarios for the place an order use case. Later in the lab, you create test cases based on the place an order use case, and you associate values in the spreadsheet as test inputs to those test cases.

You can view the spreadsheet **C:\TS Training\TCS process sales.xls**

Steps:

1. From the Test Manager main menu, click Tools > Manage > Test Input Types
2. Select Microsoft Excel, then click Edit.
3. Select the sources tab, then click Insert.
4. On the General tab, in the Name field, type Place an order use case.
5. Select the Connection Data tab, then click Browse. Browse to C:\TS Training\UCSEC-TCS Process sales.xls click open
6. Click set configuration
7. You are prompted to save the test input source before configuring it, click yes.
8. In the set configuration dialog box, enter the following values that reflect the configuration of the sample spreadsheet.
9. Click OK
10. Click OK in the New Test input source properties dialog.
11. Click OK in the Test Input Type properties dialog box.
12. Click Close in the manager test input type dialog box.

**Edit configuration attributes**

1. From the main menu, select Tools > Manage > configuration Attributes
2. Click New in the Manage Configuration Attributes dialog box.
3. Type Database Type in the Name box of the New Configuration attribute dialog box.
4. In the source of values box, click List.
5. In the List Values box, type MS Access, then click Add.
6. Add SQL Server and oracle to the list.
7. Click OK.
8. Add the values 48, 64, and 128 to the memory size attribute.
9. Click OK.
10. Click Close.

**Create a new configuration**

1. From the main menu, select Tools > Manage > Configurations
2. Click New.
3. Type Windows 98 in the Name box.
4. Click the attributes tab. Click in the empty box to the right of the operating system attribute. A Drop-down list appear. Select =
5. Click in the empty box to the right of the =. Select Windows from the drop-down list.
6. Make the OS version attribute = 98
7. Make the memory size >=64
8. Click OK.
9. Click Close.

**Building a Test Manager Test Plan**

In this lab, you use Test Manager to translate the information from test planning activities and artifacts into a TeestManager test plan.

In this process, you identify test cases, which are the items or features that you want to verify through testing. You identify when in the development cycle you need test case verification. You also identify the different system configurations the test case must be validated on.

**Objectives**

- Add a new test plan.
- Associate an external document with a test plan.
- Add test case folders to a test plan.
- Create and configure test cases.

**Build a Test Plan and Associate an External Document**

**Steps**

1. In the Test Asset Workspace, select the Planning tab.
2. From the menu, select **File> New test Plan.**
3. IN the New Test Plan box, select the **General** tab. Type **Classics Online Name** box.
4. Type Classics Online Application in the Description box.
5. Leave the **Owner** box as is.
6. Click **OK**
7. Select the **Classics Online** test plan in the test Plan Window.
8. Right-click on the **Classics Online** test plan in the Test Plan window, and select **Properties** from the shortcut menu.
9. Select the External Documents tab.
   Click **Add.**

Browse for the test plan document: C:\TS Training\Classics Online TestPlan.doc.

10. Click **Open.**
11. Click **OK.**

**Add a Test Case Folder**

Steps

1. In Test Manager, open the Classics Online Test Plan if it's not already open.
2. In the Test Plan window, select the Classics Online test plan. Right-click and Select **Insert Test Case Folder** from the shortcut menu.
3. Type **Orders** in the **Name** box.
4. Type **Classics order test cases** in the Description.
5. Click OK.
6. Add a second test case folder to the Classics Online Test plan. Name of the folder "Administration", and include "classics administrative functions" as the description.
7. Click OK

**Add Test Cases to a Test Case Folder**

Steps

1. Under the Classics Online test plan in the test plan window, select the Orders test case folder.
   Right-click, and select Insert Test Case from the Shortcut Menu.
2. In the New Test Case dialog, on the General tab, type Place an order in the Name box.
3. Type Place a single-item order in the Description.
4. Click OK.
5. Insert more test cases in the Orders test case folder: Install Classics, Login, Place order-invalid credit card, Place order-invalid quantity.

**Associate Test Inputs with test Cases**

Steps

1. In the test Plan Window, select the Place an Order test case. Right-click, and select Associate test Input from the shortcut menu.
2. In the Test Input Selection box, click to expand the Test Input Sources tree.
3. Expand the Excel source place an Order use case, and locate the input Scenario 1 –The Sales Clerk can complete a single item order. Select the input.
4. Click OK.
5. In the Test Plan Window, right-click the Place an order test case, and select Properties from the shortcut menu.
6. Select the Test Inputs tab.

7. Associate test cases Place order-Invalid credit card, Place order-invalid quantity with the Scenario 1 test input.
8. Click OK
9. Click View>Test Inputs.
10. Expand the Excel source, then expand Scenario 1 test input.
11. Close the Test Inputs View.

**Associate Iterations with New Test Cases**

Steps

1. In the Test Plan windows, select the Administration test case folder. Right-click and select Insert Test Case from the shortcut menu.
2. In the New Test Case box, on the General tab, type View customers in the Name box.
3. Select the Iterations-Configurations tab.
4. Click Select to the of the Iterations box.
5. Select Construction 1 from the Available list box.
   Click > to move it to the Selected list.
6. Associate the test case with the Transition 1 iteration also.
7. Click OK
8. Click OK to close the New Test Case Box.

**Associate Iterations with Existing Test Cases**

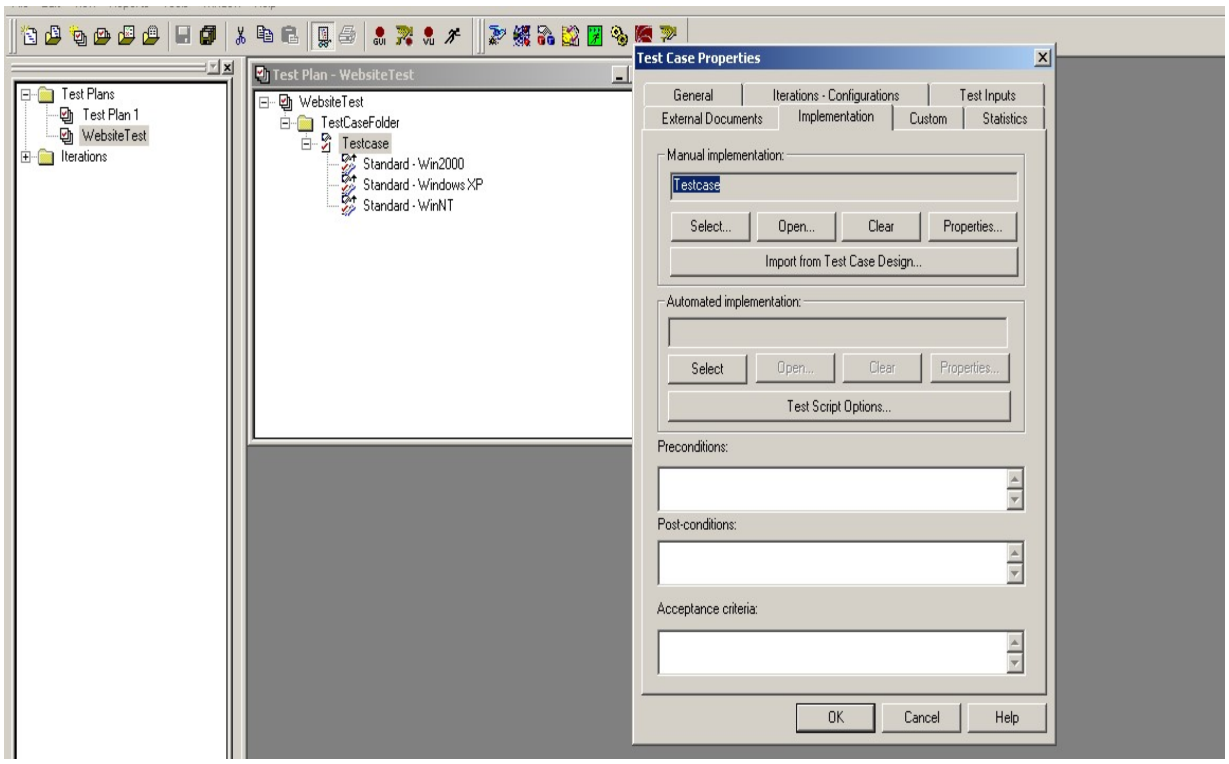
Steps

1. Select the Place an order test case in the Test Plan window. Right-click and select Associate Iteration from the Shortcut menu.
2. Select Construction 1 from the list, Hold down Ctrl Key and select Construction 3 from the list.
3. Click OK.
4. Click Yes.
5. To verify the configurations, view the Iterations-Configurations tab in Test Case Properties.

**Create Configured Test Cases**

1. Select the Place an Order test case in the Test Plan Window.
2. Right-click and select Associate Configuration from the Shortcut menu.
3. Choose Windows 98, Standard-WinNT and Standard-Win2000.
4. Click OK.

**SAMPLE INPUT/OUTPUT**


Test Plans
Test Plan 1
WebsiteTest
Iterations

Test Plan - WebsiteTest
WebsiteTest
TestCaseFolder
Testcase
Standard - Win2000
Standard - Windows XP
Standard - WinNT

Test Case Properties

General | Iterations - Configurations | Test Inputs
External Documents | Implementation | Custom | Statistics

Manual implementation:
Testcase
Select... Open... Clear Properties...
Import from Test Case Design...

Automated implementation:
Select Open... Clear Properties...
Test Script Options...

Preconditions:

Post-conditions:

Acceptance criteria:

OK Cancel Help

**Exercise No.9 Developing GUI Test Script for automated Testing**
**OBJECTIVE:** Developing Test script
**Pre-Requisite:** Test plan for minimum three Test cases.
**Pre-Lab: Online CD purchasing system**

**Creating a Manual Test Script**
Rational Test Manager is closely integrated with Rational Manual Test, a tool that allows
you to create and run test scripts for tests that cannot be automated. You can use Rational Manual Test to outline the steps to execute manually, and the verification points to confirm along the way. After you create a manual test script, you can associate it with a test case, you can execute the test case from Test Manager and its results are included in Test Manger test logs. The results can be included in Reports.

**Steps**

- Create a manual test script in Rational Manual Test.
- Implement a test case with a manual Script in Rational Test Manager.

**Create a Manual Script**
Steps

1. Open Test Manager.
2. On the Main Toolbar, select the Start Rational Manual Test icon.
3. In Rational Manual Test, select File> New.
   Example script for "Classic online Log on" is given below:
4. Put the cursor in the box that reads, "Enter the first step of this manual script" Delete the text, and type: Select user name and type your password
5. Click in the box under the Note Heading.
6. Type Use the Latest build in the note box and click OK.
7. Position the cursor back in the description box of the line, and press Enter.
8. In the description column of the second row, type "Did it login for correct username and correct password" and vice versa.
9. Click Enter.
10. Add more steps for other scenarios for authentication of correct login.
    Click Next to proceed next foot point
11. Click File > Save. Name the script log on Classics. Click OK.
12. Select File > Exit to exit Rational Manual Test.

**Implement a Test Case with a Manual Script.**
Steps

1. Open the Classics Online test plan, orders folder, and select the "Classic online Log on" test case.
2. Go to Test Case Properties, and click the Implementation tab.
3. Click Select under Manual implementation.
4. From the list of manual scripts, select "Classic online Log on" and click OK.

5. Click OK.

**Implement a Manual Script from a test Case Design**

Steps

1. Go to the Implementation tab of Test Case Properties for the Place an Order parent test case.
2. In the Manual Implementation area, click Import from Test Case Design.
3. Verify the name of the test case, the click OK.
4. Click Open.
5. Exit Rational Manual test.
6. Click OK to close the test Case Properties box.

**Recording a GUI Test Script**

In this lab, you get your first chance to record and play back a basic test script using Robot and Build A of the classics online application. If the test script you record in this lab does not work perfectly, don't be concerned. In the next few labs, you learn more about structuring test scripts and debugging them when they fail.

**Objectives**

- Record a basic GUI test script in Rational Robot.
- View and play back the test script.
- View the play back results.

**Record the basic classic script**

**Steps**

1. Start Robot. Start > Programs > Rational Team Test > Rational Robot
2. Click File > New > Script
3. In the Name box, type Basic Classics. Ensure that GUI is selected for type.
4. Click OK.
5. With the cursor positioned on the blank line before End Sub, click Record > Insert a cursor.
6. Click the display GUI insert toolbar icon on the record toolbar.
7. Click the start Application icon on the GUI insert toolbar.
8. Browse to C:\Program Files\Rational\Rational Test\Sample Applications\Classics online\ classicsA.exe
9. Ensure that the radio button next to using settings from GUI play back options dialog box is checked.
10. Click OK
11. Select Admin from the Full Name list.
12. Leave the Password box blank.
13. Click OK
14. Scroll through the albums listed in the upper window by using the down and up arrows. Select Bahc's Violin Concertos.
15. Click the details tab and view the contents.

16. Click the album tab and view the contents.
17. Click the press here to order.
18. Type 1 in the Quantity box. Press Tab.
19. In the Card Type list, click Master Card. Press TAB.
20. Type 1122334455667788 in the card Number box. Press TAB.
21. Type 02/2003 in the Expiration Date box.
22. Click Place Order.
23. Click OK.
24. Click File > Exit
25. Click the Stop Recording icon on the GUI Record toolbar.

**View the script**

Steps

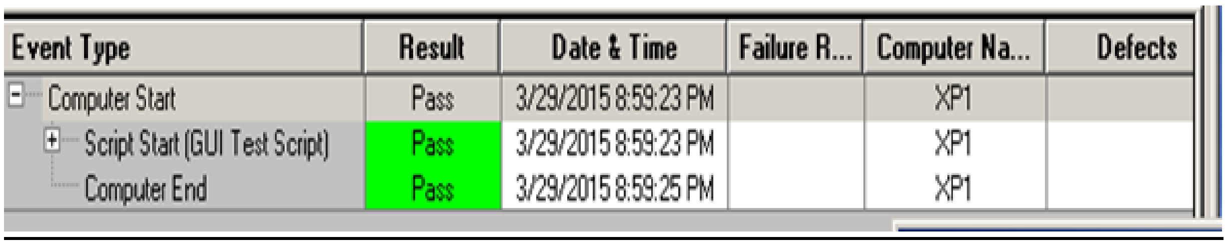
1. Maximize the robot window.
2. Maximize the main script window.
3. View the entire script by using the scroll arrows on the right side of the screen.
4. See if any of the commands make sense to you.

**Play back the script**

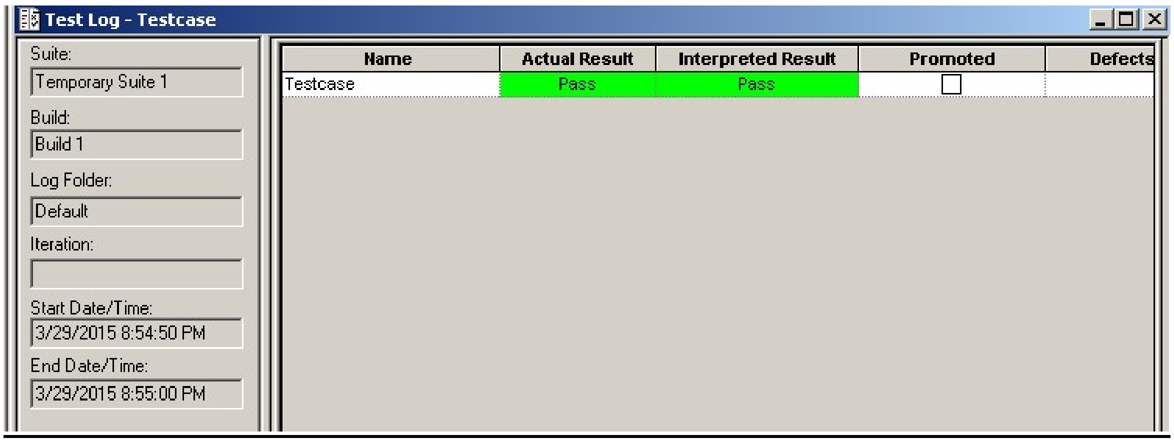
Steps:

1. From the main menu, select Tools > GUI playback options. Select the log tab.
2. Under Log management, Make sure that both output playback results to log and view log after playback are selected.
3. Click the Playback Script icon on the toolbar.
4. Select the basic classics script from the list.
5. Click OK
6. Click OK in the Specify Log Information Window to accept the defaults.
7. When the script completes, the Rational Test Manager test log appears. Can you tell if your script passed? Look in the result column.
8. Close Test Manger.
9. Close Robot.

**SAMPLE INPUT/OUTPUT**



| Event Type | Result | Date & Time | Failure R... | Computer Na... | Defects |
|---|---|---|---|---|---|
| ⊟ Computer Start | Pass | 3/29/2015 8:59:23 PM | | XP1 | |
| ⊞ Script Start (GUI Test Script) | Pass | 3/29/2015 8:59:23 PM | | XP1 | |
| Computer End | Pass | 3/29/2015 8:59:25 PM | | XP1 | |


Test Log - Testcase

Suite: Temporary Suite 1
Build: Build 1
Log Folder: Default
Iteration:
Start Date/Time: 3/29/2015 8:54:50 PM
End Date/Time: 3/29/2015 8:55:00 PM

| Name | Actual Result | Interpreted Result | Promoted | Defects |
|---|---|---|---|---|
| Testcase | Pass | Pass | ☐ | |

**Exercise No. 10: Object Insertions and adding alphanumeric verification Points**

**Objective: To understand verification points and implementing Object insertion.**
Sub Objectives

- Record logon, logoff you, and should be able to:
- Record logon, logoff, and initial window scripts in Classics A.
- Insert a window Existence verification point
- Insert Menu verification point

**Pre-Requisite: knowledge of verification Points**

**Pre-Lab: Online CD purchasing system**

**Procedure:**
**Inserting Window Existence and Menu Verification Points**

In this lab, you use Robot to record scripts on the Alpha build of the classics application.
Two of the scripts you create are the logon and logoff scripts. These scripts are small, reusable scripts, which can be played back any time you need to open or close the application. In the third script, you capture baseline test case data for the main window and its menu commands.

**Record Classic Logon Script**
Steps:

1. Minimize of close all other applications on your desktop.
2. Start Robot. Log in as admin. Be sure that the TT Project is selected.
3. Click File > New Test Script > GUI
4. In the name box, type classics logon.
5. In Description, type log on to the classics application.
6. For type, select GUI
7. Click OK.

8. From the Robot main menu, click Insert > Start application
9. In the Application Name box, browse to the classic A executable. The application can found in:
   C:\program files\Rational\Rational test\Sample Applications\classics online\classicsA.exe
10. Click OK.
11. Click on the Robot main window to return focus to robot.
12. Click Record > Insert at cursor.
13. Select admin from the Full Name list.
14. Leave the password box blank.
15. click OK.
16. Stop the recording by clicking on the stop recording icon on the GUI record box.
17. click File > Save to save the script

**Record the classics initial window script**

Steps:

1. In robot, click File > New > Script
2. Type Classics initial window in the name box. In the description box, type verify that the correct window opens after logon
3. Click Ok.
4. Click Record > Insert at cursor

**Insert a Window Existence Verification Point**
Steps:

1. Click Display GUI Insert Toolbar on the GUI Record toolbar.
2. Click Write to log on the GUI Insert toolbar.
3. Type verify the initial state of the classics application as the message text.
4. Type the next two VPs will verify initial state of classics as the description.
5. Under Result, Select None, and click.
6. Click Display GUI insert Toolbar on the GUI insert toolbar on the GUI Record.
7. Click the window Existence verification point button on the GUI insert toolbar.
8. Type window Existence 1 as the name.
9. Click OK.
10. Click Select on the window existence dialog box.
11. Click and drag the object finder over the Classics Online window. Release the mouse button once the object finder is over the classics window.
12. Click OK on the select object dialog box.
13. Click OK on the window existence dialog box.

**Insert a menu verification point**
Steps:

1. Click Display GUI insert toolbar on the GUI Record toolbar.
2. Click the menu verification point button on the GUI insert toolbar.
3. Type Menu 1 as the verification point name, and click OK,
4. Use the object finder tool to select the classics online menu.
5. click ok.
6. Verify that test menu states and test shortcut keys are checked.
7. In the Select range to test data gird, select the cells you want to test by clicking and dragging across the column headings(in gray).
   The default for the menu test case is all selected cells.
8. Verify that By title is selected in the Identification method box.
9. Verify that by location is selected in the second Identification method box.
10. Click OK.
11. Click the stop button on the GUI Record toolbar.
12. Click File > Save to save the script.

**Record the Classics Logoff Script**

Steps:

1. In Robot, start a new script called classics logoff.
2. Click Record > Insert at cursor.
3. In classics, Select file > Exit to log off the application
4. Stop recording, and save the script.

**Playing Back a Test Script and Viewing the Results**

After you record a test script, play it back on the same build of the system-under-test to ensure that it performs properly. A test script should playback without error on the same software build on which it was recorded.

In this lab, you play back the same test scripts you recorded in lab 4.3, and use the rational test manager test log to examine the results. Robot automatically compiles the test script when you play it back for the first time, or after it has been modified.

Before playback, always verify and reset the playback environment for testing.

Objectives

- Review and set Robot error recovery, log, and playback options.
- Play the previously created test scripts.
- Review and verify playback results

**Review and Set error Recovery Options**

Steps

1. From the Robot menu, click Tools > GUI Playback options
2. Click the error recovery tab and set as shown below.

**Review and set log options**

Steps:
1. Click Log Tab.
2. Set the log options as shown below.

**Review and set playback options**

Steps:

1. Click the Playback Tab.
2. Set the playback options as shown below.
3. Click OK.

**Play Back the logon Script in Robot**

Steps:

1. From the Robot menu, select File > Playback, or click the Playback Script icon.
2. Select Classics logon from the list of Available scripts.
3. Click OK
4. Click OK to accept the log information window defaults.
5. Verify that the scripts passed by looking at the result column.
6. From the Test manager menu click File > Edit.

**Play Back the initial window Script in Robot and verify Results**

Steps:

1. From the Robot toolbar, click the playback script icon.
2. Select classic initial window from the list of Available scripts, and click OK.
3. Click OK to accept the default information in the log Information window.
4. For each verification point, a dialog box appears, prompting you to acknowledge pass / fail playback results.
   Click OK in each dialog box.
5. The test manager test log appears.
   Review the pass /fail results.
6. Exit manager.

**Play back the logoff Script in Robot**

Steps:

1. Repeat the previous steps to play back the classics logoff script.
2. Repeat the previous steps to verify the play back results.

**Inserting Object Properties and Alphanumeric Verification Points.**

Robot has a number of verification point types that you can apply to verify test requirements in the application-under-test. Often, more than one verification point type may seem appropriate for a particular test requirement. In this module's lab, you learn how to distinguish between the different types of verification points and how to apply most appropriate one under given conditions.

In this lab, you develop a new script and verify tow requirements of Classics Online's main menu using the Object Properties and Alphanumeric verification points.

**Objectives**

- Record a new robot script.
- Insert an Object Properties verification point.
- Insert an Alphanumeric verification point.
- Play back and verify the script.

**Record the classics Order window scripts**

Steps:

1. If necessary, start Robot and select the Project.
2. If necessary, start the ClassicA application and logon as Admin with no password.
3. Close or minimize all other windows applications.
4. In Robot, start recording by clicking the Record GUI Script button on the toolbar.
5. Enter Classics Order Window as the script name.
6. Click OK.

**Insert an Object Properties Verification Point.**

Steps:

1. In the ClassicA application, click the Details tab.
2. In Robot, click the Display GUI Insert Toolbar button on the GUI Record Toolbar.
3. Click Object Properties verification point button on the GUI insert toolbar.
4. Type Object Properties 1 as the name.
5. Click OK
6. Use the Object Finder to select the Detail tab.

7. Click OK.
8. Click Edit List.
9. Double-click Height, Left, Top, Width and Focus to deselect the properties from the list.
10. Click Save Ad Default.
11. Click OK.
12. Click OK.

**Insert an Alphanumeric Verification Point.**

Steps:
1. In the Classics Application, click the press here to order button.
2. In Robot, click Display GUI Insert Toolbar on the GUI Record Toolbar.
3. Click the Alphanumeric verification point button on the GUI insert Toolbar.
4. Type Alphanumeric1 as the name, then click OK.
5. Select a Verification method of Case Insensitive from the list, then click OK.
6. Use the Object finder tool to select the Name edit box. Then click OK.
7. Click Yes in the message box.
8. In the Classics Window, Click Cancel.
9. Stop Recording.

**Play Back and Verify the Script.**

1. In the Classics application, select Album from the main window.
2. From the Robot menu, click Tools> GUI Playback Options.
3. Click the Log tab, then check Use default log information.
4. Click the Playback uncheck Acknowledge results, and the click OK.
5. Play back the script.
6. In the Test Manager log, verify that all verification points completed successfully.
7. Exit TestManger.

**Verification Point Type.**
**Alphanumeric**
Tests for case-sensitive or case-insensitive text, an exact number, or a number within a range; verifies that a field is blank, or allows you to run your own custom test on an alphanumeric value.
**Clipboard**
Captures and compares alphanumeric data that has been copied, using copy and paste commands, to the Clipboard, For example, this would be useful for data from the spreadsheets or from data in controls that cannot be captured using any other verification point.

**File Comparison**
Compares the contents of two files.
**File existence**
Checks for the existence of a specified file.

**Menu**
Captures and compares the text, accelerator keys, and state of all of the menus in the AUT. Captures as many as five levels of Sub-Menus.

**Module Existence**
Checks for a specified software module in memory.

**Object Data**
Captures and compares the data in menus, list type objects, 32-bit common controls, data Windows, Table Windows, Visual Basic Objects, OCXs and VBXs.

**Object Properties**
Captures and compares the properties of standard Window objects, data Windows, Table Windows, Visual Basic Objects, OCXs and VBXs.

**Region Image**
Captures a region of screen. When you play back the recorded script, the verification point tests for differences between the captured region and the playback region.

**Website Compare**
Web site Compare is used to compare two Web sites and view the basic statistics for each site.

**Window Existence**
Verifies that the specified window is in memory before continuing with playback.

**Window Image**
Captures a window for bitmap comparison. During playback, the verification point compares the same window in the AUT to the window captured during recording.

**Inserting Object Properties and Object Data Verification Points.**

Steps:

- Insert an Object Properties verification point into a script.
- Edit the object properties list.
- Insert an Object Data Verification point into a script.
- Distinguish between an Object Properties and an Object Data verification point

**Record a script and Object Properties Verification Point.**

Steps:

1. Start Robot, if necessary.
2. Start ClassicsA (Logon as Admin, with no password)
3. Close or minimize all other applications.

4. In robot, begin recording a script called Classics Place An Order. Click OK.
5. In Classics, click press here to order.
6. In Robot, Click Display GUI Insert Toolbar on the GUI Record Toolbar.
7. Click the Object Properties Verification point button on the GUI insert toolbar.
8. Type Object Properties 1 in the Name box on the Verification Point Name Window. Click OK.
9. Click Browse in the Select Object box.
10. Check Invert Object on the Object list box.
11. Select Window Name=frmorder from the list. Click OK.
12. Click OK on the Select Object dialog box.

**Edit the Object Properties.**
Steps:
1. In the Object Properties Verification Point box, select the Window Object.
2. Click Edit List.
3. Double-Click on hDc to deselect this property from the list, then click ok.
4. In the Object Properties Verification Point box, Scroll through the object on the left until you reach the first Edit Box Object. Select it.
5. Click Edit list.
6. If necessary, double-click Height, Left, Top, Width and Focus to deselect the properties.
7. Click Save as Default and Apply to all like objects.
8. Click OK.
9. Select the first Pushbutton object and repeat steps 5 through 8.
10. Select a Label object and repeat steps 5 through 8.
11. Select one of the Group box objects and repeat steps 5 through 8.
12. In the Object Properties verification Point box, Click OK.

**Define an Object Data Verification Point.**
Steps:
1. In the Make An Order dialog box of the Classics application, click in the Quantity box.
2. Press Home, then Shift+End, then Delete.
3. In the Quantity box, type 2.
4. Click in the Card Number box.
5. Press Home, then Shift+End, then Delete.
6. Type a 16-digit credit card number.
7. Click in the Expiration Date box.
8. Clear the box of any old data.
9. Type the Expiration data as 9/2003.
10. In Robot, Click Display GUI Insert Toolbar on the GUI insert toolbar.
11. Click the Object data verification point button on the GUI insert toolbar
12. Type Object Data1 in the Name box on the Verification Point Name box.
13. Click OK.

14. Drag the Object Finder tool over the Card Type box, and release the left mouse button.
15. Click OK.
16. Verify that contents is selected in the Data test box.
17. Click OK.
18. In the Classics application, click Place Order.
19. Click OK to the Order Confirmation message.
20. Stop Recording.
21. Exit the ClassicsA application.

**Play back and Verify the Script.**

Steps:

1. Restart the ClassicA application and logon as Admin.
2. In Robot, play back the Classics Place An Order script.
3. In Test Manager Log, verify that all verification points completed successfully.
4. Exit Test Manager.

**Edit a Script with SQABasic.**

You can modify and enhance SQABasic script by editing within Robot. You can apply the same simple programming techniques you use in Visual Basic to make scripts to scripts more sophisticated and powerful.

For example, you can make scripts more fault-tolerant, and you can check whether or not an application is running prior to starting the application. You can accomplish this by testing for the existence of the application's main window and conditionally branching based on the result. If the main window does not yet exist, take a path that starts the application. If the main window does exist, take a branch that bypasses the code to start the application.

**Objectives**

- Add comments to a script.
- Add a script command to test for Window Existence.
- Edit, compile, and play back the script.

**Modify the Classics Logon Script**

Steps:

1. If necessary, start Robot.
2. Log on to the Classics application.
3. In Robot, Click>Open Script.
4. Select Classics Logon as the script name, and click OK.
5. Insert a blank line in the script after the header line. "Script Name: Classics".
6. Click Insert>Comment on the Robot menu.
7. In the Comment Window, type Add code here to check if Classics is already running, and click OK.

**Test for Window Existence.**

Steps:

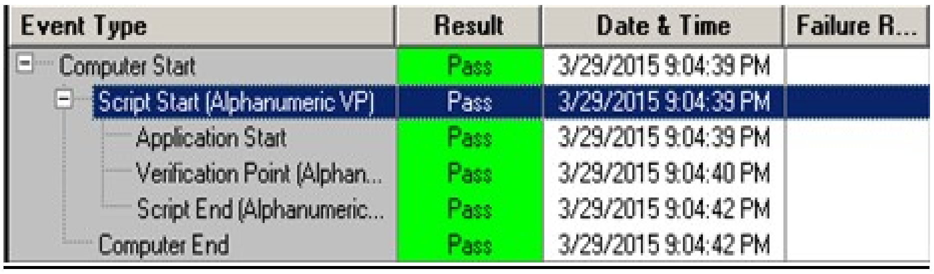
1. Click Insert>Verification Point > Window Existence.
2. Type Window Existence 1 as the name, then click OK.
3. Click Select.
4. In the Select Object box, use the object finder to select the main window of the Classics application.
5. Point and click anywhere in the Classics main window.
6. Click OK in the Select Object window.
7. In the Windows Existence box, uncheck Test for Window Status (If checked), and click OK.

**Edit, Compile, and Play Back the Script.**

Steps:

1. Maximize Robot and the script window.
2. Add the following modifications (Shown in bold) to your script.
3. Click File> Save
4. Click File> Compile
5. Make sure Classics is not running, and then play back Classics Logon.
6. Verify that Classics Started, and that the Script Start Log Even passed.
7. With Classics still running, play back Classics Logon again.
8. Verify that Classics did not start again because it was already running, and that the script Log Event still passed.
9. Exit Test Manager.

**SAMPLE INPUT/OUTPUT**



| Event Type | Result | Date & Time | Failure R... |
|---|---|---|---|
| Computer Start | Pass | 3/29/2015 9:04:39 PM | |
| Script Start (Alphanumeric VP) | Pass | 3/29/2015 9:04:39 PM | |
| Application Start | Pass | 3/29/2015 9:04:39 PM | |
| Verification Point (Alphan... | Pass | 3/29/2015 9:04:40 PM | |
| Script End (Alphanumeric... | Pass | 3/29/2015 9:04:42 PM | |
| Computer End | Pass | 3/29/2015 9:04:42 PM | |

## Exercise No.11 a Testing with an External Data Source

**Objectives**

- View and modify a data source.
- Record and then modify a script to read the data file.
- Play back the script and update the baseline.

**Pre-Requisite: text, data, csv, mdb, excel, doc files**

**Pre-Lab: Online CD purchasing system**

**Concept**

Volume testing validates that the application behaves properly when the volume of test data increases to production level. It also ensures that variations in the production data are handled appropriately. In this lab, you create a volume test for the Classics online application.

Robot can perform volume testing in a number of ways. One method involves modifying the script to enter a large amount of test data by randomizing input variables and creating a loop. Another method is tread data from an external source, such as an ASCII file, or a delimited file from an existing production database.

**Procedure**

The basic steps involved are to:

1) Record a script using baseline data, then
2) Modify that script to use variables that refer to fields in a data file.

Use the script you created in a previous lab. To change the volume, you increase or decrease the amount of data in the file. The script executes once for each record in the data file, unless you specify otherwise.

**View and Modify the Data Source**

Steps

1. In Windows Notepad, open the file C:\ TS Training\Classics orders.dat.
2. View the Contents of the file. Notice the format of the records.
3. Add another record by creating a sixth line, following the format of the other lines. Use any information for the credit card number and date, as long as they following the format as the other records.
4. Click File>Save
5. Click File >Exit

**Create the Classics Place Multiple Orders Script.**

Steps

1. If necessary, play back the Classics Logon script to open the application

2. In Robot, open the classics place an order script.
3. Save it as classics place multiple orders.
4. Make any necessary changes to the script header information to reflect the new script.

**Modify the script to read from the Data file**

Steps

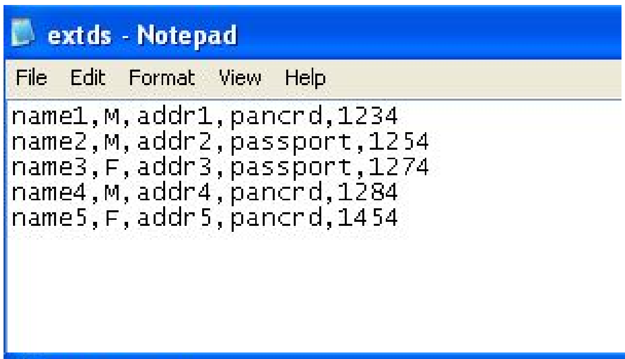
1. In the classics place multiple orders script, place your cursor below the dim result as integer statement.
2. Define the new variables for the data in the file. The variables in the Dim command should be CardNum, ExpDate.
3. Click the cursor under the script header information, as shown below.
4. Type in the commands to open the data file and loop through the script until the end of the data(EOF).
5. Type in the input command, listing all the variables you defined at the top of the script.
6. Replace the hard-coded data in the input keys commands with the variables you have just defined.
7. Comment out the object properties Verifiacation point command.
8. Type the loop command above the end sub statement.
9. Type the close command under the loop command and above the End sub statement.
10. Click File > Compile.
11. Make sure classics is running, and then play back Classics place multiple orders
12. After Reviewing the results in the Test log exit Test manager.

**Explanation of script commands**

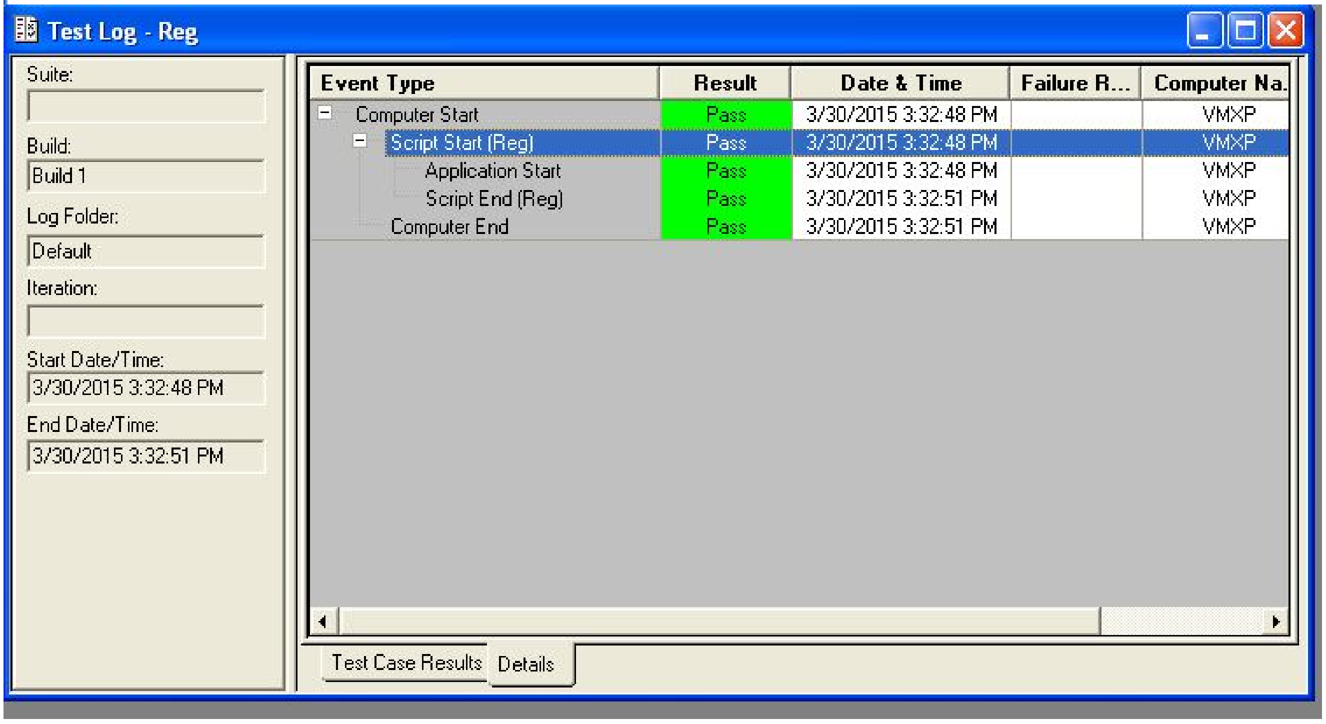
Commands

- Dim CardNum, Expdate as variant
- Open “ C:\TS Training\ Classics_Order.dat” for input as #1
- Do while Not EOF(1)
- Input #1, CardNum, ExpDate
- Input Keys &CardNum
- Input Keys &ExpDate
- Loop
- Close #1

## SAMPLE INPUT/OUTPUT



```
name1,M,addr1,pancrd,1234
name2,M,addr2,passport,1254
name3,F,addr3,passport,1274
name4,M,addr4,pancrd,1284
name5,F,addr5,pancrd,1454
```



Suite:

Build: Build 1

Log Folder: Default

Iteration:

Start Date/Time: 3/30/2015 3:32:48 PM

End Date/Time: 3/30/2015 3:32:51 PM

| Event Type | Result | Date & Time | Failure R... | Computer Na. |
|---|---|---|---|---|
| Computer Start | Pass | 3/30/2015 3:32:48 PM | | VMXP |
| Script Start (Reg) | Pass | 3/30/2015 3:32:48 PM | | VMXP |
| Application Start | Pass | 3/30/2015 3:32:48 PM | | VMXP |
| Script End (Reg) | Pass | 3/30/2015 3:32:51 PM | | VMXP |
| Computer End | Pass | 3/30/2015 3:32:51 PM | | VMXP |

## Exercise No. 11b Creating internal data sets and Using Data pools

A data pool is a test dataset that supplies data variables in a test script during playback. Using data pools allows you to run multiple iterations of a script using different data each time.

In this lab, you create a data pool by defining columns and automatically generating data. Then you modify the classics place an order script to replace some of the literal values you supplied while recording with variable data from the data pool.

**Objectives**

- Create a Data pool
- Edit a script to include data pool commands
- Playback a script drawing data from a data pool

**Pre-requisite**
**Sample data model with proper data dictionary of given sofware system**

**Pre-Lab: Online CD purchasing system**

**Procedure**
**Create data pool and generate data**

Steps:

1. Start Test manager and connect to the TT project.
2. Click Tools > Manage > Data pools
3. Click New
4. In the Name box, type order CD
5. Click OK
6. Click Yes to acknowledge that you want to define the data pool fields now
7. Click Insert After to add a data pool column to the data pool
8. Type quantity as the name of the data pool column
9. Leave the default data type integers signed
10. Scroll to the minimum box and insert 1. Insert 50 in the Maximum field.
11. Click Insert after to add another data pool column. Name it expdate.
12. Click in the type (Data type) box for the new column and select Date – MM-DD-YY from the drop-down list.
13. Click in the sequence box, and select Random from the drop-down list.

14. In the lower left corner of the window, enter 200 in the number of records to generate box.
15. Click Generate Data.
16. Click yes to view a sample record
17. Click Cancel
18. Click Close
19. Click Close.

**Add Data pool commands to script**

In this section, you edit a script to pull out data from the order CD data pool you created. The edited script will loop 10 times, each time drawing different generated data into the quantity and card expiration date fields of the classics online application order form.

Steps:

1. Start Robot.
2. Open the classics place an order script.
3. Save the script as classics place an order data pool.
4. Place the cursor on the top line of the classics place an order data pool script, above the sub main command. Type : '$include "sqautil.sbh".
5. Add the following modifications (shown in bold) to your script.

| | |
|---|---|
| Declare variables→ | **Dim x as integer 'Variable for loop**<br>**Dim dp As Long 'Variable for data pool**<br>**Dim quantity, expdate As string ' Variables for fields** |
| Open the data pool→ | **dp=SQADatapoolOpen("MyData")** |
| Start of loop → | **For x = 1 to 10** |
| Fetch a row from the Data pool → | **Call SQADatapoolFetch ( dp)** |
| Retrieve the a value from the fetched now → | **'Quantity data**<br>**Call SQADatapoolValue (dp,2,expdate**) |
| Loop → | **Next x** |
| Close the data pool → | **call SQADatapoolClose (dp)** |

**Play Back the Script**

Steps:

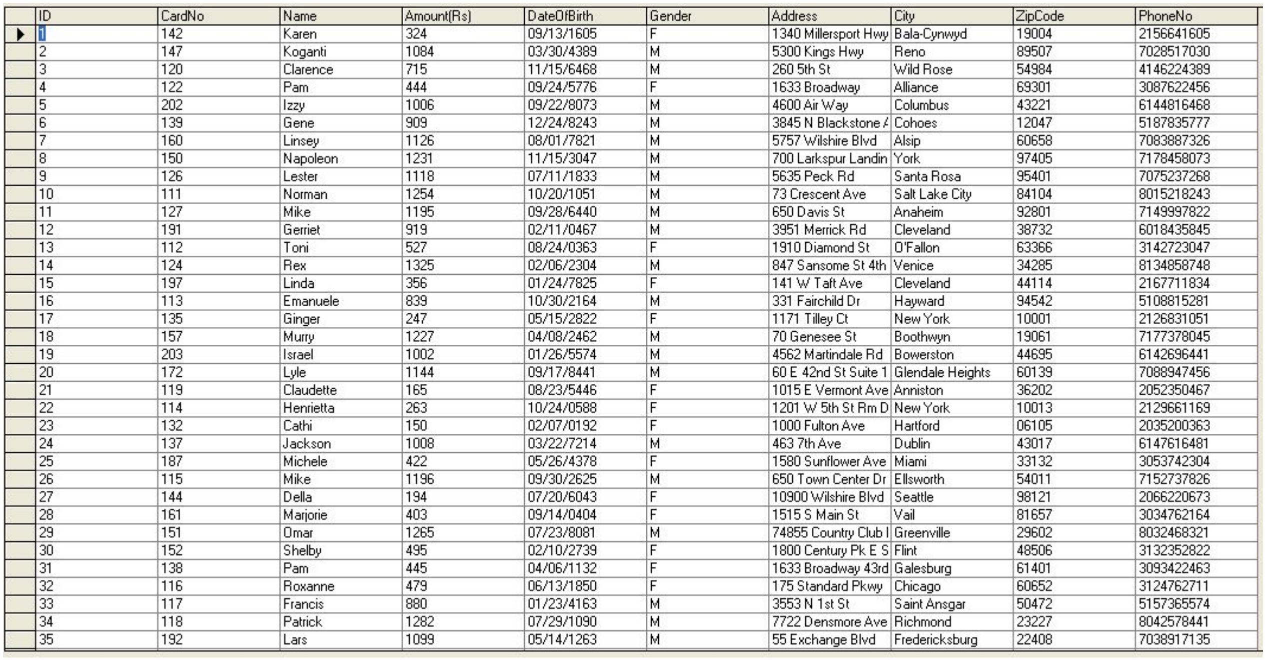
1. Save the script
2. Compile the Script
3. Reset the ClassicsA application to the appropriate starting condition.
4. Play back Classics place an order data pool
5. Review the results in the Test manager log.
6. Close Test manager.

**Explanation of Script Commands**

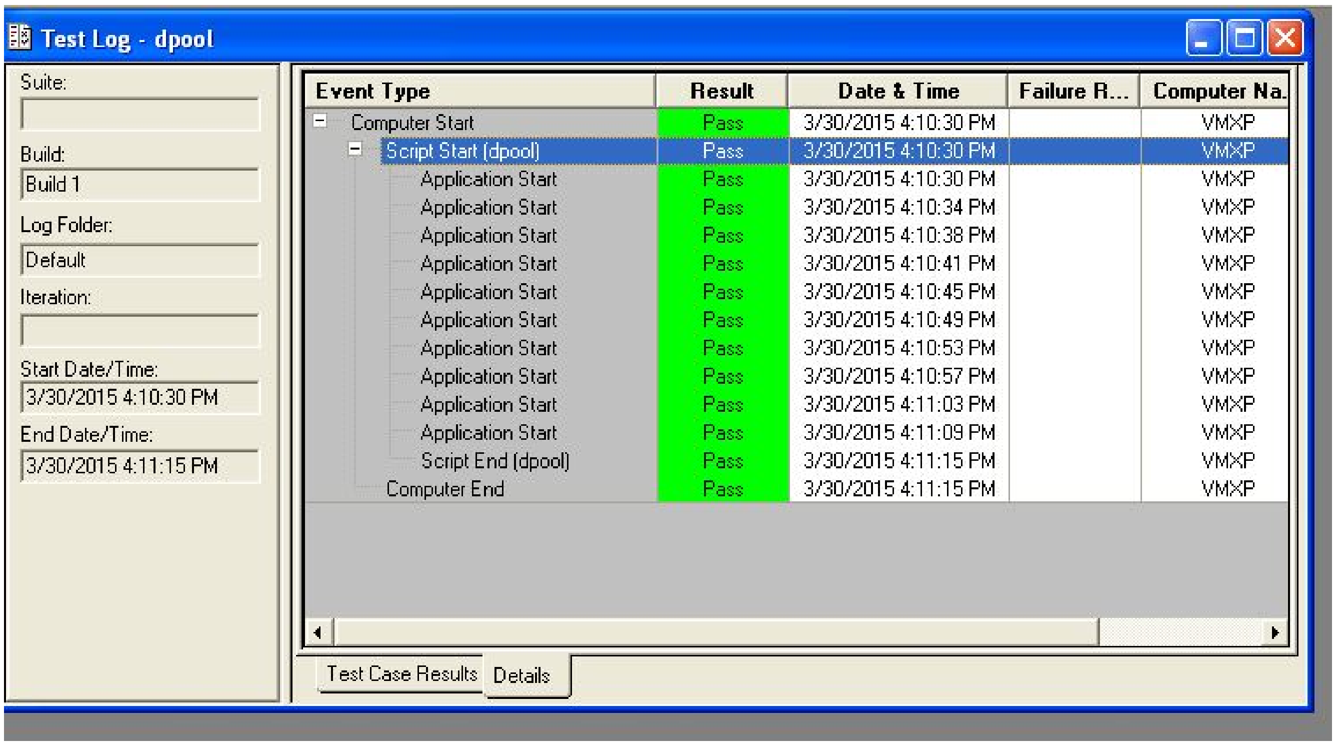
Command:

```
'$include "sqautil.sbh"
Sub main
Dim quantity, x as Integer
Dim expdate as String
Dim dp as long
        dp= SQADatapoolOpen("MyData")
For x=1 to 7
Call SQADatapoolFetch(dp)
'Goto InputKey...line  of quantity and add following commands
Call SQDatapoolValue(dp,1,quantity)
&quantity
'Goto InputKey...line  of  expdate and add following commands
Call SQDatapoolValue(dp,2,expdate)
&expdate
Next x
Call SQADatapoolClose(dp)
End sub
```

**SAMPLE INPUT/OUTPUT**



| ID | CardNo | Name | Amount(Rs) | DateOfBirth | Gender | Address | City | ZipCode | PhoneNo |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 142 | Karen | 324 | 09/13/1605 | F | 1340 Millersport Hwy | Bala-Cynwyd | 19004 | 2156641605 |
| 2 | 147 | Koganti | 1084 | 03/30/4389 | M | 5300 Kings Hwy | Reno | 89507 | 7028517030 |
| 3 | 120 | Clarence | 715 | 11/15/6468 | M | 260 5th St | Wild Rose | 54984 | 4146224389 |
| 4 | 122 | Pam | 444 | 09/24/5776 | F | 1633 Broadway | Alliance | 69301 | 3087622456 |
| 5 | 202 | Izzy | 1006 | 09/22/8073 | M | 4600 Air Way | Columbus | 43221 | 6144816468 |
| 6 | 139 | Gene | 909 | 12/24/8243 | M | 3845 N Blackstone A | Cohoes | 12047 | 5187835777 |
| 7 | 160 | Linsey | 1126 | 08/01/7821 | M | 5757 Wilshire Blvd | Alsip | 60658 | 7083887326 |
| 8 | 150 | Napoleon | 1231 | 11/15/3047 | M | 700 Larkspur Landin | York | 97405 | 7178458073 |
| 9 | 126 | Lester | 1118 | 07/11/1833 | M | 5635 Peck Rd | Santa Rosa | 95401 | 7075237268 |
| 10 | 111 | Norman | 1254 | 10/20/1051 | M | 73 Crescent Ave | Salt Lake City | 84104 | 8015218243 |
| 11 | 127 | Mike | 1195 | 09/28/6440 | M | 650 Davis St | Anaheim | 92801 | 7149997822 |
| 12 | 191 | Gerriet | 919 | 02/11/0467 | M | 3951 Merrick Rd | Cleveland | 38732 | 6018435845 |
| 13 | 112 | Toni | 527 | 08/24/0363 | F | 1910 Diamond St | O'Fallon | 63366 | 3142723047 |
| 14 | 124 | Rex | 1325 | 02/06/2304 | M | 847 Sansome St 4th | Venice | 34285 | 8134858748 |
| 15 | 197 | Linda | 356 | 01/24/7825 | F | 141 W Taft Ave | Cleveland | 44114 | 2167711834 |
| 16 | 113 | Emanuele | 839 | 10/30/2164 | M | 331 Fairchild Dr | Hayward | 94542 | 5108815281 |
| 17 | 135 | Ginger | 247 | 05/15/2822 | F | 1171 Tilley Ct | New York | 10001 | 2126831051 |
| 18 | 157 | Murry | 1227 | 04/08/2462 | M | 70 Genesee St | Boothwyn | 19061 | 7177378045 |
| 19 | 203 | Israel | 1002 | 01/26/5574 | M | 4562 Martindale Rd | Bowerston | 44695 | 6142696441 |
| 20 | 172 | Lyle | 1144 | 09/17/8441 | M | 60 E 42nd St Suite 1 | Glendale Heights | 60139 | 7088947456 |
| 21 | 119 | Claudette | 165 | 08/23/5446 | F | 1015 E Vermont Ave | Anniston | 36202 | 2052350467 |
| 22 | 114 | Henrietta | 263 | 10/24/0588 | F | 1201 W 5th St Rm D | New York | 10013 | 2129661169 |
| 23 | 132 | Cathi | 150 | 02/07/0192 | F | 1000 Fulton Ave | Hartford | 06105 | 2035200363 |
| 24 | 137 | Jackson | 1008 | 03/22/7214 | M | 463 7th Ave | Dublin | 43017 | 6147616481 |
| 25 | 187 | Michele | 422 | 05/26/4378 | F | 1580 Sunflower Ave | Miami | 33132 | 3053742304 |
| 26 | 115 | Mike | 1196 | 09/30/2625 | M | 650 Town Center Dr | Ellsworth | 54011 | 7152737826 |
| 27 | 144 | Della | 194 | 07/20/6043 | F | 10900 Wilshire Blvd | Seattle | 98121 | 2066220673 |
| 28 | 161 | Marjorie | 403 | 09/14/0404 | F | 1515 S Main St | Vail | 81657 | 3034762164 |
| 29 | 151 | Omar | 1265 | 07/23/8081 | M | 74855 Country Club I | Greenville | 29602 | 8032468321 |
| 30 | 152 | Shelby | 495 | 02/10/2739 | F | 1800 Century Pk E S | Flint | 48506 | 3132352822 |
| 31 | 138 | Pam | 445 | 04/06/1132 | F | 1633 Broadway 43rd | Galesburg | 61401 | 3093422463 |
| 32 | 116 | Roxanne | 479 | 06/13/1850 | F | 175 Standard Pkwy | Chicago | 60652 | 3124762711 |
| 33 | 117 | Francis | 880 | 01/23/4163 | M | 3553 N 1st St | Saint Ansgar | 50472 | 5157365574 |
| 34 | 118 | Patrick | 1282 | 07/29/1090 | M | 7722 Densmore Ave | Richmond | 23227 | 8042578441 |
| 35 | 192 | Lars | 1099 | 05/14/1263 | M | 55 Exchange Blvd | Fredericksburg | 22408 | 7038917135 |



Test Log - dpool

Suite:

Build: Build 1

Log Folder: Default

Iteration:

Start Date/Time: 3/30/2015 4:10:30 PM

End Date/Time: 3/30/2015 4:11:15 PM

| Event Type | Result | Date & Time | Failure R... | Computer Na... |
|---|---|---|---|---|
| Computer Start | Pass | 3/30/2015 4:10:30 PM | | VMXP |
| Script Start (dpool) | Pass | 3/30/2015 4:10:30 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:30 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:34 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:38 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:41 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:45 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:49 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:53 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:10:57 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:11:03 PM | | VMXP |
| Application Start | Pass | 3/30/2015 4:11:09 PM | | VMXP |
| Script End (dpool) | Pass | 3/30/2015 4:11:15 PM | | VMXP |
| Computer End | Pass | 3/30/2015 4:11:15 PM | | VMXP |

## Exercise No. 12 Website Testing

In this lab, you access the structure of a sample website by using a website scan verification point. When you insert a website scan verification point, you specify exactly which website related defects (broken links, orphan files, etc). You want to check for at play back. If any defects are detected during playback the website verification point will fail and you will be able to use site check to analyze and fix the problem.

**Objectives**

- Record a Robot script that contains a Web Site verification point.
- Analyze a Web site.
- Repair broken links.
- Use the Rational Site Check Wizard.

**PRE-REQUISITE: HTML KNOWLEDGE**

**Pre-Lab: Online CD Purchasing System**

**PROCEDURE:**

**Record a Robot script that contains a Web Site verification point**

Steps

1. If necessary, start Robot and connect to IT Project.
2. In the GUI Playback Options, verify that Acknowledge results are checked.
3. Record a GUI Script.
4. Type Web Test as the Script Name, then click OK.
5. Click the Display GUI Insert Toolbar on the GUI Record Toolbar.
6. Click Web Site Scan on the GUI insert toolbar.
7. Type Web Site Scan 1 as the Verification Point Name.
8. Click OK.
9. Select Local Site and browse for C;\Program Files\Rational\Rational Test\Samples\Web\default.htm.
10. Click Open.
11. Click Next.
12. Verify that Read Entire Site is selected, then click Next.
13. Select the Option as shown above
14. Click Finish.
15. Stop Recording.

**Play Back the Web Test Script to Check the Site.**

Steps

1. Play back the Script.
2. Type Web Test as the Script Name, then click OK.
3. Click OK to accept the default Log Specification Information.
4. Respond to the Acknowledge Verification Point Result message.
5. Double-click the Verification Point (Web Site Scan 1) line in the Test Log Window.

**Explore Different Views to Find Broken Links.**

Steps:

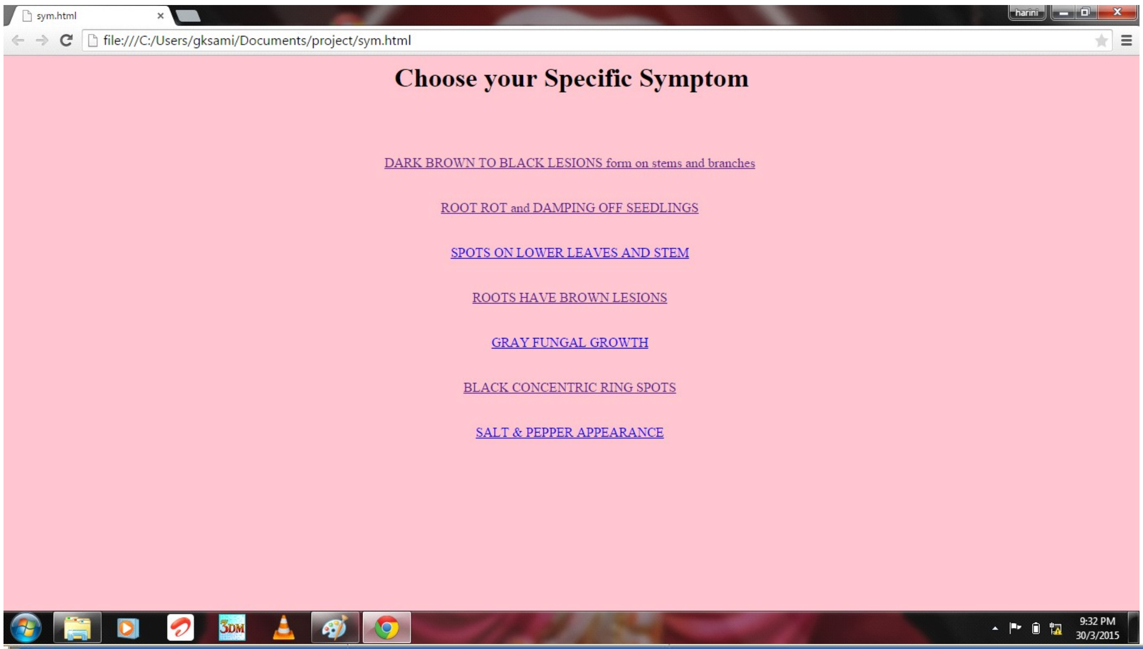
1. In the List Views edit box at the top of the screen, make sure that All Defects is selected.
2. From the Rational Site Check menu, select List View> Defects>Pages with Broken Links.
3. Right –click on the header of any column in the report, Select Add column >Title from the list.
4. Select the first broken link on the report, From the main menu, select View>Source View.

**Fix a Broken Link**

Steps:

1. At the Source View Page, examine the broken link highlighted in red:
   href= "..//offices/offices.htm". Browse the offices folder on the file tree in the left
   pane.
2. Click Save Src (not the Save button) on the Rational Site check toolbar.
3. Make sure that this link has changed from red to green.
4. Click View>Page View.
5. Expand the offices folder in the file tree and select the slogan.htm file.
6. Change to source view.
7. Repair the broken Link. Then, save the source.
8. View the pages with Broken Links page. Select the remaining broken link page and try to find the error.
9. Change the file name in the left pane list to match what is in the source.
10. Click Yes.
11. Click OK in the Rational Site Check Link Wizard.
12. View All Defects.

**SAMPLE INPUT/OUTPUT**



file:///C:/Users/gksami/Documents/project/sym.html

Choose your Specific Symptom

DARK BROWN TO BLACK LESIONS form on stems and branches

ROOT ROT and DAMPING OFF SEEDLINGS

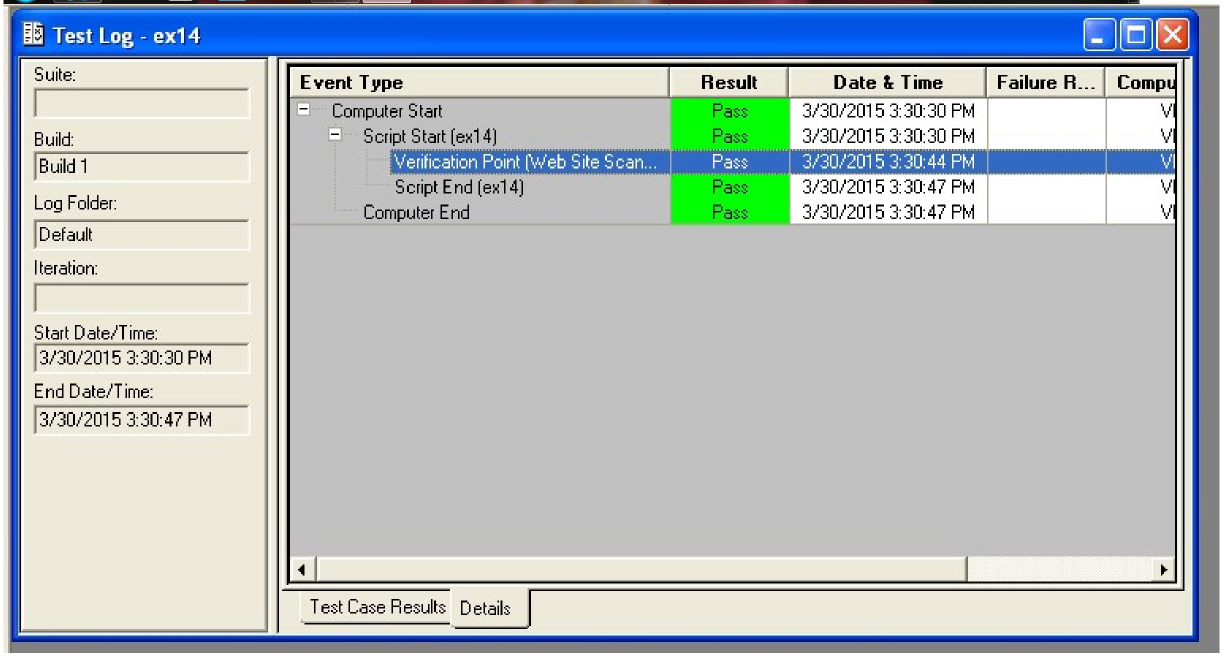
SPOTS ON LOWER LEAVES AND STEM

ROOTS HAVE BROWN LESIONS

GRAY FUNGAL GROWTH

BLACK CONCENTRIC RING SPOTS

SALT & PEPPER APPEARANCE



Test Log - ex14

| | |
|---|---|
| Suite: | |
| Build: | Build 1 |
| Log Folder: | Default |
| Iteration: | |
| Start Date/Time: | 3/30/2015 3:30:30 PM |
| End Date/Time: | 3/30/2015 3:30:47 PM |

| Event Type | Result | Date & Time | Failure R... | Compu |
|---|---|---|---|---|
| Computer Start | Pass | 3/30/2015 3:30:30 PM | | V |
| Script Start (ex14) | Pass | 3/30/2015 3:30:30 PM | | V |
| Verification Point (Web Site Scan... | Pass | 3/30/2015 3:30:44 PM | | V |
| Script End (ex14) | Pass | 3/30/2015 3:30:47 PM | | V |
| Computer End | Pass | 3/30/2015 3:30:47 PM | | V |

Test Case Results | Details

**Additional Experiment**

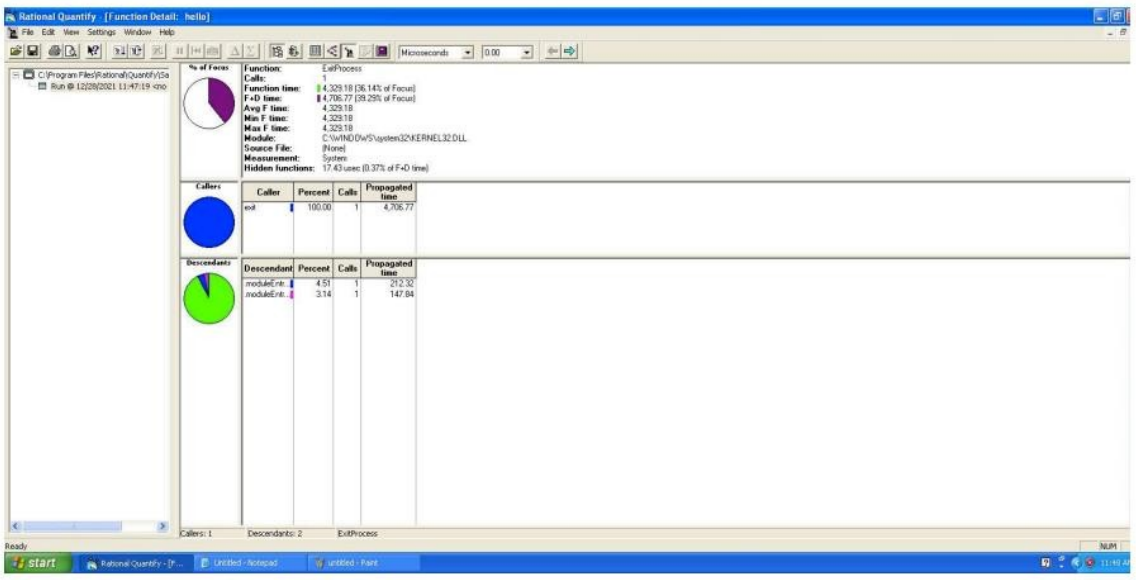
## ExerciseNo.14:

## Find speed and efficiency of application program using RationalQuantify

**Aim:** To find speed and efficiency of application
program using RationalQuantifyProcedure:

Step1: Start Rationalsuite
Step2: Create yourapplicationand generateitas*.exe
Step3: Open RationalQuantifyto
measurethespeedand timeofgivenapp.Step4:Give
input of your*.exefile
Step5: Run Rational Quantifytool
Step6: Get callgraph, timeline chart
as in the given output
Step7: Observe the speed and time of given algorithms or sub
routines
Step8: stop the process after checking each unit of given module.
Note: If you are using VBproject, select quantify integrate tool to get
profile.

SampleI/O:



Rational Quantify - [Function Detail: hello]

```
Function:          ExitProcess
Calls:             1
Function time:     4,329.18 (36.14% of Focus)
F+D time:          4,706.77 (39.29% of Focus)
Avg F time:        4,329.18
Min F time:        4,329.18
Max F time:        4,329.18
Module:            C:\WINDOWS\system32\KERNEL32.DLL
Source File:       (None)
Measurement:       System
Hidden functions:  17.43 usec (0.37% of F+D time)
```

| Caller | Percent | Calls | Propagated time |
|---|---|---|---|
| exit | 100.00 | 1 | 4,706.77 |

| Descendant | Percent | Calls | Propagated time |
|---|---|---|---|
| moduleEntr... | 4.51 | 1 | 212.32 |
| moduleEntr... | 3.14 | 1 | 147.84 |

Callers: 1    Descendants: 2    ExitProcess