



Application Developers'
Intro School
Specifications Binder
Segment 6



Table of Contents

Activity Instructions	3
Overview	3
Activity Steps	3
Key Learning Points	3
Learning Materials	3
Step 1 – Prepare for this Activity	4
Step 2 – Prepare the Environment	5
Step 3 – Code the Java Module	6
Step 4 – Save Files to CVS	6
Step 5 – Compile and Debug Code	7
Step 6 – Execute Scripts, Verify, Debug and Fix	7
Step 7 – Reflect on Key Learning Points	8
Technical Specifications	9
Purpose	9
Project Description	9
Terminologies Used	9
Functionalities	10
Architectural Overview	11
Systems Integration	12
Unit Testing	13

Activity Instructions

Overview

In this segment, you are expected to make changes upon the mediation systems you created in a previous segment by coding additional functionalities based on changing business requirements.

This segment is designed for you to experience change management. It also introduces you to the advanced Java feature that facilitates object function calls between Java Virtual Machines (JVMs) –Remote Method Invocation (RMI).

Activity Steps



To complete the activity you need to follow the steps above. These steps are explained later in greater detail in this section of the binder.

Key Learning Points

At the end of the module, you need to learn and understand:

- What change management is
- How Remote Method Invocation (RMI) works

Learning Materials

The following information may be found within this binder, the Reference Guide or the Performance Support Tool. Because you have used the materials in the previous segments you should be familiar with what the resources contain and how they will be used. This section describes the purpose and location of each document.

Input

What	Why	Where
Activity Instructions	Guides you through the process of completing the activity	Activity Instructions section in this binder
Technical Specifications	Used as a blueprint for coding java logic	Technical Specifications section in this binder

Reference Materials

What	Why	Where
Java Overview and Java Glossary	Provides information on Java syntax	Reference Guide
Remote Method Invocation Overview	Provides information on RMI concepts	Reference Guide
Change Management Procedures	Provides information on change management procedures in the Smart SDLC	SDLC in the Performance Support Tool
Frequently Asked Questions	Provides answers to questions found throughout the Activity Instructions	Frequently Asked Questions in the Performance Support Tool

Outputs

What	Why	Where
Mediation2.java	Process transaction records, filter records or reject records based on criteria and create a report of the processing to submit through RMI	Your computer
RMIserver.java	Server for RMI	Your computer

Step 1 – Prepare for this Activity

1. Review the information contained in each tab of the Activity Binder.

Briefly review the Activity Overview. Browse the sections of the technical specifications. Get an overview of the information they contain. You will use these sections extensively when you are coding and debugging so it is helpful to gain a general understanding of these materials.

2. Mark your Reference Guide and Performance Support Tool.

Throughout this binder you will encounter questions inside text boxes. These are relevant to the tasks at hand. You will find the answers to these questions from either the Reference Guide or Performance Support Tool provided to you.

3. Confirm your understanding of the application logic.

To confirm your understanding of the application logic, try to answer the questions below:

- a. What business requirement is being fulfilled by the Mediation updates?

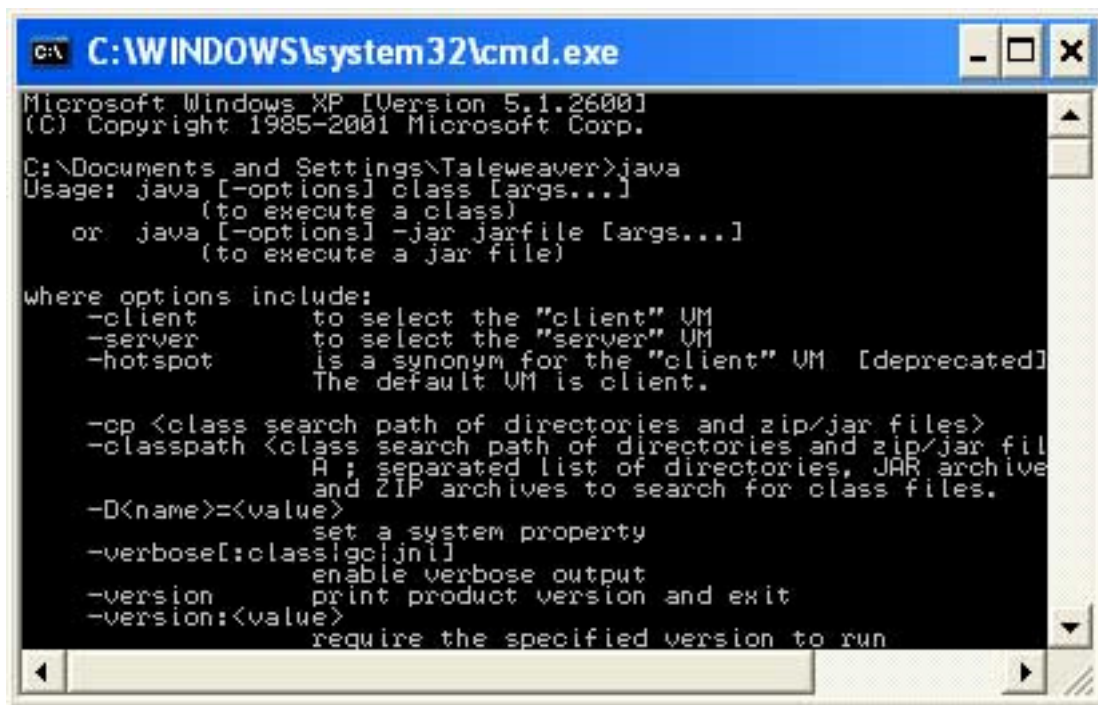
- b. How does the Mediation class and RMIclient class interface?
- c. How are rejected records handled by the module? Why is this so?

Step 2 – Prepare the Environment

1. Create the following directory: `c:\rmi`.
2. Add the path `c:\rmi` to your `CLASSPATH` variable under `USER VARIABLES` found in `ENVIRONMENT VARIABLES`.

Note: Refer to the document APPDEV PREPARATION if you don't remember how to modify Environment Variables in Windows.

3. You will use the command line to compile your java code for this segment. You need to make 4 java class files: the interface, remote object, server and client. Click on `Start – Run` then type `CMD` and then press `Enter`. You will get a command window. Once the command window is up, type `Java` and press `Enter`. If you don't get the error "Unrecognized command" you should see the screen below:



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Taleweaver>java
Usage: java [-options] class [args...]
           (to execute a class)
   or: java [-options] -jar jarfile [args...]
           (to execute a jar file)

where options include:
  -client           to select the "client" VM
  -server           to select the "server" VM
  -hotspot          is a synonym for the "client" VM [deprecated]
                   The default VM is client.

  -cp <class search path of directories and zip/jar files>
  -classpath <class search path of directories and zip/jar files>
              A ; separated list of directories, JAR archive
              and ZIP archives to search for class files.
  -D<name>=<value>  set a system property
  -verbose[:class|gc|jni]
                   enable verbose output
  -version          print product version and exit
  -version:<value> require the specified version to run
  
```

After this type the command `Javac` and press `Enter`, this is the old style java compiler. You will see the screen below if there are no problems:

Step 5 – Compile and Debug Code

1. Open a command window. Go to the RMI directory where you've saved your code by typing `cd\rmi` and then type: `javac <filename>` in the command line. This will compile your .java files.

Note: The path to the compiler has been previously setup way back in Segment 2. If you get an error message saying that the command does not exist follow the steps in the Appdev Preparation document.

2. If there are errors in your code, you will receive an error message in the following general format:
`<filename>:<line number>: error message`
3. Edit your code in Eclipse or your favorite text editor, save your changes and compile.
4. Should you perform a successful "clean" compile, i.e. syntax and runtime errors were not found in your code, you will not receive a message.
5. List the contents of the directory to verify that the class file has been saved in your directory.
6. Make sure you have saved any changes you made and have recompiled the code.
7. Run the rmi registry by typing the following command in the command window: `start rmiregistry`. Press `enter`. A blank command window will open. Do not close this. This is the rmi registry. Without it your Server will have errors.
8. Run your RMI server by typing the following: `start java RMIServer`. Press `enter`. A blank command window will open. Do not close this. While this window is open your Server will listen for client requests.
9. You are now ready to execute Unit Testing on your code.

Step 6 – Execute Scripts, Verify, Debug and Fix

To perform this step, refer to the Unit Testing tab of this segment binder.

1. Execute steps for each cycle of your Unit Test Script and verify your actual results against the expected results.
2. If you find discrepancies, analyze your code to determine the cause of the error. Correct the error and rerun your test scripts to retest your code.
3. Save files after making changes.
4. Create a Unit Test Report Form using the template provided at the `c:\appdevschool\reports` folder.
5. Save the file as `Segment 6 – Unit Test Report.doc` on the same folder.
6. Submit your work to your instructor.

Step 7 – Reflect on Key Learning Points

Inform your instructor that you have completed the coding and testing activities of this segment. He/she will review your work.

Review the key learning points for this segment. Answer the following questions and discuss them with your co-participants and faculty.

General

1. What was difficult about today's assignment?
2. What will you do differently if you had to repeat today's activity?

Specific

1. How did you determine the extent of changes you will make on your code?
2. What were the advantages of using the RMI feature?
3. Were the comments you created, while coding the application previously, clear enough to help you recall the intent of your original code?

Technical Specifications

Purpose

This document details the processing logic for the java manager objects. Sufficient information is provided to write the code. This technical specifications document is a hybrid of sections derived from the full-length Functional Specifications and DB Design Documents. In the future (depending on the scope of your project), you may encounter shorter or longer versions of these documents.

- [What are functional specifications? Why do we need it?](#)

Project Description

Based on the study of the Network Services Division of the Call Detail Records (CDRs), they've found that successive attempts (TC=2) of subscribers in activating the call forwarding feature of their phones (SD=20) is being processed as billable transactions. Charges for successive attempts to activate call forwarding should be waived. The mediation system should filter these usage types.

In addition, specifications of the switch state that TC should not be generating values more than or equal to 50. However, these are generated by the system. We want to create a problem report for the vendor to further investigate the issue. The mediation system should reject these transactions, i.e. place them in a separate directory and add the tag "rejected" in the filename prefix.

A Count Report of the total input, total output, total rejected and total filtered records should also be created by the system. The Count Report is reviewed by the Revenue Assurance Group and are used by them in auditing (input-output-reject-filter=0) transactions, as well in recon with the billing system and network element.

The Network Services Division also wants a copy through RMI (Remote Method Invocation).

Terminologies Used

Acronym	Stands for	Description
RN	record number	sequence number from 000000-999999
RT	record type	1- mobile originating, 2-
MSISDN	mobile station integrated services digital network	mobile number of subscriber e.g. 09192293333
IMSI	international mobile subscriber identity	unique ID of SIM used across mobile networks

TC	tariff class	sequence number from 00-99
SD	service description	sequence number from 00-99
TS	time stamp	time stamp of transaction (YYYYMMDDHHMMSS)
UT	usage tyoe	alphanumeric value G<TC><SD> e.g. G1389

Functionalities

- Given a filename and directory, the module should read the files and process them to the format given below.

Item	Input Format	Output Format
Record	<RN><RT><MSISDN><IMSI><TC><SD><TS>	<IMSI><TS><UT><TRANS_ID>

Where:

RN	6 digits (000000-999999)
RT	1 digit (1-mobile originating)
MSISDN	15 digit left justified, padded with space
IMSI	20 digit left justified padded with space
TC	2 digits (00-99)
SD	2 digits (00-99)
TS	14 digits
UT	5 alphanumeric value consisting of the constant "G"+TC+SD e.g. G1234
TRANS_ID	20 alphanumeric value consisting of the constant "NS-"+ RN + space padding e.g. NS-123456 _____

- Processed files should be saved in a subdirectory called "OUTPUT" and given a filename specified in the table below.

Item	Input Format	Output Format
Filename	CDR_YYYYMMDD_NNNN.DAT	MED_NNNN_YYYYMMDD.DAT

- If the TC =2 and SD=20, this should be filtered

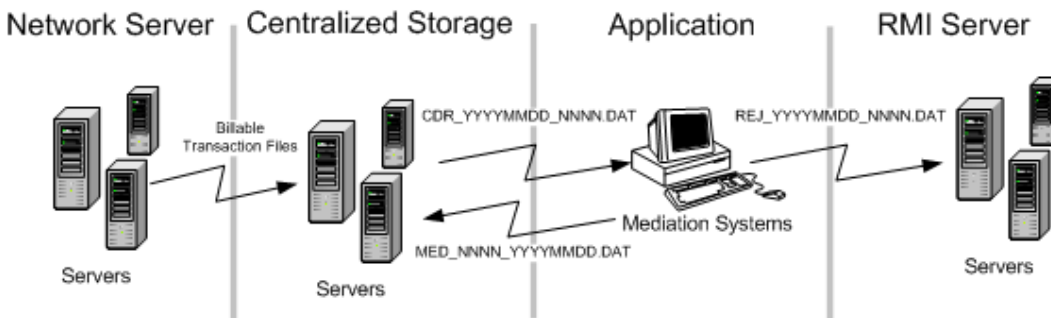
- If the TC \geq 50, it should be outputted in subdirectory called REJECT and given a filename specified in the table below.

Item	Input Format	Output Format
Filename	CDR_YYYYMMDD_NNNN.DAT	REJ_NNNN_YYYYMMDD.DAT

- A given input directory will contain one or more files. If it does not, the system should show an error that the directory is empty.
- Each file contains 60 characters. If it does not, the system should show an error message informing the user that the file has incorrect format and will not be processed, but it must continue to process the other files in the directory.
- Only processed files should be deleted from the input directory.
- The system should count:
 - all the files inputted into the system;
 - all the files filtered;
 - all the files rejected, and;
 - all the files outputted by the system.
- A copy of the rejected files should be passed to the RMIClient as a String

Architectural Overview

The diagram below illustrates the where the billing system component fit in the overall IT infrastructure of the new data service.



Systems Integration

The java module can work with the following java packaged classes to accomplish the requirements defined.

External File, Function or Procedure	Description or Purpose	Type
java.io.file	abstract representation of file and directory pathnames	class
java.io.filereader	Convenience class for reading character files	class
java.io.filewriter	Convenience class for writing character files	class
java.io.reader	Abstract class for reading character streams	class
java.io.writer	Abstract class for writing to character streams	class
java.lang.String	represents character strings; all string literals in Java programs, such as "abc", are implemented as instances of this class.	class
Database class	connects to the database	custom class in custom connections package



Unit Testing

Replace this page with the Unit Test Script and Unit Test Data documents.