NATIONAL INSTITUTE OF TECHNOLOGY

HAMIRPUR (H.P)

SIX WEEKS SUMMER TRAINING REPORT

On

DATABASE EXPLORER

Under the Guidance of

Mr. Lalit Panwar Jangra

Java tutor & Project Instructor

CMC (India) Pvt. Ltd.

Chandigarh, Punjab.

Shrey Saroch

Roll No. : 11528

Dept. Of Computer Science & Engineering

DECLARATION

I hereby declare that I have completed my six weeks summer training at CMC Pvt. Ltd., Chandigarh, Sector-34A from June 2, 2014 to July 17, 2014 under the guidance of Mr. Lalit Panwar Jangra. I have declared that I have worked with full dedication during these six weeks of training and my learning outcomes fulfil the requirements of training for the award of degree of Bachelor of Technology, National Institute of Technology, Hamirpur

SHREY SAROCH
ROLL NO. : 11528
DATE: 16/08/2014

Acknowledgement

I have taken efforts in preparing this project which is a part of my partial fulfilment for the B.Tech component. However, it would not have been possible without the kind support and help of many individuals. Firstly, I would like to express my special gratitude and thanks to my instructor, Mr. Lalit Panwar Jangra for giving me such time and attention. This project has helped me in learning a lot about java language and its applications and importance. It and also has helped me to know about so many new things about Database connectivity in java, and how the queries are executed in similar software like MySQL etc. I am really thankful to my teacher for his guidance and helping completing the project within the limited time.

With best regards.

Contents

Organization Profile

CMC Limited is a leading systems engineering and Integration Company in India, offering application design, development, testing services and asset based solutions in niche segments through turnkey projects of national importance. CMC has also been expanding its service presence in international markets offering off-shoring advantages and delivering value through service-level based and project scope-based deliveries.

It is a subsidiary of Tata Consultancy Services Limited, one of the world's leading information technology consulting services and business process outsourcing organization. CMC Ltd. is a part of the US Tata Group, India's best known conglomerate. Today, CMC Ltd., an ISO 9001:2000, certified and internationally accredited organization, is positioned as a premier IT solutions provider in the fast growing and competitive IT market. CMC's execute large and complex turnkey projects, and have built, managed and supported customers IT systems across the value chain infrastructure, applications and business processes.

With 18 offices, 150 service locations, and 520 non-resident locations and over 10,775 employees worldwide, CMC Ltd. provides a wide spectrum of unique Information Technology solutions and services to a clientele of premier organizations in the government and private sectors.

CMC Academy, Chandigarh was born in 2009 with a mission to enhance the knowledge of students in latest technology which help the students to achieve success in their career. CMC Academy is among largest training provider in India which preferably working in four most important domains. It includes Engineering Design and IT Training, Power Plants engineering, consulting and On-campus training and placement solutions.

CMC Academy, Sector 34, Chandigarh is An ISO 9001:2008 certified Institute and follows the entire quality standard under ISO 9001:2008. The company has a very strong placement cell that assists students to place in various corporates after successful completion of the training.

Basics of Core Java

Java is related to C++, which is a direct descendant of C. Much of the character of Java is inherited from these two languages. From C, Java derives its syntax. Many of Java's object-oriented features were influenced by C++.
In fact, several of Java's defining characteristics come from or are responses to its predecessors. Moreover, the creation of Java was deeply rooted in the process of refinement and adaptation that has been occurring in computer programming languages for the past several decades

History of Java

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called "Oak," but was renamed "Java" in 1995.

Features of Java

Java Applets

An *applet* is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser. Furthermore, an applet is downloaded on demand, without further interaction with the user. If the user clicks a link that contains an applet, the applet will be automatically downloaded and run in the browser. Applets are intended to be small programs.

Security

As you are likely aware, every time you download a "normal" program, you are taking a risk, because the code you are downloading might contain a virus, Trojan horse, or other harmful code. Java achieved this protection by confining an applet to Java execution environment and not allowing access to other parts of computer.

Portability

Portability is a major aspect of the Internet because there are many different types of computers and operating systems connected to it. If a Java program were to be run on virtually any computer connected to the Internet, there needed to be some way to enable that program to execute on different systems. For example, in the case of an applet, the same applet must be able to be downloaded and executed by the wide variety of CPUs, operating systems, and browsers connected to the Internet.

Object-Oriented

It is an object oriented programming language which involves classes, instance of classes (objects), inheritance, interfaces, packages and many other features of classes.

Robust

Java is a strictly typed language, it checks the typed code at compile time. However, it also checks the code at run time. Many hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java.

Multithreaded

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things simultaneously.

Structure of Java Program

```
Package pkgname;
import pkgname.subpackage.class;
public  class_name
{
public static void main(String [] args)
        {
        //Declaration of variable
        //Rest of program
        }
}
```

1. Package Statement: This statement is used to define the location of the program

2. Import Statement: It is used to import already defined packages or classes which may be either java predefined package or the user defined package.

```
import java.util.*;

import java.awt.color;
```

3. Definition of interfaces (if required):

```
Interface name
{
//Members of interface
 }
```

4. Definition of Main Class

5. Definition of Other required Classes

6. Declaration of the variables, array and objects required in the program

Data Types, Variables and Arrays

Data Type:

Data type is the type of value to be stored in a variable. It can be classified as either Basic Data type or Derived Data type.

Basic Data Types: Basic Data Types are listed below:

Sr. No.	Data Type	Size(in bytes)	Range of Values
1.	Byte	1	-2^7 to 2^7-1
2.	Short	2	-2^{15} to $2^{15}-1$
3.	Char	2	-2^{15} to $2^{15}-1$
4.	Int	4	-2^{31} to $2^{31}-1$
5.	Float	4	-2^{31} to $2^{31}-1$
7.	Double	8	-2^{63} to $2^{63}-1$
8.	Long	8	2^{63} to $2^{63}-1$

Derived Data Type

These data types always depends on basic data types. e.g Objects, array etc.

Object Declaration

Class name object_name=new class_name(); // Will be explained later

Array Declaration

Data_type array_name[]=new Data_type(Size of array);

Variable

It is a named memory location which can store a value of a particular type.

Declaration of variable:

Data_type variable_name=value;

Type Casting and Type Conversion

In this type of methods, we can convert the value of a variable to another data type as required in the program.
We have two types of techniques of Typecasting:

- Implicit Type Casting: The type casting which is performed by the system itself is called Implicit Type Casting.
  E.g. int a=456;
  long l=a; //Widening/expansion of data types.
- Explicit Type Casting: This type of type casting is performed by the user. In this case narrowing of the data types.
  e.g –int a=454;
  byte b=a;

Operators

Operators are the symbols which are used to perform specific operation and the value of the operation depends upon the type of operation that we have perfomed on the values.

Classification of Operators as per No. of operands:

Unary Operator: In this type of operators, we have only one value for the operation to be performed.

E.g.: ++, --, ~, !

Binary Operator: In this type of operators, we have two values for the operation to be performed.

E.g.: +,-,*,%,/.

Ternary Operator:-In this type of operators, we have three operands for the operation to be performed.

E.g.: Conditional Operator:

Condition ? true expression : False expression.

Classification as per Operation:

We have various categories of Operation:

1. Arithmetic Operators: These operators are used to perform the arithmetic operation.

e.g.: +, -, /, *

2. Assignment Operators: These operators assign value to variables.

e.g.: =, ==, /=, *=

3. Compound Operators: More than one operator in symbols is called Compound Operator.

e.g.: +=, -=, /=

4. Increment/Decrement Operator: These are of two types:

1. Postfix Operator: The value is incremented after it is assigned to the variable.

2. Prefix Operator: The value is incremented first and then it is assigned to the variable.

5. Comparison Operator: These Operators compare the value of the operands.

e.g.: ==, >, <, >=, <=

Control Statements

Java Control statements control the order of execution in a java program, based on data values and conditional logic. There are three main categories of control flow statements;

· Selection statements: if, if-else and switch.

· Loop statements: while, do-while and for.

· Transfer statements: break, continue, return, try-catch-finally and assert.

We use control statements when we want to change the default sequential order of execution.

Math Class and String Class.

Math is a Predefined class of Java.lang package which consist of various Mathematical functions. All of its functions are static functions and hence we can invoke that directly without using any object.

Functions of Math Class are:

1. *Absolute value using java Math Class*
   Syntax:
   Math.abs(variable)
2. *Minimum value using java Math Class*
   Syntax:
   Math.min(variable1,variable2)
3. *Maximum value using java Math Class*
   Syntax:
   Math.max(variable1,variable2)
4. *Rounding number using java Math Class*
   Syntax:
   Math.ceil(variable)
5. *Power of the number using java Math Class*
   Syntax:
   Math.pow(variable1,variable2)
6. *Square root of the number using java Math Class*
   Syntax:
   Math.sqrt(variable)
7. *Euler's number e to power of a number using java Math Class*
   Syntax:
   Math.exp(variable)

String Class in Java

The java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.

1. Finding the length of the string

The length() method can be used to find the length of the string.

```
String str = "Car insurance"

System.out.println(str.length());
```

2. Comparing strings

The equals() method is used to comapre two strings.

3. Comparing strings by ignoring case

The equalsIgnoreCase() method is used to compare two strings by ignoring the case.

4. Finding which string is greater

The CompareTo() method compares two strings to find which string is alphabetically greater.

5. Position of a string in another string

The indexOf() method is used to find the position of a string in another string.

6. Extract single character from a string

The CharAt() method is used to extract a single character by specifying the position of the character.

Introduction to Objects, its features

2.1 Classes and Objects

A class is nothing but a blueprint or a template for creating different objects which defines its properties and behaviors. Java class objects exhibit the properties and behaviors defined by its class. A class can contain field and methods to define the behavior of an object.

Methods are nothing but members of a class that provide a service for an object or perform some business logic. Java fields and member functions names are case sensitive. Current states of a class's corresponding object are stored in the object's instance variables. Methods define the operations that can be performed in java programming.

*A **class** has the following general syntax:*

[access specifier][modifier]class class_name[extends baseclass][implements interface]

```
{
//Declaration of variables
Static
        {
//Initilisation of class member
        }

                {
//Initilisation of instance members
                }
        Constructor()
        {

        }

}
```

Access specifier can be public,private or protected used in inheritance.
Modifier can be final,abstract,static used with interfaces.Final means constant, which may stop class from further inheritance.

OBJECTS**:** Instance of class, which can store its independent values and can also access all members of the class.

Declaration:
Classname objectname=new Classname();
We can access the members of an object using the dot operator.
Objectname.variablename=value

2.2 Inheritance

Inheritance allows us to inherit the members from one class to another class. The class from which members are inherited are called base/parent class/super class. And the class in which members of base class are inherited is known as derived/sub/child class.

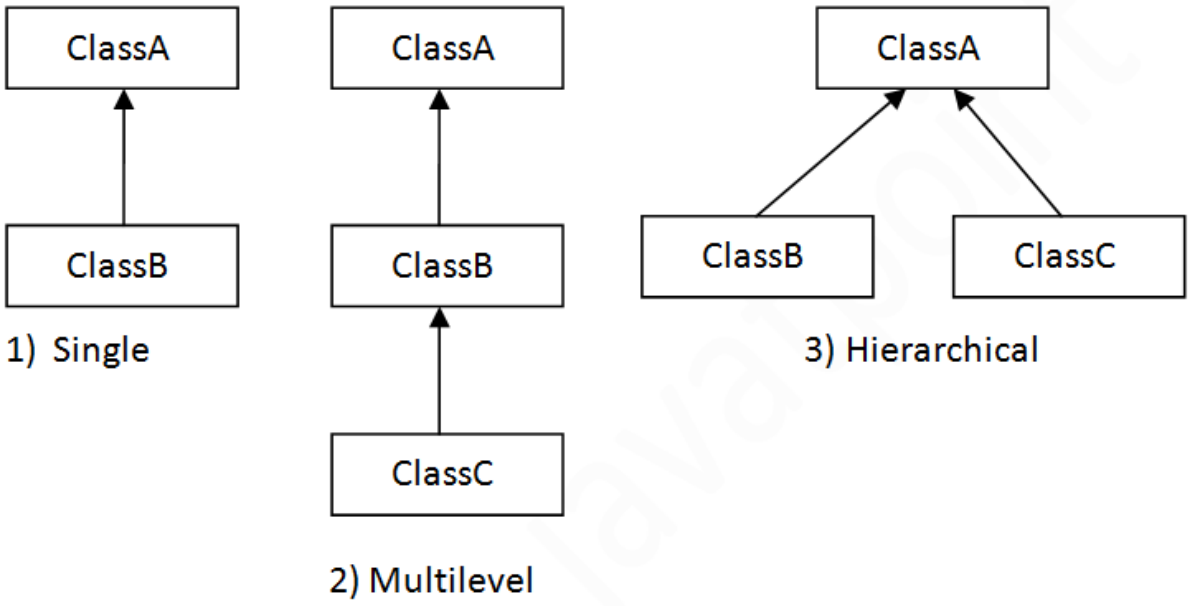
Syntax of Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

Types of Inheritance

On the basis of class, there can be three types of inheritance: single, multilevel and hierarchical in java. In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.

Multiple inheritance is not supported in java in case of class.



2.3 Packages and Interfaces

A **package** is a namespace that organizes a set of related classes and interfaces.
This is the general form of the **package** statement:
package pkg;

Here, pkg is the name of the package. For example, the following statement creates a package called **MyPackage**.
package MyPackage;

Interfaces: These are the the set of attributes and functions like the classes but the interfaces can contain only the declaration of its function and initilisation of the final variables.

Syntax of Interfaces:

[access specifier]interface [interface name] [extends superinterface]
{
//Definition of final variables
//Declaration of member functions
}

NOTE: A class can inherit interface by using the “implements” keyword.

Network Programming

3.1 Introduction to URL class

URL is the acronym for Uniform Resource Locator. It is a reference (an address) to a resource on the Internet.
The constructor of URL class which is used for creating the URL are:-
URL object_name=new URL(string)

S.N.	Methods with Description
1	**public String getPath()** Returns the path of the URL.
2	**public String getQuery()** Returns the query part of the URL.
3	**public int getPort()** Returns the port of the URL.
4	**public int getDefaultPort()** Returns the default port for the protocol of the URL.
5	**public String getProtocol()** Returns the protocol of the URL.
6	**public String getHost()** Returns the host of the URL.
7	**public String getHost()** Returns the host of the URL.

3.2 Introduction to Sockets

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes--Socket and Server Socket--that implement the client side of the connection and the server side of the connection, respectively.

TCP/IP Client Sockets

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point,
Stream-based connections between hosts on the Internet
The creation of a **Socket** object implicitly establishes a connection between the client and server.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.

- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

UDP Sockets

In this type of sockets, these sockets work upon the UDP protocols in which we pass the required data in the form of packets of bytes, only one class DatagramSocket is used for the UDP sockets.

3.3 Introduction to JDBC and ODBC Connectivity

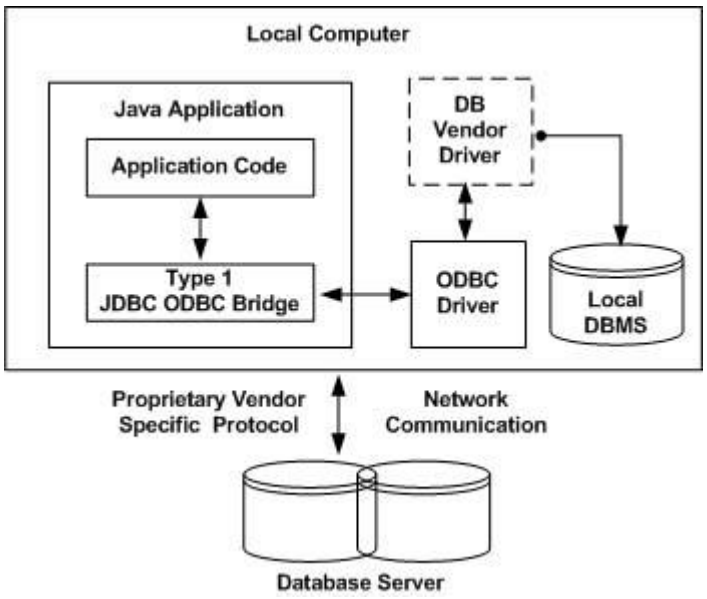
JDBC (Java Database Connectivity)

• JDBC API allows Java programs to connect to DBs
• Provides cross-vendor connectivity and data access across relational databases from different vendors
• Classes and interfaces allow users to access the database in a standard way
• The JVM uses the JDBC driver to translate generalized JDBC calls into vendor specific database calls.

ODBC is an API (Application Programming Interface) definition, compliant with ANSI SQL and X/Open's SQL Call Level Interface (CLI).
Applications can be written to this API standard without considering the intricacies of the various database engines, as the ODBC driver takes care of all the database-specific code, if necessary mapping the structure of the underlying system to a relational framework.
ODBC permits the DBMS-specific parts of the program to be separated from the part that fulfils the business requirement.

JDBC-ODBC Bridge Driver:

In this driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database. When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.



Functionality

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

- ✓ Statement – the statement is sent to the database server each and every time.
- ✓ PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Introduction to GUI

4.1 Applets:

These are the Java's processed code, which can be used in a web page.

Steps to design an applet are:

- Write down the source code of the applet and save it with .java extension
- Compile the program using *javac program_name.java*
- Design applet code and save it with .htm or .html extension
- Execute the applet using applet viewer

4.1.1 Sample Applet code

```
/*
<applet code="MyApplet" width=200 height=60>
</applet>
*/
```

This comment contains an APPLET tag that will run an applet called MyApplet in a window that is 200 pixels wide and 60 pixels high. Because the inclusion of an APPLET command makes testing applets easier, all of the applets shown in this book will contain the appropriate APPLET tag embedded in a comment.

4.1.2 Applet Initialization and Termination (Life Cycle)

It is important to understand the order in which the various methods shown in the skeleton are called. When an applet begins, the following methods are called, in this sequence:

- init( )
- start( )
- paint( )

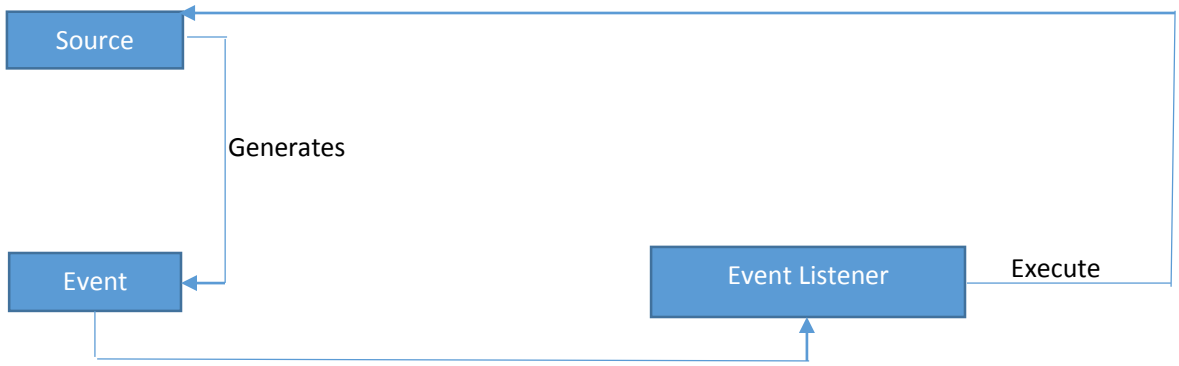
When an applet is terminated, the following sequence of method calls takes place:

- stop( )
- destroy( )

4.2 Event handling

Event handling is fundamental to Java programming because it is integral to the creation of applets and other types of GUI-based programs.

Event Delegation Model



Event Listeners

A listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications.

Event Classes

The classes that represent events are at the core of Java's event handling mechanism. Thus, a discussion of event handling must begin with the event classes. It is important to understand, however, that Java defines several types of events and that not all event classes can be discussed in this chapter. The most widely used events are those defined by the AWT and those defined by Swing.

The ActionEvent Class

An ActionEvent is generated when a button is pressed, a list item is double-clicked, or a menu item is selected.

Sources of Events

EVENT SOURCE	DESCRIPTION
Button	Button Generates action events when the button is pressed.
Check box	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu Item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scroll bar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

4.3 Introducing the AWT: Working with windows text & Graphics

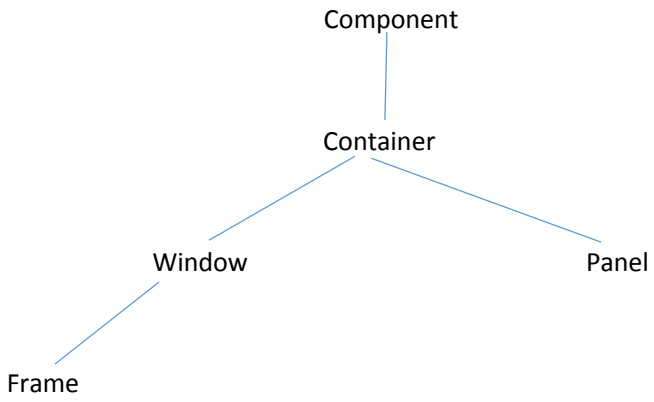
AWT Classes

The AWT classes are contained in the java.awt package. It is one of Java's largest packages. Fortunately, because it is logically organized in a top-down, hierarchical fashion, it is easier to understand and use than you might at first believe.

Classes	Description
AWTEvent	Encapsulates AWT events
AWTEventMulticaster	Dispatches events to multiple listeners
BorderLayout	The border layout manager. Border layouts use five components: North, South, East, West, and Center.
Button	Creates a push button control.
Canvas	A blank, semantics-free window.
CardLayout	The card layout manager. Card layouts emulate index cards. Only the one on top is showing.
Checkbox	Creates a check box control.
CheckboxGroup	Creates a group of check box controls.
Choice	Creates a pop-up list.
CheckboxMenuItem	Creates an on/off menu item.
Color	Manages colors in a portable, platform-independent fashion.
Dimension	Specifies the dimensions of an object. The width is stored in width, and the height is stored in height.
FlowLayout	The flow layout manager. Flow layout positions components left to right, top to bottom.
Font	Encapsulates a type font.
Frame	Creates a standard window that has a title bar, resize corners, and a menu bar.
GridLayout	The grid layout manager. Grid layout displays components in a two-dimensional grid.
Label	Creates a label that displays a string.
Menu	Creates a pull-down menu.
MenuBar	Creates a menu bar.
MenuItem	Creates a menu item.
Panel	The simplest concrete subclass of Container.
Scrollbar	Creates a scroll bar control.
ScrollPane	A container that provides horizontal and/or vertical scroll bars for another component.
TextArea	Creates a multiline edit control.
TextField	Creates a single-line edit control.
Window	Creates a window with no frame, no menu bar, and no title.

4.3.1 Window Fundamentals

The AWT defines windows according to a class hierarchy that adds functionality and specificity with each level. The two most common windows are those derived from Panel, which is used by applets, and those derived from Frame, which creates a standard application window



4.4 Using AWT Controls, Layout Managers, and Menus

4.4.1 The AWT supports the following types of controls:

- Labels
- Push buttons
- Check boxes
- Choice lists
- Lists
- Scroll Bars
- Text Editing

4.4.2 Understanding Layout Managers

All of the components that we have shown so far have been positioned by the default layout manager. Java has several predefined LayoutManager classes, several of which are described next. You can use the layout manager that best fits your application.

FlowLayout

FlowLayout is the default layout manager. This is the layout manager that the preceding examples have used. FlowLayout implements a simple layout style, which is similar to how words flow in a text editor.

BorderLayout

The BorderLayout class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. Here are the constructors defined by BorderLayout:

BorderLayout( )
BorderLayout(int horz, int vert)

GridLayout

GridLayout lays out components in a two-dimensional grid. When you instantiate a GridLayout, you define the number of rows and columns. The constructors supported by GridLayout are shown here:
GridLayout( )
GridLayout(int numRows, int numColumns)
GridLayout(int numRows, int numColumns, int horz, int vert)

The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the horizontal and vertical space left between components in horz and vert, respectively.

CardLayout

The CardLayout class is unique among the other layout managers in that it stores several different layouts. Each layout can be thought of as being on a separate index card in a deck that can be shuffled so that any card is on top at a given time. This can be useful for user interfaces with optional components that can be dynamically enabled and disabled upon user input. CardLayout provides these two constructors:
CardLayout( )
CardLayout(int horz, int vert)
The first form creates a default card layout. The second form allows you to specify the horizontal and vertical space left between components in horz and vert, respectively.

4.4.3 Menu Bars and Menus

A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. This concept is implemented in the AWT by the following classes: MenuBar, Menu, and MenuItem. To create a menu bar, first create an instance of MenuBar. This class only defines the default constructor. Next, create instances of Menu that will define the selections displayed on the bar. Following are the constructors for Menu:

Menu( ) throws HeadlessException
Menu(String optionName) throws HeadlessException
Menu(String optionName, boolean removable) throws HeadlessException

Here, optionName specifies the name of the menu selection. If removable is true, the menu can be removed and allowed to float free. Otherwise, it will remain attached to the menu bar.

Problem Analysis

4.1 Project Definition:

The My Database Explorer is an interface from which a requested information can be accessed from a database. If you are requesting that information from a SQL database (or from a database accessed via ODBC), your query must be written using the Structured Query Language (SQL). The SQL is not difficult to learn, but mastering the fine points of creating and retrieving data with SQL can take quite a while. Since the explorer eliminates the need to understand SQL, it can get you building effective queries right away.

The Explorer has been designed to meet the needs of individuals having query background but looking for light weight query executor to run the queries. If you are looking for a software that works with MySQL and having the simplest features, you will enjoy the way the Explorer. It helps you create queries.

If you are more comfortable writing your own SQL queries, you will find it easy to enter the queries directly or even paste them in from another source.

The DBJ explorer can be a powerful tool for many of your information gathering needs. The DBJ explorer can only access data stored in an ODBC data source. Any SQL or other type of database mentioned in the query topics must be accessed through an ODBC data source.

Modules Description:

Transaction Module

1. Insert: By selecting this option the user can easily insert the values into the table of which he chosen.

2. Select: By selecting this select option the user can able to view the different table names that are present in the database and also records present in the respective chosen table.

3. Delete: By selecting the delete option the user can able to delete the respective record of a particular table.

4. Create: By selecting the create option the user can create a new table.

4.2 Feasibility Analysis

All projects are feasible given unlimited resources and infinite time. But in reality both resources and time are scare. Projects should confirm to time bounds and should be optimal in their consumption of resources. This places a constant approval of any project. After problem is clearly understood and the solutions are proposed the next step is the part of system analysis. The main objective of this study is to determine whether the proposed system is feasible or not.

Feasibility as applied to our system pertains to the following areas:

Operational Feasibility

Operational feasibility is willingness and ability of the employers in the organization to use and support the proposed system. System will give good support and makes the service easy because

- The system has a user interface enabling an easier usage of the software.
- No special training is necessary for users to use this system

Technical feasibility

This software is technically feasible as it is developed in java and it only requires Java, MySQL.

Economic feasibility

Economic feasibility is concerned with the cost savings, increased profits and reductions in effort. This project is economically feasible as it doesn't require any special costs.

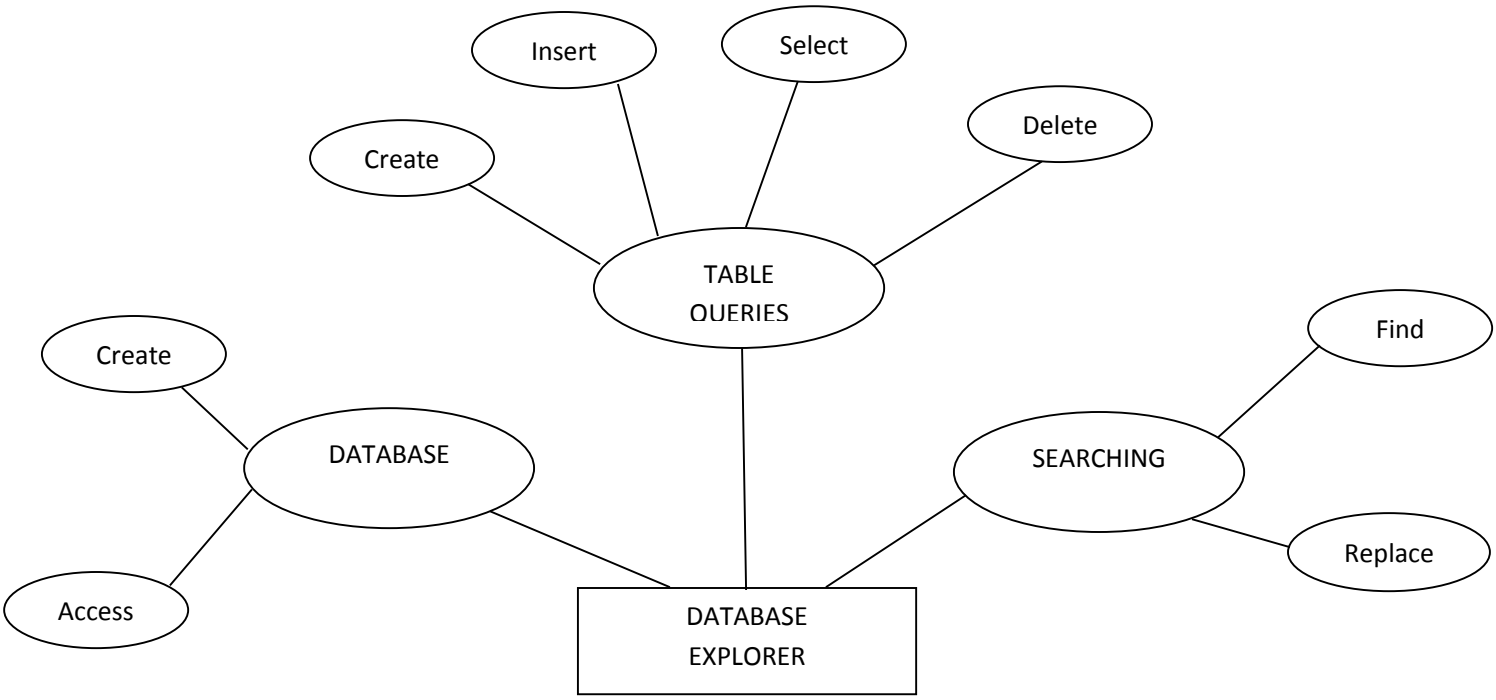
Operating System: Windows 7,Windows 8, Windows 8.1

Front-end: JDK 1.7/1.8,AWT,Swings

Pre-requisites: MySQL

DESIGN

E-R DIAGRAM



Screenshots

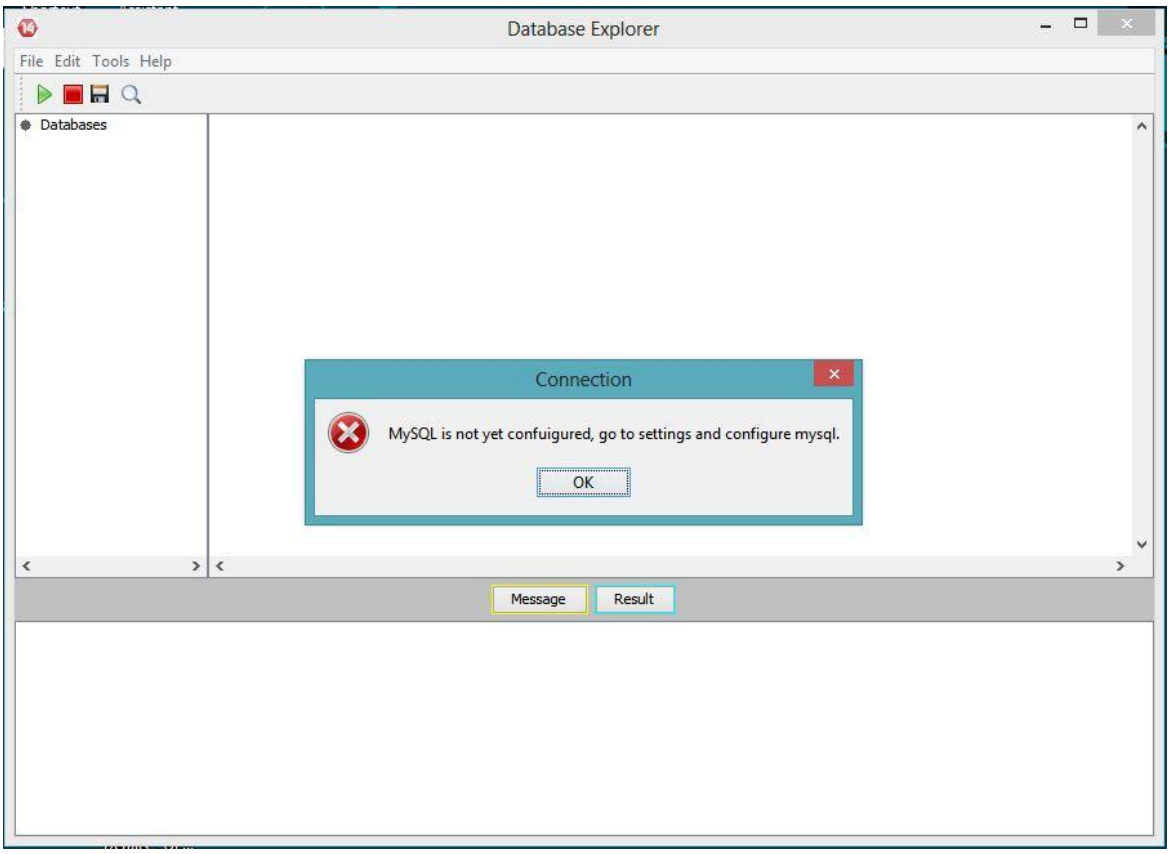


Fig: 8.1: Error message

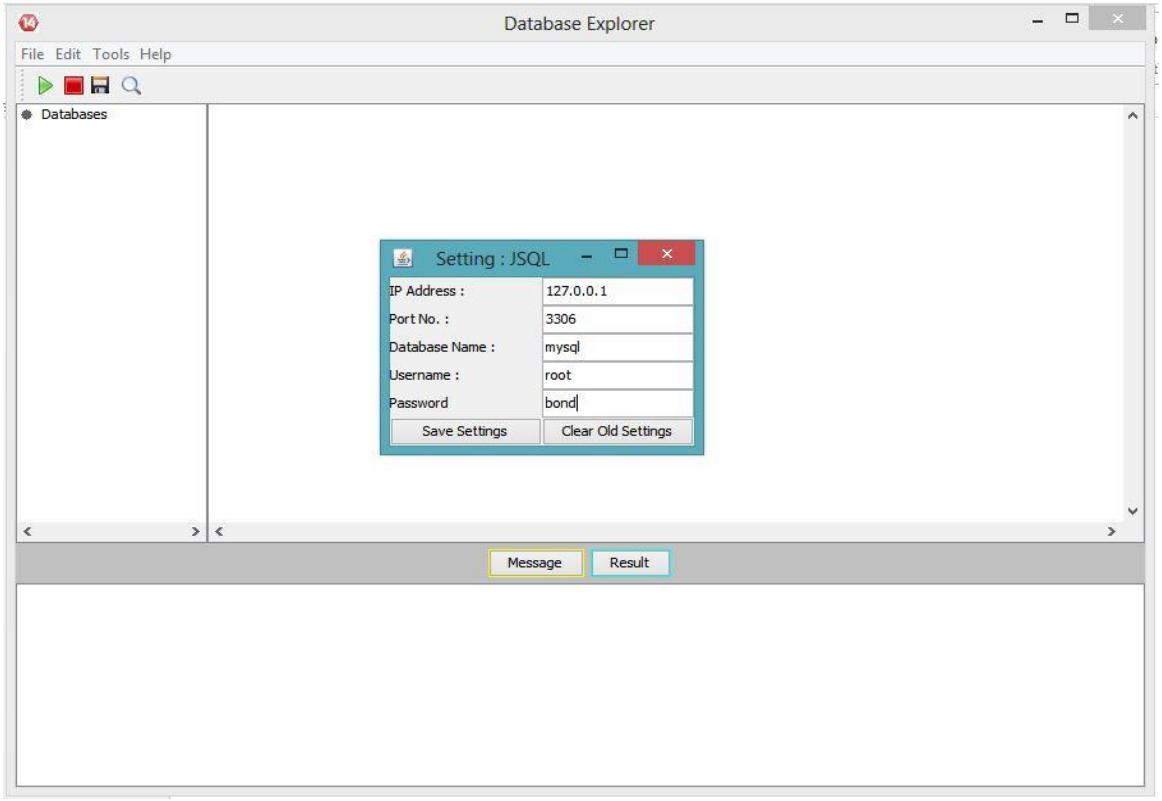


Fig: 8.2: Configuration

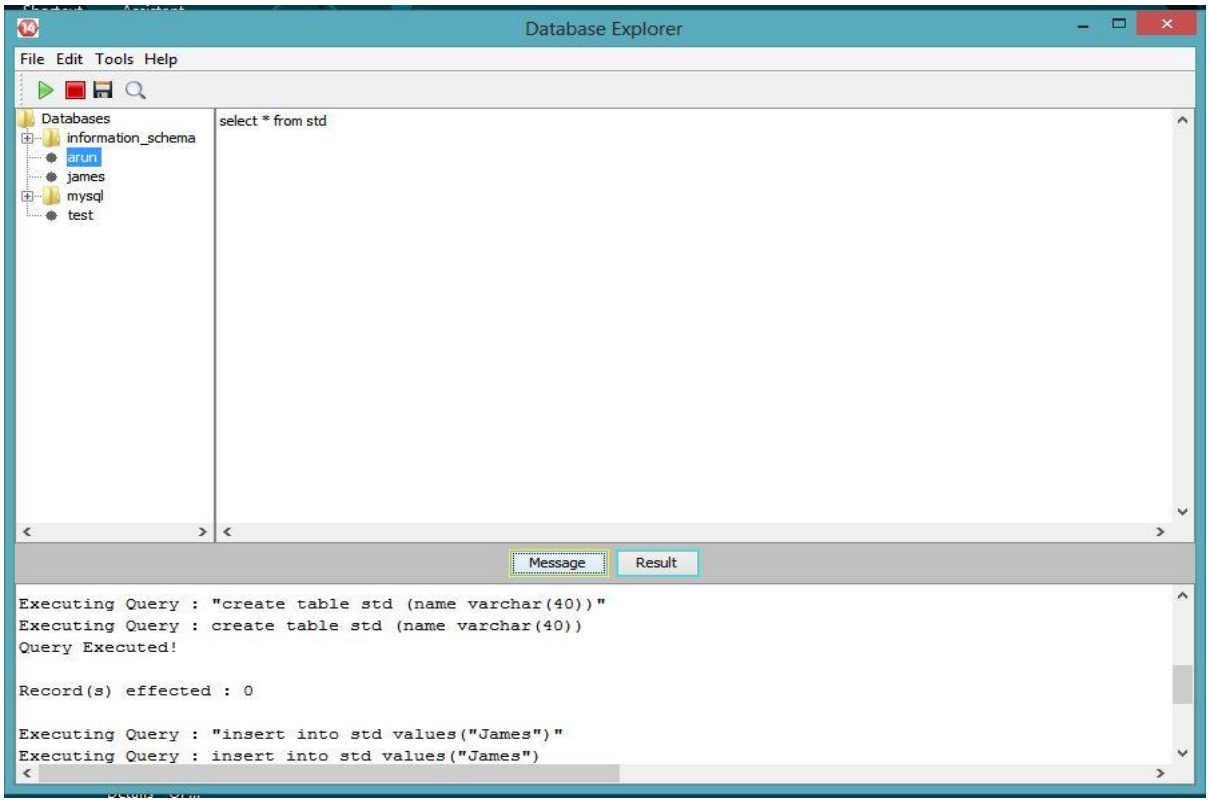


Fig: 8.3: Message pane

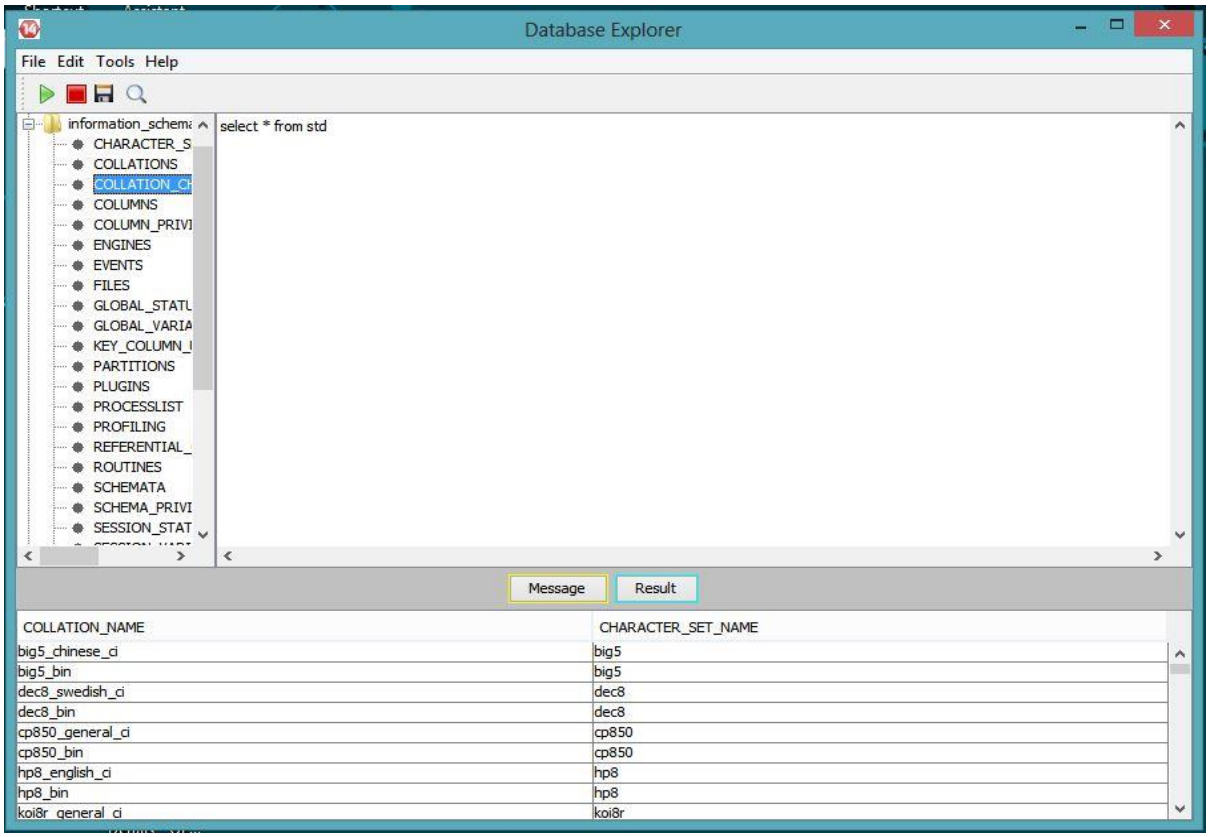


Fig: 8.4: Result pane

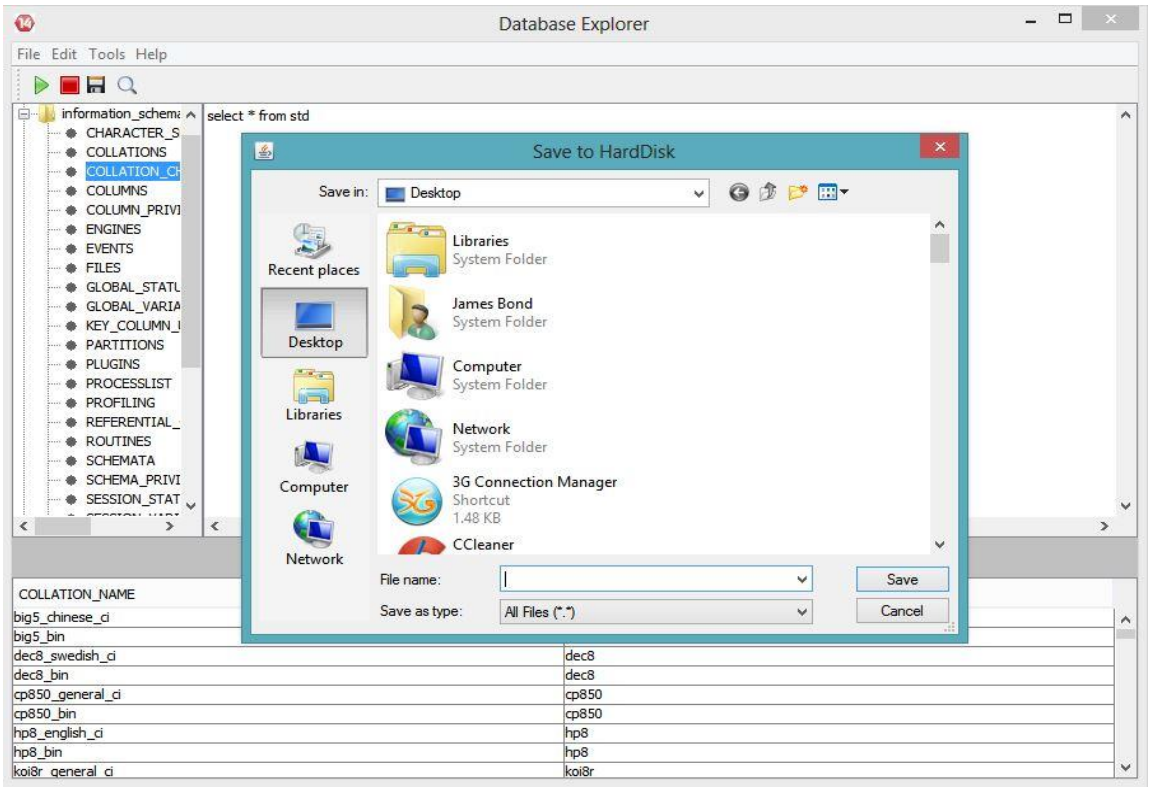


Fig: 8.5: Save dialogue box

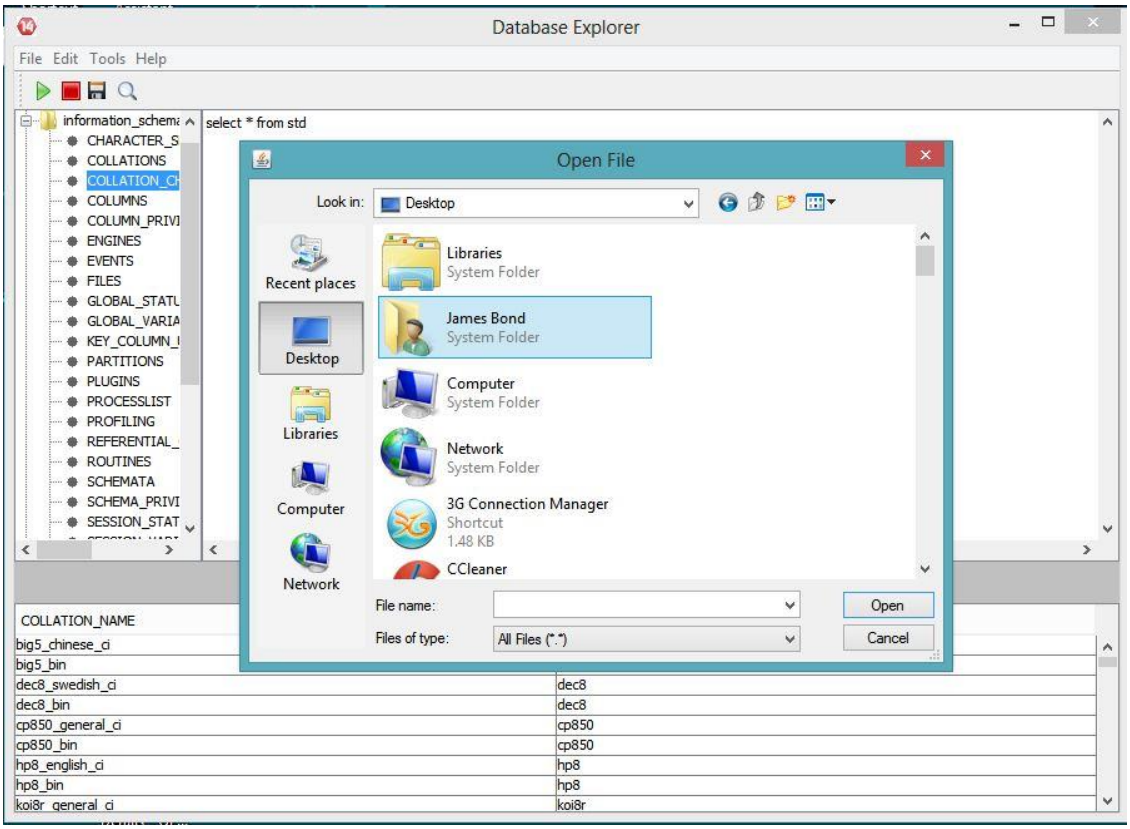


Fig: 8.6: Open dialogue

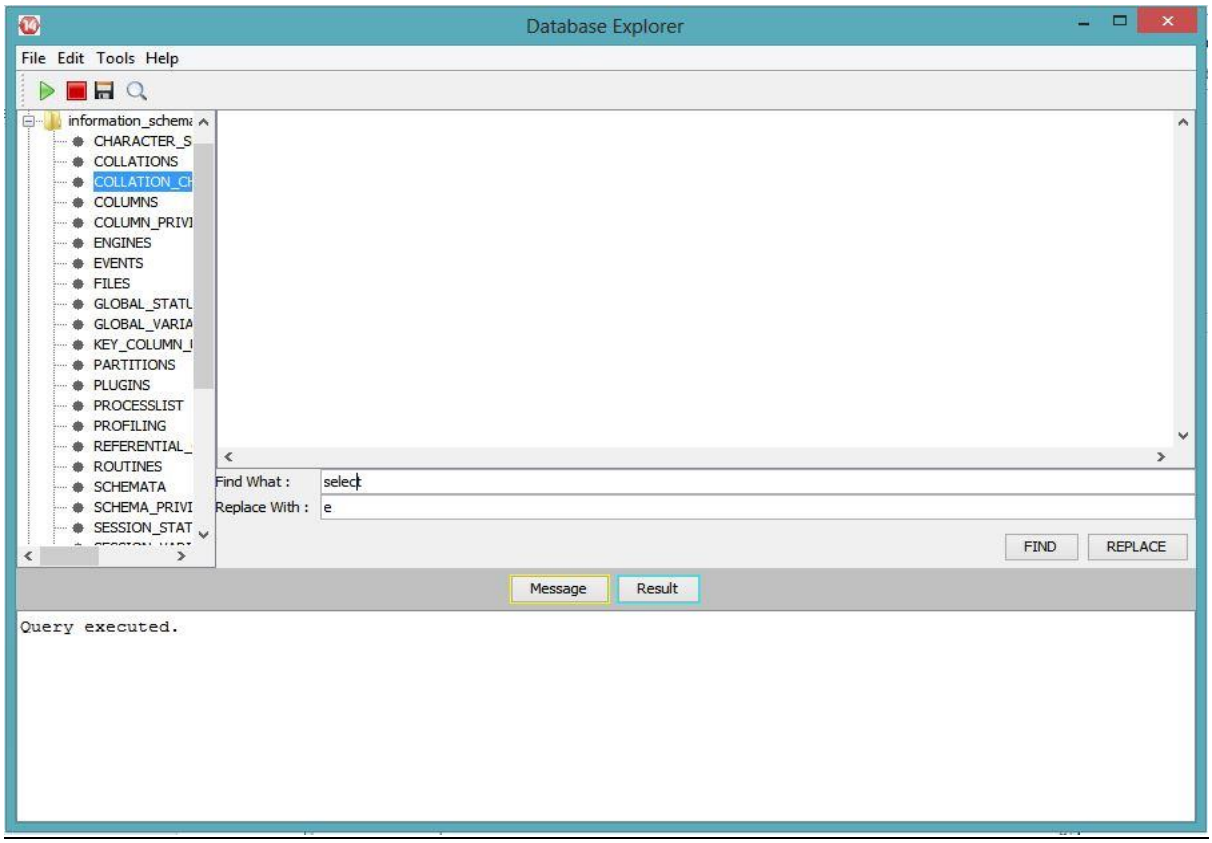


Fig: 8.7: Replace

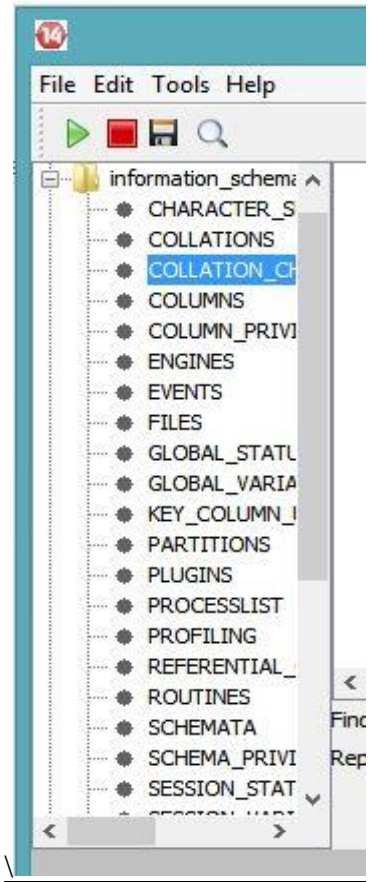


Fig: 8.8: Options

References:

The Complete Reference, Java, 7th Edition – Herbert Schildt