Design of a Motif Database with a Java GUI

Final Project

Darrell Nabors
Towson, MD
darrell32@mac.com

Abstract— *The purpose of this document is to illustrate the phases that went into completing the final project of the Biological Databases and Tools class.*

From the Biological Databases Tools class, I became very interested in motifs and the genes that are associated with them. It would be ideal if a researcher or scientist had their own database in which to store frequently viewed motifs and the genes that are associated with them. They would also be able to download them from ProSite's database and store them in a database on their computer.

To get started, the scope of my project included designing a small standalone database of motifs and their associated genes. The database was built using MySQL tools. Along with the database, A small Java application was designed to interface with the motif database. The Java interface will allow users to sign in to access saved motifs in which they are most interested. Also, users can search for a motif using a ProSite accession number, view current motif listings in the database and add or update listings in the database.

Project Objective

The goal of this project is to design and implement a database, using a relational and SQL-based DBMS, that captures informational aspects of a motif database. The database will be accessed via a Java program that will be design to connect to the database and manipulate information.

Scope and Purpose

The purpose of this document is to illustrate the different phases that went into designing a database for a listing of motifs and associated information. The scope of this project includes the basic design of a database as well as the design of a Java interface that will allow a user to manipulate data from a MySQL database. The standalone database program will allow users to manage their own database and they could have quick access to their most common studied or retrieved motifs without having to go to ProsSite or KEGG.

System Requirements

Hardware

System: Apple MacBook Pro
Processor: 2.4 GHz Intel Core 2 Duo
Memory: 2GB 1067 MHz DDR3

Software

Mac OS X, version 10.7
MySQL Community Server Edition, version 5.1.53
MySQL Workbench, version 5.2.34, revision 7780
Eclipse Java EE IDE, version Indigo, build 20110615-0604

Functional Requirements

- View current listings of motif listings in the database
- Add or edit new motif listings
- Links to other databases or tools regarding the motif listing being viewed (possibly PROSITE, KEGG and UNIPROT)
- Links displaying the most common viewed motifs
- Allow users to sign in and access their most common used motifs
- Allow users to search form protein using a PROSITE ID

Database Design Rationale

For the basic design of the motif database, the perspective of the user was taken into consideration. The database should allow the user to view all of there motifs listings at once. Also, the database should allow the user to add motifs to their database and have a space to store their most common viewed motifs. Also, the user should have access to some basic genes and journals that are related to the motif. In taking these ideas into consideration, there were basically for basic tables established. These tables are the motif, gene, journal and users. The motif table holds five basic items of motif. These items can be found in Table 3 (motif database dictionary). Another table in the motif database is the gene table. This table contains basic gene data that the user may find informative. One unique aspect of the gene table is that each gene of the table is associated with at least one motif. The journal table contains basic data on journals that are associated with the motif. Finally, the users table hold the user ID of people who wish to utilize the database and store their favorite motifs for future reference. The following sections of tables and figures explains the tables, fields and their associated relationships of the motif database.

Because of the consideration of the scope of the project, there was not an option to delete and entry from the database. If future implementations of the database could be considered, the option to delete a motif, an gene or a journal from the database would have to be considered so that the database could be more functional. Also, having the ability to add extra genes as well as journal data. This would add more functionality to the motif database.

Entity Relationship Model

The following data are tables and diagrams that illustrate and explain the entities and there associated relationship between the other entities.

Entity	Description
MOTIF	Contains basic data for the motif
GENE	Contains gene data in which motifs are found
JOURNALS	Contains reference publications in which the motifs are mentioned
USERS	Contains the user's identification to sign on to the database
MOTIF_HAS_USERS	An intersection table that contains the users identification and their corresponding motifs of interest

Table 1. List of database entities.

Entity	Relationships	Entity	Cardinality	Type
MOTIF	is part of	GENE	1:M	identifying
MOTIF	is found in	JOURNALS	1:M	identifying
USERS	have	MOTIF	M:M	non-identifying
MOTIF	is recognized by	USERS	M:M	non-identifying

Table 2. List of database entities and their relationships.

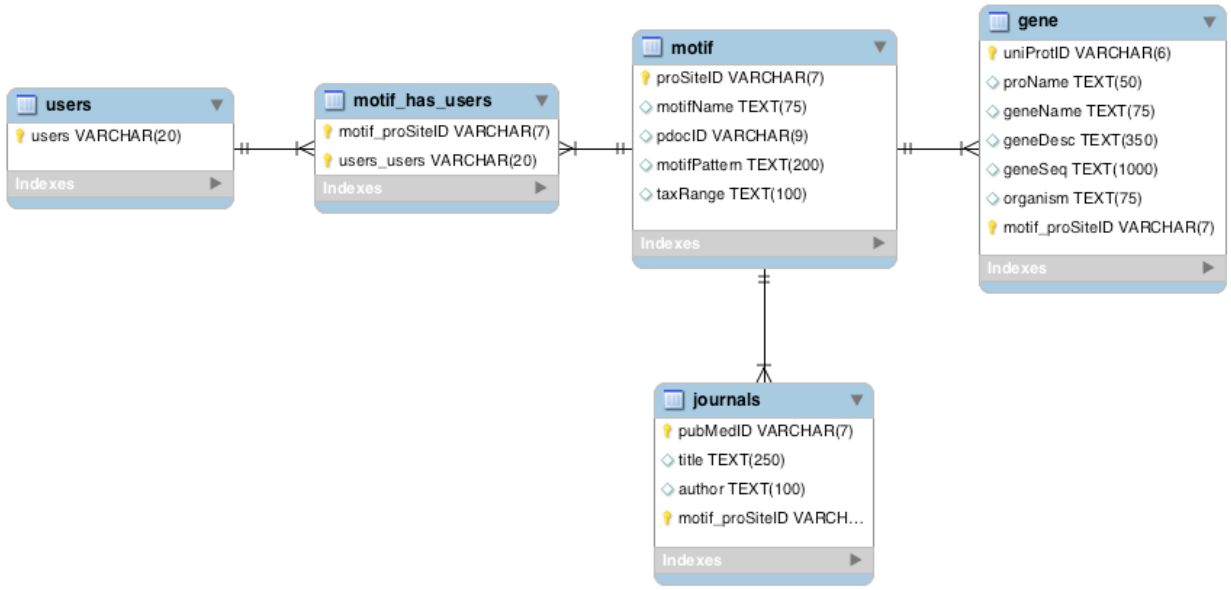


Figure 1. ER Diagram for the motif database

Table	Field	Data Type/Size	Key/Value	Description
MOTIF	proSiteID	VARCHAR(7)	PK/NOT NULL	ProSite listing ID
	motifName	TEXT(75)		Name of motif
	pdocID	VARCHAR(9)		ProSite documentation listing
	motifPattern	TEXT(200)		the pattern of the motif
	taxRange	TEXT(100)		the taxonomic range of motif
GENE	uniProtID	VARCHAR(6)	PK/NOT NULL	Gene ID in UniProt
	proName	TEXT(50)		name of gene
	geneName	TEXT(75)		code name of the gene
	geneDescription	TEXT(350)		description of gene/function
	geneSequence	TEXT(1000)		the amino acid sequence
	organism	TEXT(75)		name of organism of gene
	motif_proSiteID	VARCHAR(7)	FK/NOT NULL	ProSite listing ID
JOURNALS	pubMedID	VARCHAR(7)	PK/NOT NULL	NCBI PubMed listing ID
	motif_proSiteID	VARCHAR(7)	FK/NOT NULL	ProSite listing ID
	title	TEXT(250)		title of journal article
	author	TEXT(100)		authors of journal
USERS	users	VARCHAR(20)	PK/NOT NULL	user ID for signon
MOTIF_HAS_USERS	motif_proSiteID	VARCHAR(7)	FK/NOT NULL	ProSite listing ID
	users_users	VARCHAR(20)	FK/NOT NULL	user ID for signon

Table 3. Motif database dictionary.

CRUD Matrix

The CRUD matrix for this project is constructed to reflect the operations of a database. Of course, the database administrator would have the ability to create, retrieve, update and delete and entry into the motif database. As of now, the users has the ability to retrieve information and add records to the database. In future implementations, it would be ideal to have the user to be able to update and delete records from the motif table as well as other tables such as the gene and journal tables of the motif database.

Entity	Description
E1	Motif: Data on Motif
E2	Gene: Data on Gene that contains motif
E3	Journals: Data on Journal written about the motif
E4	Users: The signon ID for users that have saved a motif
E5	Motif_has_Users: Data on the users and their saved motifs

Table 4. Entity Types for the CRUD Matrix.

Function	Description
F1	**C**reate/**U**pdate/**D**elete/**R**etrieve a motif
F2	**C**reate/**U**pdate/**D**elete/**R**etrieve a gene
F3	**C**reate/**U**pdate/**D**elete/**R**etrieve a journal entry
F4	**C**reate/**U**pdate/**D**elete/**R**etrieve a user signon

Table 5. Function Types for the CRUD Matrix.

	E1	E2	E3	E4	E5
F1	CRUD				
F2		CRUD			
F3			CRUD		
F4				CRUD	

Table 5. The CRUD Matrix for the motif database.

MySQL Tools

In creating the database for the motifs, MySQL and MySQL Workbench were implemented.

MySQL Workbench was used in designing the ER diagram for the database as well as entering the test data and running queries against the data. MySQL proved to be a great tool in implementing a database. The tool makes running queries very simple. It has a great user interface that is also simple to use. MySQL was also great in allowing the user to add, edit and update the data seamlessly by implementing a "worksheet" in which the user can enter data without worrying about the syntax of an SQL command.

The only drawback of using MySQL Workbench was in the error messages. For example, if there is a limit on a datatype, it will set the value of the data being entered in a particular field. to the datatype's limit. An error may be encountered when trying to enter data in that particular field if that data is at the maximum value. Usually if this occurs, the system will generate an error message that states there is a duplicate message. Therefore, the error messages do not always reflect what need to be corrected.

Java GUI Design Rationale

In designing the GUI that would interface the motif database, there were for main tasks that the user should be able to accomplish. Those tasks are to view all the records in the motif database, the ability to add a record to the database as well as update a motif record in the database, the ability to search for a particular motif in the database and the capability to log in the database and access saved motifs. For the task of designing the programs and designing the GUIs that would display the various menus that would go along with the database, two basic tools were implemented. First, the Eclipse Java IDE tools was used to construct and test all of the Java programs. The second tool that was implemented was JForm Designer. This tool allowed for the easy construction of the GUIs that were used to access the motif database. Once the GUIs were constructed, the Java source code that was generated from the construction of the GUIs was manipulated to accomodate for the implementation of other Java programs that would interface with the MySQL database.

The first main task of the Java interface is to allow the user to view all the motif entries in the database. The two main Java programs that accomplish this task are called MotifMenu and ViewDB. The ViewDB program is called from the MotifMenu program and displays the current entries into the database.

The second main task is adding a new motif to the motif table in the database. This is accomplished by having MotifMenu call a java program called ADDrecord. This program is the Java GUI that will allow the user to enter in the data for the new motif. Once the new data is entered into the form and the button is pressed, the ADDrecord routine calls a program called AddMotif. This program connects to the motif database and adds the new information into the motif table in the database.

The next task that is available for the motif database is to update a motif entry in the database. This is achieved by having the MotifMenu program call a program called GetMotifID. This program is an GUI that grabs the desired ProSite ID entered into the GUI by the user. Next, the the ProSite ID is passed to a routine called UPDATErecord. Once this routine has the ProSite ID, it then calls a routine called Get1Rec which searches the motif database for the motif that is wanted by the user. Once this is done, the motif takes the data from Get1Rec and then populates the fields of the UPDATErecord GUI. Now, the user can edit any part of the motif records that is needed. Once the user is done, they can send the data to be written to the database. The UPDATErecord routine then passes the data to a routine called UpdateMotif. This routine connects to the motif database and then writes the new values out to the motif table in the database.

Another task that is done by the program is allowing the user to search for an individual record in the database. First, the MotifMenu program calls the GetMotifID routine to capture the ProSite ID from the user. Once this is done, the program passes the ProSite

ID to a routine called SearchMotifForm. This program takes the ID and uses it to call a routine called SearchMotif. SearchMotif connects to the database and gets the information that corresponds to the ProSite ID and passes the information back to SearchMotifForm. Finally, the SearchMotifForm routine displays the desired data in its. form.

The last major task that is handled by the Java program is allowing the user to sign on to the database and view their favorite motifs. This is accomplished by having the MotifMenu program call a routine called LoginScreen. This routine captures a user ID and password data and verifies them. Once the verification is done, the routine calls UsersMotifForm. This program takes the user ID and passes it to a program called FaveMotif. This routine connects to the motif database and gets the desired user ID and their corresponding motifs that are saved in the motifs_has_users table. This data is sent back to the UsersMotifForm routine and it is then displayed in the GUI.

The main goal of designing the GUI was to allow the user to execute basic tasks in managing the database without having to interface with the MySQL server. The user is able to add to the motif table. However, if the user wanted to add records to the other tables such as the gene table or Journal table, the user would have to access those tables through MySQL. However, MySQL workbench is an excellent way for a novice to access their database and manipulate data in their specific database. For a future implementation of this database program, it would be ideal for the users to be able to add new genes and new journal article information to the database.

The implementation section of this paper contains figures of the GUIs and the test data that was used for the project. Located in tables 6 and 7 are the names of the Java programs constructed to interact with the database along with a brief description of the routines.

Java GUI Program	Description
MotifMenu	main GUI for the program options to add, update, search, etc.
ADDrecord	GUI to accept new data to add a motif
UPDATErecord	GUI to accept new or old data to update a motif
GetMotifID	GUI to get ProSite ID to do an update or a search
LoginScreen	GUI to hold userID and password
UsersMotifForm	holds the users saved motif data
SearchMotifForm	holds the data from a motif search
ViewDB	hold the motif table data for all entries

Table 6. The Java GUI programs.

Java Program	Description
AddMotif	routine that adds a new record to the database
UpdateMotif	routine that updates a record in the motif database
FaveMotif	routine that gets the user's saved motif data
SearchMotif	routine that gets the motif of interest from the database
Get1Rec	routine that gets a selected motif from the database

Table 7. The Java programs that connect to the motif database.

Implementation

To test the prototype of the motif database program, test data was constructed to populate fields in the database. Also, a motif was chosen (ProSiteID PS00528, Ribosomal protein S4e signature) to test the add and update routines.

First, the MotifMenu GUI was tested along with the ViewDB program. These programs tested successfully. The MotifMenu was able transfer control to the corresponding programs successfully. Also, the ViewDB routine was able to display all the motif entries in the database.

Figure 2. The main menu of the motif database.

Next, the ADDrecord GUI along with the ADDmotif routine was tested. As input, the ProSite ID PS00528 was used. The data for the motif was keyed into the fields of the ADDrecord GUI. The routine was able to successfully pass control to AddMotif routine and the test record was successfully added to the database (See Figures 3 and 4).

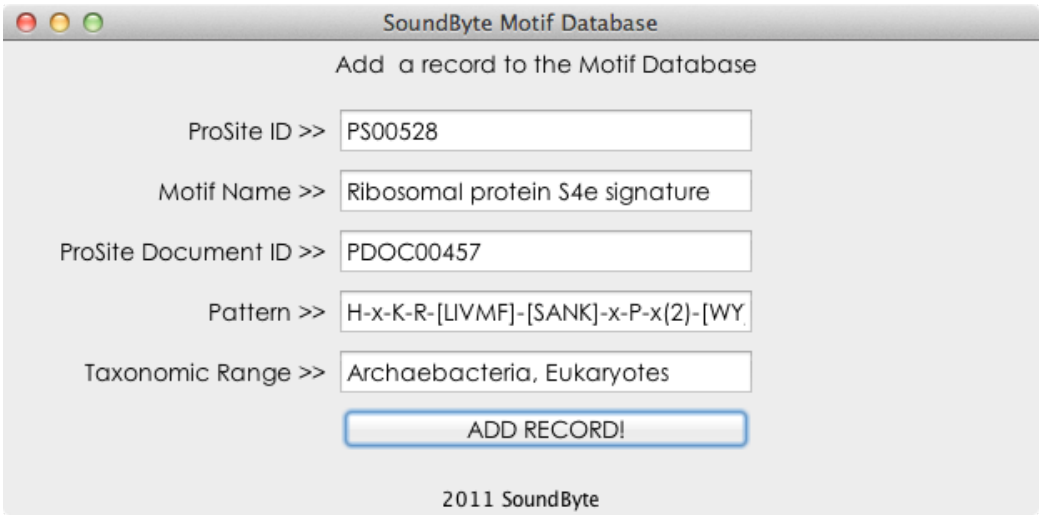
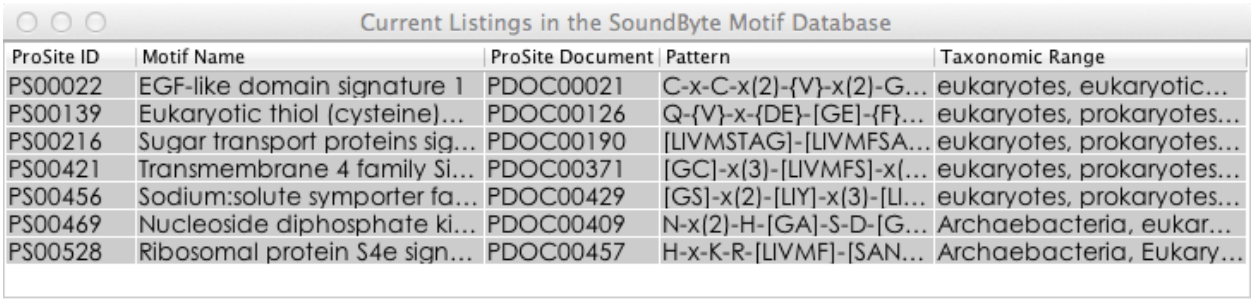


Figure 3. The ADDrecord GUI with the test record.

Next, the UPDATErecord GUI was tested along with the GetMotifID, Get1Rec and the UpdateMotif routines. The GetMotifID captured the desired ProSiteID and passed it to the UPDATErecord routine. After this is acc UPDATErecord routine passed the ProSiteID to Get1Rec. This routine connected to the database and passed the motif data back to UPDATErecord so that program could populate the data fields for UPDATErecord. The UPDATErecord routine was able to successfully pass the data from it collected in its for and pass it to the UpdateMotif routine. Finally, the UpdateMotif routine was able to successfully connect to the database and write the new information for the record out to the motif database (See Figures 5, 6 and 7).



ProSite ID	Motif Name	ProSite Document	Pattern	Taxonomic Range
PS00022	EGF-like domain signature 1	PDOC00021	C-x-C-x(2)-{V}-x(2)-G...	eukaryotes, eukaryotic...
PS00139	Eukaryotic thiol (cysteine)...	PDOC00126	Q-{V}-x-{DE}-[GE]-{F}...	eukaryotes, prokaryotes...
PS00216	Sugar transport proteins sig...	PDOC00190	[LIVMSTAG]-[LIVMFSA...	eukaryotes, prokaryotes...
PS00421	Transmembrane 4 family Si...	PDOC00371	[GC]-x(3)-[LIVMFS]-x(...	eukaryotes, prokaryotes...
PS00456	Sodium:solute symporter fa...	PDOC00429	[GS]-x(2)-[LIY]-x(3)-[LI...	eukaryotes, prokaryotes...
PS00469	Nucleoside diphosphate ki...	PDOC00409	N-x(2)-H-[GA]-S-D-[G...	Archaebacteria, eukar...
PS00528	Ribosomal protein S4e sign...	PDOC00457	H-x-K-R-[LIVMF]-[SAN...	Archaebacteria, Eukary...

Figure 4. The view option of the motif database.

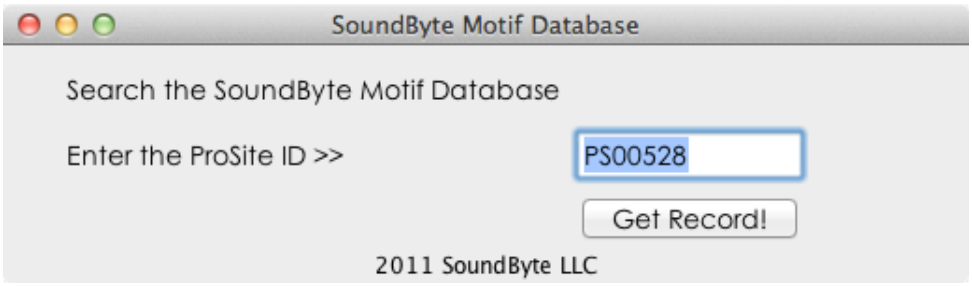


Figure 5. The GetMotifID GUI.

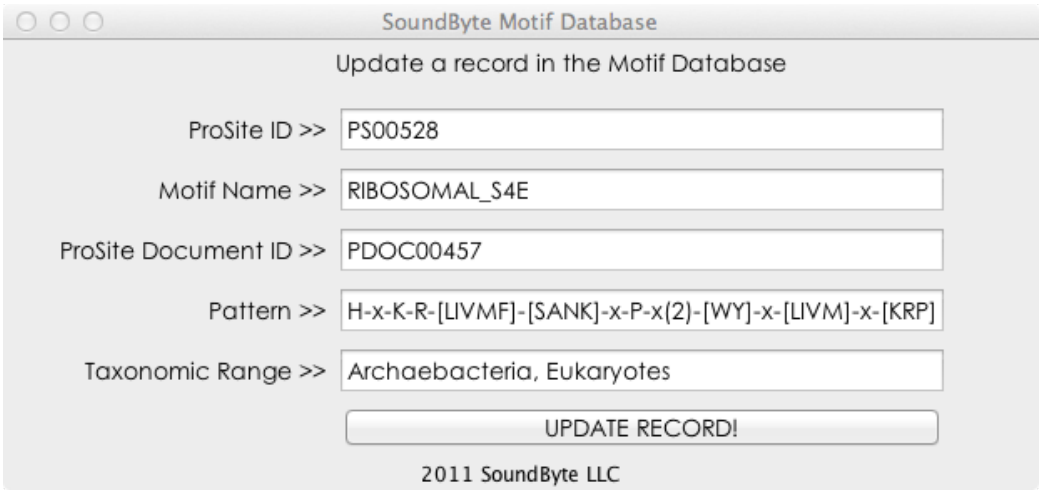


Figure 6. The UPDATErecord GUI.

proSiteID	motifName	pdocID	motifPattern	taxRange
PS00022	EGF-like domain sign...	PDOC00021	C-x-C-x(2)-{...	eukaryotes, e...
PS00139	Eukaryotic thiol (cyste...	PDOC00126	Q-{V}-x-{DE}-...	eukaryotes, pr...
PS00216	Sugar transport protei...	PDOC00190	[LIVMSTAG]-[...	eukaryotes, pr...
PS00421	Transmembrane 4 fa...	PDOC00371	[GC]-x(3)-[LIV...	eukaryotes, pr...
PS00456	Sodium:solute sympo...	PDOC00429	[GS]-x(2)-[LIY...	eukaryotes, pr...
PS00469	Nucleoside diphospha...	PDOC00409	N-x(2)-H-[GA...	Archaebacteri...
PS00528	RIBOSOMAL_S4E	PDOC00457	H-x-K-R-[LIV...	Archaebacteri...
NULL	NULL	NULL	NULL	NULL

Figure 7. A view of the updated record in the database.

After the Update aspect of the program was tested, the Search part of the program was examined. The Java programs that were used for this portion of the program was GetMotifID, SearchMotif and SearchMotifForm. The GetMotifID captured the ProSite ID that was entered by the user and passed it to the SearchMotifForm GUI. The GUI then passed the ProSite ID to SearchMotif. SearchMotif connected to the database and retrieved the data associated with the ProSite ID. That data was then displayed in the SearchMotifForm GUI (see Figures 5 and 8).

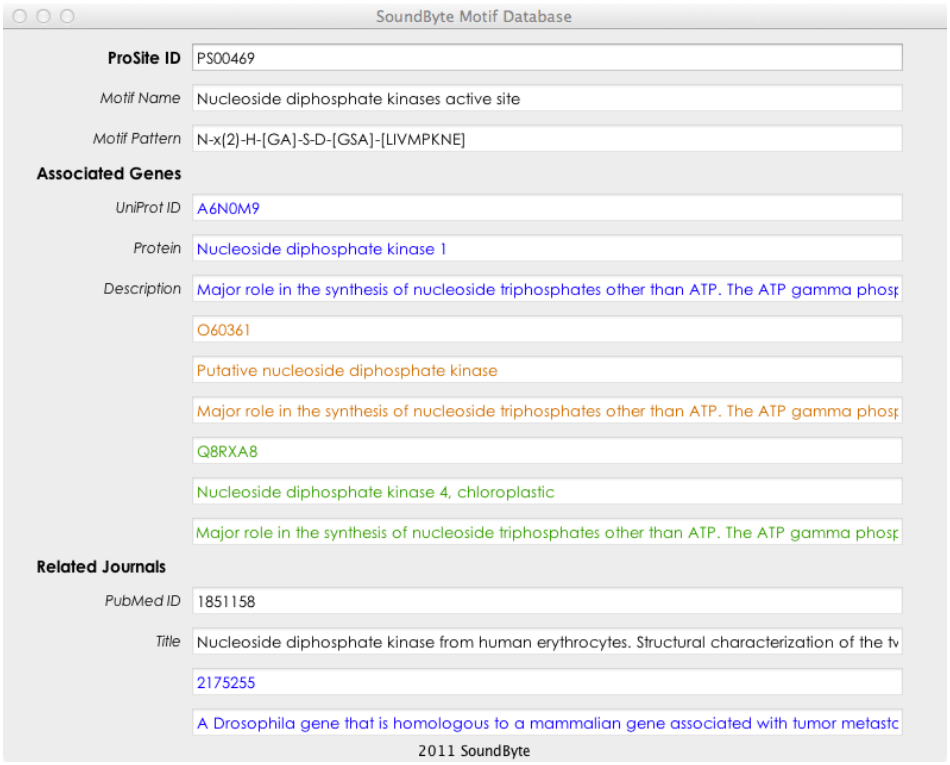


Figure 8. The SearchMotifForm GUI.

Finally, the last segment of the program that was tested was the LoginScreen GUI, the UsersMotifForm and the FaveMotif routines. The MotifMenu successfully called the LoginScreen to capture the user ID and password. The LoginScreen did test to check to see if the correct password was entered for the user. Once it was established that the correct password has been entered for the user, LoginScreen passed control and the user ID to the the FaveMotif routine. This program connected to the database and retrieved the user's favorite motif information, and then passed that data to the UsersMotifForm to be displayed (See Figures 9 and 10).

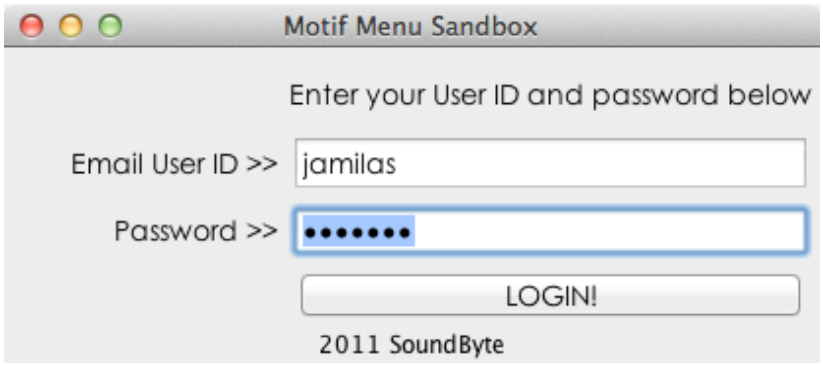


Figure 9. The LoginScreen GUI.

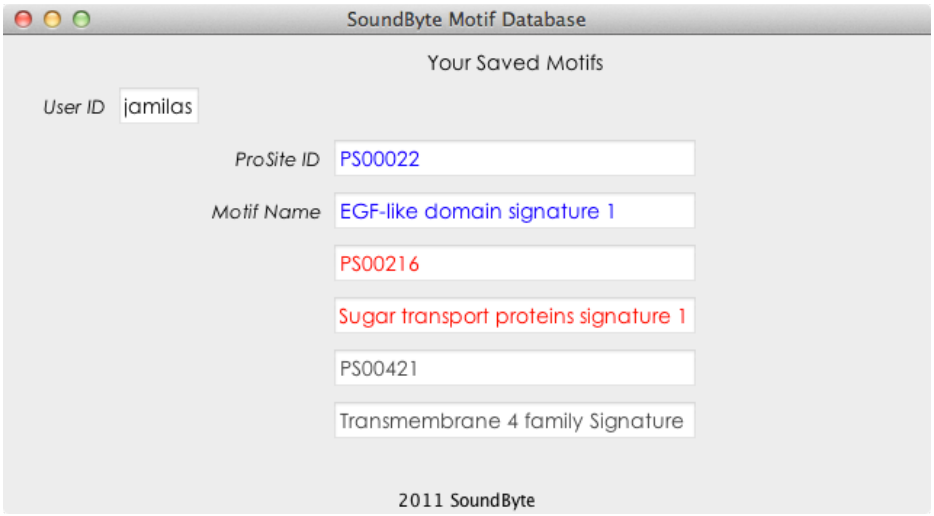


Figure 10. The UsersMotifForm GUI.

Some of the issues that affected the implementation of the program involved passing the the ProSite ID from the GetMotifID form to the UPDATErecord and the SearchMotifForm GUIs. The way those forms were originally constructed, I could not pass the the ProSite ID to them. I had to alter the way I was constructing the forms in the Java programs. Once I was able to do this, the programs worked successfully.

Conclusion

Overall, the implementation of the the motif database and the Java GUI for the database was successful. The data in the tables satisfies all of the data type constraints placed on them. Successful queries from the Java GUI as well as from MySQL workbench were able to be executed to display the various data contained in the database.

MySQL proved to be a valuable tool in designing and implementing the database portion of the project. The only problems encountered were some cryptic error messages during the design of the database and having to rebuild the database because of the omission of the Users table in the original database. Also, the Eclipse Java IDE along with the JForm designer tool were powerful tools that allowed me to construct GUIs and the associated programs that accompanied the motif database.

Some of the problems encountered during the design and implementation of the database and the Java GUI involved calling other classes and from certain classes. This was the especially the case with the GetMotifID routine. The way I had constructed the classes, the program was not able to pass a variable for the ProSite ID to the UPDATErecord GUI and the SearchMotifForm GUI. To test the program, I had to set up variables to emulate some of the instances that the programs were designed to handle. However, the underlying programs such as SearchMotif and UpdateMotif were successful when interfacing with the MySQL motif database. These underlying programs were able to update and retrieve information from the motif database. Also, I was not able to implement the hyperlinks for web locations such as NCBI and ProSite. This was due to the complexity of representing hyperlinks in Java programs and time constraints. Also, my knowledge of developing Java GUIs is limited. However, the other routines and their implementations proved to be successful.

Some of the aspects that could be implemented to make the database and the GUI more functional are data integrity and the GUI implementation. As of now, there is no data integrity involved when users add, update or sign on to the motif database. It would be advantageous to have the programs test for data integrity when users enter various data into the database. This would assist in making sure that the correct data formats as well as correct values are entered into the database. Also, it would probably be beneficial to have the GUI represented as a Java Server Page instead of a standalone program. This would allow for a better design of the GUI as well as easier construction for the servlets and underlying routines that would accompany the JSPs. Another refinement for the database is allowing for more fields in the database. For example, the Users table in the motif database could have fields such and email address. In the future this would allow you to hold users information and allow the administrator of the database to email users notifications any changes on the database as well as notifications on the availability of the database (maintenance issues). Finally, some other changes could include the implementation of the database on mobile media (tablet and smartphone applications), the ability to search the motif database using an amino acid sequence and ability to download specific data regarding a motif from ProSite. These changes would definitely add more functionality to the database.

The motif database project was a fantastic way to gain further education on the design and implementation of a database. I gained a tremendous amount of knowledge on designing and testing GUIs as well as additional knowledge on using and implementing MySQL tools to build a database. On a personal note, although there were some aspects of the project that I did not get to implement, overall, I am very pleased that I was able to get the project done. This is because I had no knowledge of constructing Java GUIs until the undertaking of this project. So, there was a steep learning curve to get it done. I wish I had known about Java Server Pages earlier. I think I would have been able to successfully implement all aspects of the project in a timely fashion. However, I did learn a lot of information constructing the database and the GUI program that will assist me in future endeavors.

Appendix

Test Data

proSiteID	motifName	pdocID	motifPattern	taxRange
PS00022	EGF-like domain sig...	PDOC00021	C-x-C-x(2)-{V}-x(2)-G-{C}...	eukaryotes, e...
PS00139	Eukaryotic thiol (cys...	PDOC00126	Q-{V}-x-{DE}-[GE]-{F}-C-[...	eukaryotes, pr...
PS00216	Sugar transport pro...	PDOC00190	[LIVMSTAG]-[LIVMFSAG]-{S...	eukaryotes, pr...
PS00421	Transmembrane 4 f...	PDOC00371	[GC]-x(3)-[LIVMFS]-x(2)-[...	eukaryotes, pr...
PS00456	Sodium:solute symp...	PDOC00429	[GS]-x(2)-[LIY]-x(3)-[LIVM...	eukaryotes, pr...
PS00469	Nucleoside diphosp...	PDOC00409	N-x(2)-H-[GA]-S-D-[GSA]...	Archaebacteri...
NULL	NULL	NULL	NULL	NULL

Table 6. Motif Test Data

uniProtID	proName	geneName	geneDesc	geneSeq	organism	motif_proSiteID
A1Z8N1	Facilitated trehalose tr...	Tret1-1	Low-capacity facilitativ...	"MSGRDNRGA...	Drosophila mela...	PS00216
A6N0M9	Nucleoside diphospha...	NDKR	Major role in the synthe...	"MEQSFIMIKP...	Oryza sativa sub...	PS00469
B5X316	tetraspanin-9	tspan9	Membrane; Mulit-pass...	MARGCLCCVK...	salmo salar (atla...	PS00421
O60361	Putative nucleoside di...	NME2P1	Major role in the synthe...	"MQCGLVGKII...	Homo sapiens (H...	PS00469
O75078	Disintegrin and metall...	ADAM11	Probable ligand for inte...	MRLLRRWAFA...	Homo sapiens (H...	PS00022
P0AE24	Arabinose-proton sy...	araE	Uptake of arabinose acr...	"MVTINTESAL...	Escherichia coli (...	PS00216
P11168	Solute carrier family 2...	SLC2A2	Facilitative glucose tran...	"MTEDKVTGTL...	Homo sapiens (H...	PS00216
P48747	Complement compon...	C9	Constituent of the mem...	MAASHSFAFV...	Oryctolagus cuni...	PS00022
P53791	Sodium/glucose cotra...	SLC5A1	Actively transports gluc...	"MDSSTWSPPA...	Ovis aries (Sheep)	PS00456
P53794	Sodium/myo-inositol...	SLC5A3	Prevents intracellular ac...	"MRAVLDTADI...	Homo sapiens (H...	PS00456
P87362	Bleomycin hydrolase	BLMH	The normal physiologic...	"MNAHGLSTE...	Gallus gallus (Ch...	PS00139
Q05094	Cysteine proteinase 2	CYS2	The cysteine proteinase...	"MATSRAALC...	Leishmania pifanoi	PS00139
Q1C0M8	Cation/acetate sympo...	actP	Transports acetate	"MKIRHWSALS...	Yersinia pestis b...	PS00456
Q4R7W6	tetraspanin-1	TSPAN1	Lysosome membrane;...	MQCFSFIKTIMI...	Macaca fascicula...	PS00421
Q5R6D1	Cathepsin B	CTSB	Thiol protease which is...	"MWQLWASLC...	Pongo abelii (Su...	PS00139
Q8RXA8	Nucleoside diphospha...	NDK4	Major role in the synthe...	"MRSQIYRSAT...	Spinacia olerace...	PS00469
Q9N0J9	CD81 protein	CD81	May play an important r...	MGVEGCTKCI...	Saguinus oedipu...	PS00421
Q9W6F9	Wnt inhibitory factor 1	wif1	Binds to WNT proteins...	MAFRTPAVQL...	Danio rerio (Zebr...	PS00022
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table 7. Gene Test Data

pubMedID	title	author	motif_proSiteID
1851158	Nucleoside diphos...	Gilles A.-M., Prese...	PS00469
1860863	Structure and me...	Levy S., Nguyen V....	PS00421
1965458	The Na+/pantothe...	Reizer J., Reizer A....	PS00456
2175255	A Drosophila gene...	Biggs J., Hersperge...	PS00469
2288911	The many faces of...	Davis C.G.	PS00022
3148320	Sequence homolog...	Dufour E.	PS00139
3543693	Mammalian and ba...	Maiden M.C.J., Dav...	PS00216
6334307	Vaccinia virus 19-...	Blomquist M.C., H...	PS00022
7678222	Epitope mapping o...	Tomlinson M.G., W...	PS00421
7845226	Families of cystein...	Rawlings N.D., Bar...	PS00139
8031825	A functional super...	Reizer J., Reizer A....	PS00456
8507645	Mammalian passiv...	Baldwin S.A.	PS00216
NULL	NULL	NULL	NULL

Table 8. Journals Test Data

users
darrelln
jamilas
kenyattab
NULL

Table 9. Users Test Data

	motif_proSiteID	users_users
▶	PS00022	jamilas
	PS00139	darrelln
	PS00216	jamilas
	PS00421	darrelln
	PS00421	jamilas
	PS00456	darrelln
	NULL	NULL

Table 10. Motif_has_users Test Data

Java Programs

MotifMenu.java

```
package appletPlay;


import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

/**
 * @author darrell nabors
 *
 * this is the main program (MotifMenu)...  all other routines get called from here...
 */
public class MotifMenu extends JPanel {


        static JFrame frame;

        RadioListener myListener = null;   // set listeners to do action
        ButtonListener goListener = null;

        String choice = null;       // choice is the variable that will hold the action command


        public MotifMenu() {
```

```
		initComponents();
	}

	private void initComponents() {
		// JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
		// Generated using JFormDesigner Evaluation license - darrell nabors
		//declare buttons for menu

		label1 = new JLabel();
		radioButton1 = new JRadioButton();
		radioButton2 = new JRadioButton();
		radioButton3 = new JRadioButton();
		radioButton4 = new JRadioButton();
		radioButton5 = new JRadioButton();
		button1 = new JButton();


		//======== this ========

		// JFormDesigner evaluation mark
		setBorder(new javax.swing.border.CompoundBorder(
			new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(50, 50, 50, 50),
				"2011 SoundByte", javax.swing.border.TitledBorder.CENTER,
				javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
				java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

		setLayout(new FormLayout(
			"10*(default, $lcgap), default",
			"11*(default, $lgap), default"));

		//---- label1 ----
		label1.setText("SoundByte Motif Database");
		label1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(label1, CC.xy(21, 1, CC.CENTER, CC.DEFAULT));

		//---- radioButton1 ----
		radioButton1.setText("View all Motif Records");
		radioButton1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		radioButton1.setActionCommand("VIEW");
//		radioButton1.setSelected(true);
		add(radioButton1, CC.xy(21, 7));

		//---- radioButton2 ----
		radioButton2.setText("Add Motif Record");
		radioButton2.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		radioButton2.setActionCommand("ADD");
		add(radioButton2, CC.xy(21, 9));

		//---- radioButton4 ---- (extra button needed)
		radioButton4.setText("Update Motif Record");
		radioButton4.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		radioButton4.setActionCommand("UPDATE");
		add(radioButton4, CC.xy(21, 11));

		//---- radioButton3 ----
```

```
            radioButton3.setText("Search Motif DB");
            radioButton3.setFont(new Font("Century Gothic", Font.PLAIN, 14));
            radioButton3.setActionCommand("SEARCH");
            add(radioButton3, CC.xy(21, 13));

            //---- radioButton4 ----
            radioButton5.setText("Login to Access Saved Motifs");
            radioButton5.setFont(new Font("Century Gothic", Font.PLAIN, 14));
            radioButton5.setActionCommand("LOGIN");
            add(radioButton5, CC.xy(21, 15));

            //---- button1 ----
            button1.setText("GO!");
            button1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
            button1.setActionCommand("GO");
            add(button1, CC.xy(21, 17));



              // group radio and action buttons

        ButtonGroup group = new ButtonGroup();
        group.add(radioButton1);
        group.add(radioButton2);
        group.add(radioButton3);
        group.add(radioButton4);
        group.add(radioButton5);
        group.add(button1);

     // register an action listener for the radio buttons

        myListener = new RadioListener();
        radioButton1.addActionListener(myListener);
        radioButton2.addActionListener(myListener);
        radioButton3.addActionListener(myListener);
        radioButton4.addActionListener(myListener);
        radioButton5.addActionListener(myListener);


    // register an action listener for the action button
        goListener = new ButtonListener();
        button1.addActionListener(goListener);




            // JFormDesigner - End of component initialization  //GEN-END:initComponents
}

// JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
// Generated using JFormDesigner Evaluation license - darrell nabors
private JLabel label1;
private JRadioButton radioButton1;
private JRadioButton radioButton2;
private JRadioButton radioButton3;
private JRadioButton radioButton4;
private JRadioButton radioButton5;
private JButton button1;
```

```
// JFormDesigner - End of variables declaration  //GEN-END:variables

// controls  the action of the radio buttons
   class RadioListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
                choice = e.getActionCommand();   // move the command from the last radio button to hold area
      }
    }


 // grabs the radio command and then calls associated routine
 class ButtonListener implements ActionListener {
              ButtonListener() {}

              public void actionPerformed(ActionEvent e) {


                if (e.getActionCommand().equals("GO"))
                {

                        if(choice.equals("VIEW"))
                        { ViewDB viewAll = new ViewDB();  viewAll.showViewform();}   // go to view routine

                        if(choice.equals("ADD"))
                        { ADDrecord addMotif = new ADDrecord();  addMotif.main(null);  }  //go to add routine

                        if(choice.equals("UPDATE"))
                        { GetMotifID getMotif = new GetMotifID();  getMotif.main(null);  }  //call update routine

                        if(choice.equals("SEARCH"))
                        { GetSearchID searchDB = new GetSearchID();  searchDB.main(null);  } //call search routine

                        if(choice.equals("LOGIN"))
                        { LoginScreen logon = new LoginScreen();  logon.main(null);  }//call login routine

                }

              }
             }




 //main program to display motifmenu
 public static void main(String s[]) {
       frame = new JFrame("Motif Menu Sandbox");
       frame.addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {System.exit(0);}
       });

       frame.getContentPane().add(new MotifMenu(), BorderLayout.CENTER);
       frame.pack();
       frame.setVisible(true);
   }
```

```
}
```

ViewDB.java

```
package appletPlay;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;

import java.sql.*;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;

import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;


import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

//  this routine connects to the motif database and then displays all motif listings
//  in the database
public class ViewDB
{


public static void showViewform ()
{

                JTable table = new JTable();   // initialize table

                        Connection conn = null;
        int columnCount;
        DefaultTableModel dataModel = new DefaultTableModel();
        String[] headers = {"ProSite ID","Motif Name","ProSite Document","Pattern","Taxonomic Range"};  // headers for
table


        try
        {

          //set variables 2 connect 2 database
          String userName = "root";
          String password = "";
          String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
          Class.forName ("com.mysql.jdbc.Driver").newInstance ();
```

```
  conn = DriverManager.getConnection (url, userName, password);

  Statement stmt = conn.createStatement();    //connect to the database

  String Query = "select * from motifDB.motif;";  // sql query to connect

  System.out.println ("Database connection established");


  ResultSet rs = stmt.executeQuery(Query);   //execute query and set result set
  ResultSetMetaData meta = rs.getMetaData();
  columnCount = meta.getColumnCount();       //get the number of columns


   dataModel.setColumnIdentifiers(headers);    //set the headers to the table datamodel

  while (rs.next())
  {
     Object[] rowData = new String[columnCount];  //prepare to grab row data

  for (int i = 1; i <= columnCount; i++)
  {  rowData[i-1] = rs.getString(i);
     //print line for verification
     System.out.println(rowData[i-1]);  }

     dataModel.addRow(rowData);   //add row data to the table

}
  table.setModel(dataModel);   //set the data model to the table

  table.setBackground(new Color(204, 204, 204));             //set the fonts and background of the table
                    table.setFont(new Font("Century Gothic", Font.PLAIN, 14));


// create the scroll pane and add the table to it
  JScrollPane scrollPane = new JScrollPane(table);
  JPanel panel = new JPanel();
  panel.setLayout(new BorderLayout(10, 10));

  //Add the scroll pane to this panel
  panel.add(scrollPane);

    JFrame f = new JFrame("Current Listings in the SoundByte Motif Database");  //declare the JFrame to use
    f.add(panel);  // add the panel to the frame
                      f.pack();
                      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                      f.setVisible(true);

    rs.close();  //close result set
    stmt.close();


}
catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin' else, bee!"); }
finally
{
  if (conn != null)
  {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
```

```
            catch (Exception e) { /* ignore close errors */ }
        }
       }

    }

}
```

ADDrecord.java

```
package appletPlay;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

import appletPlay.MotifMenu2.ButtonListener;

import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 *
 * this program displays the form to add a record to the motif database
 */
public class ADDrecord extends JPanel {

            static JFrame frame;                    // set frame to use

            ButtonListener addListener = null;          // declare action for button


            public ADDrecord() { initComponents(); }

            private void initComponents() {
                        // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
                        // Generated using JFormDesigner Evaluation license - darrell nabors
                        label6 = new JLabel();
                        prositeIDlabel = new JLabel();
                        prositeID = new JTextField();
                        motifNamelabel = new JLabel();
                        motifName = new JTextField();
                        prodocIDlabel = new JLabel();
                        prodocID = new JTextField();
                        patternlabel = new JLabel();
                        pattern = new JTextField();
                        taxrangelabel = new JLabel();
                        taxrange = new JTextField();
```

```
		addButton = new JButton();

		//======== this ========

		// JFormDesigner evaluation mark
		setBorder(new javax.swing.border.CompoundBorder(
			new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
				"2011 SoundByte", javax.swing.border.TitledBorder.CENTER,
				javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
				java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

		setLayout(new FormLayout(
			"6*(default, $lcgap), default",
			"8*(default, $lgap), default"));

		//---- label6 ----
		label6.setText("Add  a record to the Motif Database");
		label6.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(label6, CC.xy(13, 1));

		//---- prositeIDlabel ----
		prositeIDlabel.setText("ProSite ID >>");
		prositeIDlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(prositeIDlabel, CC.xy(11, 7, CC.RIGHT, CC.DEFAULT));
		prositeID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(prositeID, CC.xy(13, 7));

		//---- motifNamelabel ----
		motifNamelabel.setText("Motif Name >>");
		motifNamelabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(motifNamelabel, CC.xy(11, 9, CC.RIGHT, CC.DEFAULT));
		motifName.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(motifName, CC.xy(13, 9));

		//---- prodocIDlabel ----
		prodocIDlabel.setText("ProSite Document ID >>");
		prodocIDlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(prodocIDlabel, CC.xy(11, 11, CC.RIGHT, CC.DEFAULT));
		prodocID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(prodocID, CC.xy(13, 11));

		//---- patternlabel ----
		patternlabel.setText("Pattern >>");
		patternlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(patternlabel, CC.xy(11, 13, CC.RIGHT, CC.DEFAULT));
		pattern.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(pattern, CC.xy(13, 13));

		//---- taxrangelabel ----
		taxrangelabel.setText("Taxonomic Range >>");
		taxrangelabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(taxrangelabel, CC.xy(11, 15, CC.RIGHT, CC.DEFAULT));
		taxrange.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(taxrange, CC.xy(13, 15));
```

```
                //---- addButton ----
                addButton.setText("ADD RECORD!");
                addButton.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(addButton, CC.xy(13, 17));
                addButton.setActionCommand("ADD");


                //set listener for button and add that button to the panel
                addListener = new ButtonListener();
                addButton.addActionListener(addListener);



                // JFormDesigner - End of component initialization  //GEN-END:initComponents
        }

        // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
        // Generated using JFormDesigner Evaluation license - darrell nabors
        private JLabel label6;
        private JLabel prositeIDlabel;
        private JTextField prositeID;
        private JLabel motifNamelabel;
        private JTextField motifName;
        private JLabel prodocIDlabel;
        private JTextField prodocID;
        private JLabel patternlabel;
        private JTextField pattern;
        private JLabel taxrangelabel;
        private JTextField taxrange;
        private JButton addButton;
        // JFormDesigner - End of variables declaration  //GEN-END:variables

        // listener class to take action once button is depressed

         class ButtonListener implements ActionListener {
                    ButtonListener() {}

                    public void actionPerformed(ActionEvent e)
                    {
                      if (e.getActionCommand().equals("ADD"))
                      {
                       String prositeIDhold = prositeID.getText();  // all data from the field...
                       String motifNamehold = motifName.getText();
                       String prodocIDhold = prodocID.getText();
                       String patternhold = pattern.getText();
                       String taxrangehold = taxrange.getText();

                       AddMotif addnew = new AddMotif();
            addnew.addRecord(prositeIDhold, motifNamehold, prodocIDhold, patternhold, taxrangehold);    // pass all
data to addmotif routine...


                      }
                    }
                  }


        //main program to display the form
```

```
        public static void main(String s[])
        {
                Connection conn = null;

            frame = new JFrame("SoundByte Motif Database");
            frame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {System.exit(0);}
            });

            frame.getContentPane().add(new ADDrecord(), BorderLayout.CENTER);
            frame.pack();
            frame.setVisible(true);

    }



}
```

AddMotif.java

```
package appletPlay;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

//  this routine will connect to the database and add the new record
public class AddMotif {

        /**
         * @param args
         */


public static void addRecord (String motifID, String motifName,String pdocNum, String pattern, String taxRange)
{
                Connection conn = null;

                //print out data for verification
                System.out.println(motifID+"   "+motifName+"    "+pdocNum+"     "+pattern+"   "+taxRange);

                //set variables 2 connect 2 database
    try
    {
      String userName = "root";
      String password = "";
      String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
      Class.forName ("com.mysql.jdbc.Driver").newInstance ();
      conn = DriverManager.getConnection (url, userName, password);

      Statement stmt = conn.createStatement();

      //  sql statement to insert the data into the motif table
      String insRec = "INSERT INTO `motifDB`.`motif` (`proSiteID`, `motifName`, `pdocID`, `motifPattern`, `taxRange`)"
```

```
                    + "VALUES ('"+motifID+"', '"+motifName+"', '"+pdocNum+"', '"+pattern+"', '"+taxRange+"');";

      System.out.println ("Database connection established");

       stmt.execute(insRec);  //  executes the sql statement

       stmt.close();         // close statement

    }
    catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin' else"); }
    finally
    {
     if (conn != null)
     {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
         catch (Exception e) { /* ignore close errors */ }
     }
    }

        }

}
```

SearchMotifForm.java

```
package appletPlay;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.Connection;

import javax.swing.*;

import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 */
public class SearchMotifForm extends JPanel {

        static JFrame frame;                    // set frame to use

        public String [] motifData = new String[16];      // set array to hold data from searchmotif


        public SearchMotifForm(String motifID) {   initComponents(motifID);  }

        private void initComponents(String motifID) {

                SearchMotif get1record = new SearchMotif();       //declare routine to get record

                motifData = get1record.SearchRecord(motifID);       //   get data from searchmotif

                // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
                // Generated using JFormDesigner Evaluation license - darrell nabors
```

```
                // set data to the textfields
                label13 = new JLabel();
                motifIDnum = new JTextField(motifData[0]);
                label14 = new JLabel();
                motifName = new JTextField(motifData[1]);
                label15 = new JLabel();
                pattern = new JTextField(motifData[2]);
                label4 = new JLabel();
                label1 = new JLabel();
                gene1 = new JTextField(motifData[3]);
                label2 = new JLabel();
                proname1 = new JTextField(motifData[4]);
                label3 = new JLabel();
                genedes1 = new JTextField(motifData[5]);
                gene2 = new JTextField(motifData[6]);
                proname2 = new JTextField(motifData[7]);
                genedes2 = new JTextField(motifData[8]);
                gene3 = new JTextField(motifData[9]);
                proname3 = new JTextField(motifData[10]);
                genedes3 = new JTextField(motifData[11]);
                label8 = new JLabel();
                label5 = new JLabel();
                uniprot1 = new JTextField(motifData[12]);
                label6 = new JLabel();
                title1 = new JTextField(motifData[13]);
                uniprot2 = new JTextField(motifData[14]);
                title2 = new JTextField(motifData[15]);

                //======== this ========

                // JFormDesigner evaluation mark
                setBorder(new javax.swing.border.CompoundBorder(
                        new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
                                "2011 SoundByte", javax.swing.border.TitledBorder.CENTER,
                                javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
                                java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

                setLayout(new FormLayout(
                        "10*(default, $lcgap), default",
                        "18*(default, $lgap), default"));

                //---- label13 ----
                label13.setText("ProSite ID");
                label13.setFont(new Font("Century Gothic", Font.BOLD, 14));
                add(label13, CC.xy(9, 3, CC.RIGHT, CC.DEFAULT));

                //---- motifID ----
                motifIDnum.setFont(new Font("Century Gothic", Font.PLAIN, 13));
                motifIDnum.setEditable(false);
                add(motifIDnum, CC.xy(11, 3));

                //---- label14 ----
                label14.setText("Motif Name");
                label14.setFont(new Font("Century Gothic", Font.ITALIC, 12));
                add(label14, CC.xy(9, 5, CC.RIGHT, CC.DEFAULT));
```

```
//---- motifName ----
motifName.setFont(new Font("Century Gothic", Font.PLAIN, 13));
motifName.setEditable(false);
add(motifName, CC.xy(11, 5));

//---- label15 ----
label15.setText("Motif Pattern");
label15.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label15, CC.xy(9, 7, CC.RIGHT, CC.DEFAULT));

//---- pattern ----
pattern.setFont(new Font("Century Gothic", Font.PLAIN, 13));
pattern.setEditable(false);
add(pattern, CC.xy(11, 7));

//---- label4 ----
label4.setText("Associated Genes");
label4.setFont(new Font("Century Gothic", Font.BOLD, 14));
add(label4, CC.xy(9, 9));

//---- label1 ----
label1.setText("UniProt ID");
label1.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label1, CC.xy(9, 11, CC.RIGHT, CC.DEFAULT));

//---- gene1 ----
gene1.setForeground(Color.blue);
gene1.setFont(new Font("Century Gothic", Font.PLAIN, 13));
gene1.setEditable(false);
add(gene1, CC.xy(11, 11));

//---- label2 ----
label2.setText("Protein");
label2.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label2, CC.xy(9, 13, CC.RIGHT, CC.DEFAULT));

//---- proname1 ----
proname1.setForeground(Color.blue);
proname1.setFont(new Font("Century Gothic", Font.PLAIN, 13));
proname1.setEditable(false);
add(proname1, CC.xy(11, 13));

//---- label3 ----
label3.setText("Description");
label3.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label3, CC.xy(9, 15, CC.RIGHT, CC.DEFAULT));

//---- genedes1 ----
genedes1.setForeground(Color.blue);
genedes1.setFont(new Font("Century Gothic", Font.PLAIN, 13));
genedes1.setEditable(false);
add(genedes1, CC.xy(11, 15));

//---- gene2 ----
gene2.setFont(new Font("Century Gothic", Font.PLAIN, 13));
gene2.setForeground(new Color(204, 102, 0));
gene2.setEditable(false);
```

```
add(gene2, CC.xy(11, 17));

//---- proname2 ----
proname2.setFont(new Font("Century Gothic", Font.PLAIN, 13));
proname2.setForeground(new Color(204, 102, 0));
proname2.setEditable(false);
add(proname2, CC.xy(11, 19));

//---- genedes2 ----
genedes2.setFont(new Font("Century Gothic", Font.PLAIN, 13));
genedes2.setForeground(new Color(204, 102, 0));
genedes2.setEditable(false);
add(genedes2, CC.xy(11, 21));

//---- gene3 ----
gene3.setFont(new Font("Century Gothic", Font.PLAIN, 13));
gene3.setForeground(new Color(51, 153, 0));
gene3.setEditable(false);
add(gene3, CC.xy(11, 23));

//---- proname3 ----
proname3.setFont(new Font("Century Gothic", Font.PLAIN, 13));
proname3.setForeground(new Color(51, 153, 0));
proname3.setEditable(false);
add(proname3, CC.xy(11, 25));

//---- genedes3 ----
genedes3.setFont(new Font("Century Gothic", Font.PLAIN, 13));
genedes3.setForeground(new Color(51, 153, 0));
genedes3.setEditable(false);
add(genedes3, CC.xy(11, 27));

//---- label8 ----
label8.setText("Related Journals");
label8.setFont(new Font("Century Gothic", Font.BOLD, 14));
add(label8, CC.xy(9, 29));

//---- label5 ----
label5.setText("PubMed ID");
label5.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label5, CC.xy(9, 31, CC.RIGHT, CC.DEFAULT));

//---- uniprot1 ----
uniprot1.setEditable(false);
uniprot1.setFont(new Font("Century Gothic", Font.PLAIN, 13));
add(uniprot1, CC.xy(11, 31));

//---- label6 ----
label6.setText("Title");
label6.setFont(new Font("Century Gothic", Font.ITALIC, 12));
add(label6, CC.xy(9, 33, CC.RIGHT, CC.DEFAULT));

//---- title1 ----
title1.setEditable(false);
title1.setFont(new Font("Century Gothic", Font.PLAIN, 13));
add(title1, CC.xy(11, 33));

//---- uniprot2 ----
```

```
            uniprot2.setEditable(false);
            uniprot2.setFont(new Font("Century Gothic", Font.PLAIN, 13));
            uniprot2.setForeground(Color.blue);
            add(uniprot2, CC.xy(11, 35));

            //---- title2 ----
            title2.setEditable(false);
            title2.setFont(new Font("Century Gothic", Font.PLAIN, 13));
            title2.setForeground(Color.blue);
            add(title2, CC.xy(11, 37));


            // JFormDesigner - End of component initialization  //GEN-END:initComponents
    }

    // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
    // Generated using JFormDesigner Evaluation license - darrell nabors
    private JLabel label13;
    private JTextField motifIDnum;
    private JLabel label14;
    private JTextField motifName;
    private JLabel label15;
    private JTextField pattern;
    private JLabel label4;
    private JLabel label1;
    private JTextField gene1;
    private JLabel label2;
    private JTextField proname1;
    private JLabel label3;
    private JTextField genedes1;
    private JTextField gene2;
    private JTextField proname2;
    private JTextField genedes2;
    private JTextField gene3;
    private JTextField proname3;
    private JTextField genedes3;
    private JLabel label8;
    private JLabel label5;
    private JTextField uniprot1;
    private JLabel label6;
    private JTextField title1;
    private JTextField uniprot2;
    private JTextField title2;

    // JFormDesigner - End of variables declaration  //GEN-END:variables


    // main program to display form
    public static void showSearchForm(String motifID)
     {
frame = new JFrame("SoundByte Motif Database");
          frame.addWindowListener(new WindowAdapter() {
              public void windowClosing(WindowEvent e) {System.exit(0);}
          });

          frame.getContentPane().add(new SearchMotifForm(motifID), BorderLayout.CENTER);
          frame.pack();
          frame.setVisible(true);
```

```
            }


}
```

SearchMotif.java

```
package appletPlay;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


public class SearchMotif {

            /**
             * @param args
             *
             * this routine connects to the database and grabs the motif and associated data for a given prosite id
             */
//public static void main (String[] args)

public String[] SearchRecord (String motifID)
{
                        Connection conn = null;

                        String[] motifData = new String[16];  //array to hold the records found in the database
                        int k = 0;

                        //set variables 2 connect 2 database
      try
      {
        String userName = "root";
        String password = "";
        String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
        Class.forName ("com.mysql.jdbc.Driver").newInstance ();
        conn = DriverManager.getConnection (url, userName, password);

        Statement stmt = conn.createStatement();

        // search the database to get the appropriate motif, associated genes and journal article data
        // for the prosite id given

        String srch1 = "select proSiteID, motifName, motifPattern from motifDB.motif where proSiteID = \""+motifID+"\";";
        String srch2 = "select uniProtID, proName, geneDesc from motifDB.gene where motif_proSiteID = \""+motifID
+"\";";
        String srch3 = "select pubMedID, title from motifDB.journals where motif_proSiteID = \""+motifID+"\";";

          System.out.println ("Database connection established");


          ResultSet rs1 = stmt.executeQuery(srch1);
```

```
            while (rs1.next()) {for(k=0;k<3;k++) motifData[k]=rs1.getString(k+1);}     // get the motif data and move into array
            rs1.close();

            ResultSet rs2 = stmt.executeQuery(srch2);

            k=3;
            while (rs2.next())
            {motifData[k]=rs2.getString(1);                    // get the correct gene information and move into array
             motifData[k+1]=rs2.getString(2);
             motifData[k+2]=rs2.getString(3);
             k=k+3;}
             rs2.close();

            ResultSet rs3 = stmt.executeQuery(srch3);

            k=12;
            while (rs3.next())                                 //get the correct journal data and move into the array
            {motifData[k]=rs3.getString(1);
             motifData[k+1]=rs3.getString(2);
             k=k+2;}
            rs3.close();
            stmt.close();

        }
        catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin' else!"); }
        finally
        {
          if (conn != null)
          {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
              catch (Exception e) { /* ignore close errors */ }
          }
        }

        return motifData;   //send data back to the searchmotifform routine

            }

}
```

UPDATErecord.java

```
package appletPlay;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

import appletPlay.MotifMenu2.ButtonListener;
```

```
import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 *
 * this routine populates the fields with motif data so the user can update any field
 */
public class UPDATErecord extends JPanel {

static JFrame frame;                    // declare the frame for display

//public String searchID = null;
public String [] motifData = new String[5];       //  array to hold motif data from get1rec routine

ButtonListener addListener = null;

public UPDATErecord(String motifID) {   initComponents(motifID); }

private void initComponents(String motifID) {

                        Get1Rec darrell = new Get1Rec();



                        motifData = darrell.GetRec4U(motifID);

                        // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
                        // Generated using JFormDesigner Evaluation license - darrell nabors

                        //populate text fields with motif data
                        label6 = new JLabel();
                        prositeIDlabel = new JLabel();
                        prositeID = new JTextField(motifData[0]);
                        motifNamelabel = new JLabel();
                        motifName = new JTextField(motifData[1]);
                        prodocIDlabel = new JLabel();
                        prodocID = new JTextField(motifData[2]);
                        patternlabel = new JLabel();
                        pattern = new JTextField(motifData[3]);
                        taxrangelabel = new JLabel();
                        taxrange = new JTextField(motifData[4]);
                        addButton = new JButton();

                        //======== this ========

                        // JFormDesigner evaluation mark
                        setBorder(new javax.swing.border.CompoundBorder(
                                new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
                                        "2011 SoundByte LLC", javax.swing.border.TitledBorder.CENTER,
                                        javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
                                        java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

                        setLayout(new FormLayout(
                                "6*(default, $lcgap), default",
```

```
            "8*(default, $lgap), default"));

//---- label6 ----
label6.setText("Update a record in the Motif Database");
label6.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(label6, CC.xy(13, 1));

//---- prositeIDlabel ----
prositeIDlabel.setText("ProSite ID >>");
prositeIDlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(prositeIDlabel, CC.xy(11, 7, CC.RIGHT, CC.DEFAULT));
prositeID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(prositeID, CC.xy(13, 7));

//---- motifNamelabel ----
motifNamelabel.setText("Motif Name >>");
motifNamelabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(motifNamelabel, CC.xy(11, 9, CC.RIGHT, CC.DEFAULT));
motifName.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(motifName, CC.xy(13, 9));

//---- prodocIDlabel ----
prodocIDlabel.setText("ProSite Document ID >>");
prodocIDlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(prodocIDlabel, CC.xy(11, 11, CC.RIGHT, CC.DEFAULT));
prodocID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(prodocID, CC.xy(13, 11));

//---- patternlabel ----
patternlabel.setText("Pattern >>");
patternlabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(patternlabel, CC.xy(11, 13, CC.RIGHT, CC.DEFAULT));
pattern.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(pattern, CC.xy(13, 13));

//---- taxrangelabel ----
taxrangelabel.setText("Taxonomic Range >>");
taxrangelabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(taxrangelabel, CC.xy(11, 15, CC.RIGHT, CC.DEFAULT));
taxrange.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(taxrange, CC.xy(13, 15));

//---- addButton ----
addButton.setText("UPDATE RECORD!");
addButton.setFont(new Font("Century Gothic", Font.PLAIN, 14));
add(addButton, CC.xy(13, 17));
addButton.setActionCommand("UPDATE");


//declare the listener for the update button
// add the listener action to the button
addListener = new ButtonListener();
addButton.addActionListener(addListener);
// JFormDesigner - End of component initialization  //GEN-END:initComponents


//print statement for debug purposes
System.out.println("from iC, motifID = "+motifID);
```

```
        System.out.print("from iC = "+motifData[0]+"   "+motifData[1]+"   "+motifData[2]+"   "+motifData[3]+"
"+motifData[4]+"\n");

            }



            // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
            // Generated using JFormDesigner Evaluation license - darrell nabors
            private JLabel label6;
            private JLabel prositeIDlabel;
            private JTextField prositeID;
            private JLabel motifNamelabel;
            private JTextField motifName;
            private JLabel prodocIDlabel;
            private JTextField prodocID;
            private JLabel patternlabel;
            private JTextField pattern;
            private JLabel taxrangelabel;
            private JTextField taxrange;
            private JButton addButton;
            // JFormDesigner - End of variables declaration  //GEN-END:variables



    // class that processes the action from the update button
            class ButtonListener implements ActionListener {
                        ButtonListener() {}

                        public void actionPerformed(ActionEvent e)
                        {
                          if (e.getActionCommand().equals("UPDATE"))
                          {
//                                  System.out.println("from action, motifID = "+motifID);

                            String prositeIDhold = prositeID.getText();  //move the updated field to the hold area
                            String motifNamehold = motifName.getText();
                            String prodocIDhold = prodocID.getText();
                            String patternhold = pattern.getText();
                            String taxrangehold = taxrange.getText();

                            UpdateMotif updateDB = new UpdateMotif();  //  call the updatemotif routine to write the record to
the file
                updateDB.updateRecord(prositeIDhold, motifNamehold, prodocIDhold, patternhold, taxrangehold);


                          }
                        }
                      }



            //main program to display form
            public void showUpdateForm(String motifID)
            {
```

```
                frame = new JFrame("SoundByte Motif Database");
                frame.addWindowListener(new WindowAdapter() {
                    public void windowClosing(WindowEvent e) {System.exit(0);}
                });

                frame.getContentPane().add(new UPDATErecord(motifID), BorderLayout.CENTER);
                frame.pack();
                frame.setVisible(true);

 }



}
```

UpdateMotif.java

```
package appletPlay;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


public class UpdateMotif {

            /**
             * @param args
             *
             * this routine connects to the database and writes the updated information out to the motif database
             *
             */
//public static void main (String[] args)

public static void updateRecord (String motifID, String motifName,String pdocNum, String pattern, String taxRange)
{
                    Connection conn = null;

      // for debugging purposes
                    System.out.println(motifID+"   "+motifName+"    "+pdocNum+"     "+pattern+"   "+taxRange);

                    //set variables 2 connect 2 database
      try
      {
       String userName = "root";
       String password = "";
       String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
       Class.forName ("com.mysql.jdbc.Driver").newInstance ();
       conn = DriverManager.getConnection (url, userName, password);

       Statement stmt = conn.createStatement();

       //write each field out to the database

      String updt1 = "update motifDB.motif set proSiteID = '"+motifID+"' where proSiteID = \""+motifID+"\";";
```

```
        String updt2 = "update motifDB.motif set motifName = '"+motifName+"' where proSiteID = \""+motifID+"\";";

        String updt3 = "update motifDB.motif set pdocID = '"+pdocNum+"' where proSiteID = \""+motifID+"\";";

        String updt4 = "update motifDB.motif set motifPattern = '"+pattern+"' where proSiteID = \""+motifID+"\";";

        String updt5 = "update motifDB.motif set taxRange = '"+taxRange+"' where proSiteID = \""+motifID+"\";";


        System.out.println ("Database connection established");

          stmt.execute(updt1);     // executes update statements
          stmt.execute(updt2);
          stmt.execute(updt3);
          stmt.execute(updt4);
          stmt.execute(updt5);

          stmt.close();

      }
      catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin' else!"); }
      finally
      {
        if (conn != null)
        {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
            catch (Exception e) { /* ignore close errors */ }
        }
      }


          }

}
```

Get1Rec.java

```
package appletPlay;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


public class Get1Rec {

          /**
           * @param args
           *
           *   this routine retrieves the motif record from the database according to the prosite ID
           */


public String[] GetRec4U (String motifID)
{
                    Connection conn = null;
```

```
                String[] motifData = new String[5];   //array to hold the motif data
                int k = 0;


                System.out.println("get1rec4u....."+motifID);  //print statement to help debug

                //set variables 2 connect 2 database
                try
        {
          String userName = "root";
          String password = "";
          String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
          Class.forName ("com.mysql.jdbc.Driver").newInstance ();
          conn = DriverManager.getConnection (url, userName, password);

          Statement stmt = conn.createStatement();

          String srch1 = "select * from motifDB.motif where proSiteID = \""+motifID+"\";";  // grab the record from the motif
database

          System.out.println ("Database connection established");

          ResultSet rs1 = stmt.executeQuery(srch1);
          System.out.println();

          while (rs1.next())
          {   for(k = 0; k<motifData.length; k++)              //store the record into the array
                motifData[k] = rs1.getString(k+1);    }

           System.out.println();
           rs1.close();
           stmt.close();

          //print statement for debug
           System.out.print("getrec4u...."+motifData[0]+"   "+motifData[1]+"   "+motifData[2]+"   "+motifData[3]+"
"+motifData[4]+"\n");



        }
        catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin' else!"); }
        finally
        {
          if (conn != null)
          {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
              catch (Exception e) { /* ignore close errors */ }
          }
        }

       return motifData;   //return the data to the updaterecord GUI

             }

}
```

GetMotifID.java

```
package appletPlay;

import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.Connection;

import javax.swing.*;

import appletPlay.ADDrecord.ButtonListener;

import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 *
 *   this routine grabs the prosite ID for the motif and passes it to a search program
 */
public class GetMotifID extends JPanel {

static JFrame frame;                //declare the frame for display
public String searchID = null;

        ButtonListener getRecListener = null;     // declare the listener for the action button


        public GetMotifID() { initComponents(); }

        private void initComponents() {
                // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
                // Generated using JFormDesigner Evaluation license - darrell nabors
                titlelabel = new JLabel();
                getReclabel = new JLabel();
                getUpdateRec = new JTextField();
                getRecButton = new JButton();

                //======== this ========

                // JFormDesigner evaluation mark
                setBorder(new javax.swing.border.CompoundBorder(
                        new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
                                "2011 SoundByte LLC", javax.swing.border.TitledBorder.CENTER,
                                javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
                                java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

                setLayout(new FormLayout(
                        "6*(default, $lcgap), default",
                        "5*(default, $lgap), default"));
```

```
                //---- titlelabel ----
                titlelabel.setText("Search the SoundByte Motif Database");
                titlelabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(titlelabel, CC.xy(11, 5, CC.RIGHT, CC.DEFAULT));

                //---- getReclabel ----
                getReclabel.setText("Enter the ProSite ID >>");
                getReclabel.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(getReclabel, CC.xy(11, 9));

                //---- getUpdateRec ----
                getUpdateRec.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(getUpdateRec, CC.xy(13, 9));

                //---- getRecButton ----
                getRecButton.setText("Get Record!");
                getRecButton.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(getRecButton, CC.xy(13, 11));
getRecButton.setActionCommand("GET");


// declare the actionlistener for the button
                getRecListener = new ButtonListener();
                getRecButton.addActionListener(getRecListener);

                // JFormDesigner - End of component initialization  //GEN-END:initComponents
    }

    // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
    // Generated using JFormDesigner Evaluation license - darrell nabors
    private JLabel titlelabel;
    private JLabel getReclabel;
    private JTextField getUpdateRec;
    private JButton getRecButton;
    // JFormDesigner - End of variables declaration  //GEN-END:variables


    //class to take action for the action button
    class ButtonListener implements ActionListener {
                ButtonListener() {}

                public void actionPerformed(ActionEvent e)
                {
                  if (e.getActionCommand().equals("GET"))
                  {
                   searchID = getUpdateRec.getText();

                    UPDATErecord updateDB = new UPDATErecord(searchID);  // call the updaterecord GUI
          updateDB.showUpdateForm(searchID);


                  }
                 }
               }


    //main program to display the getmotifID form
```

```
        public static void main(String[] args)
        {
                Connection conn = null;

            frame = new JFrame("SoundByte Motif Database");
            frame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {System.exit(0);}
            });

            frame.getContentPane().add(new GetMotifID(), BorderLayout.CENTER);
            frame.pack();
            frame.setVisible(true);

//              return   searchID;

 }



}
```

LoginScreen.java

```
package appletPlay;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.*;



import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 *
 * this routine captures the userID and password and verifies that the userID and password match. it also passes the
userID to the usersmotifform routine
 */
public class LoginScreen extends JPanel
{

        static JFrame frame;                            //declare frame for display
        ButtonListener logon = new ButtonListener();            //declare action for logon button
        String [] passWordz = {"wolverine","sunburn","duluth"};    //the passwords of the users
        String [] userz     = {"darrelln","jamilas","kenyattab"};    // the userIDs
        String passchek = null;                         //hold area for password
        String userchek =
null;                                                                                   // hold area for userID

        public LoginScreen() { initComponents(); }
```

```
	private void initComponents() {
		// JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
		// Generated using JFormDesigner Evaluation license - darrell nabors
		label1 = new JLabel();
		label2 = new JLabel();
		userID = new JTextField();
		label3 = new JLabel();
		passWord = new JPasswordField();
		logonButton = new JButton();

		//======== this ========

		// JFormDesigner evaluation mark
		setBorder(new javax.swing.border.CompoundBorder(
			new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
				"2011 SoundByte", javax.swing.border.TitledBorder.CENTER,
				javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
				java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

		setLayout(new FormLayout(
			"6*(default, $lcgap), default",
			"6*(default, $lgap), default"));

		//---- label1 ----
		label1.setText("Enter your User ID and password below");
		label1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(label1, CC.xy(13, 5));

		//---- label2 ----
		label2.setText("Email User ID >>");
		label2.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(label2, CC.xy(11, 9, CC.RIGHT, CC.DEFAULT));

		//---- userID ----
		userID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(userID, CC.xy(13, 9));

		//---- label3 ----
		label3.setText("Password >>");
		label3.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		add(label3, CC.xy(11, 11, CC.RIGHT, CC.DEFAULT));
		add(passWord, CC.xy(13, 11));

		//---- logonButton ----
		logonButton.setText("LOGIN!");
		logonButton.setFont(new Font("Century Gothic", Font.PLAIN, 14));
		logonButton.setActionCommand("LOGON");
		add(logonButton, CC.xy(13, 13));

	    logonButton.addActionListener(logon);  //  add listener action to logon button

		// JFormDesigner - End of component initialization  //GEN-END:initComponents
	}

	// JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
```

```
        // Generated using JFormDesigner Evaluation license - darrell nabors
        private JLabel label1;
        private JLabel label2;
        private JTextField userID;
        private JLabel label3;
        private JPasswordField passWord;
        private JButton logonButton;
        // JFormDesigner - End of variables declaration  //GEN-END:variables


        // class to process action on logon button
        class ButtonListener implements ActionListener {
                ButtonListener() {}

                public void actionPerformed(ActionEvent e) {
                        if (e.getActionCommand().equals("LOGON"))
                   {
                        passchek = passWord.getText();  //move password and userID to hold area
                        userchek = userID.getText();

                          for(int k = 0; k < 3; k++)
                          {
                                  if(userchek.equals(userz[k]))
                                  {
                                          if(passchek.equals(passWordz[k])) { UsersMotifForm getfav = new
UsersMotifForm(userchek);  getfav.showFaveForm(userchek);  }  //check to see if userID and password match

                                          else {  LoginScreen goBack = new LoginScreen();   goBack.main(null); }  //
go back to beginning of program

                                  }  // this bracket is for the if statement

                          }   //this bracket ends the for loop
                   }  //   ths bracket is for the main if statement

                  }
                 }

        //main program to display logon form
         public static void main(String s[]) {
               frame = new JFrame("Motif Menu Sandbox");
               frame.addWindowListener(new WindowAdapter() {
                  public void windowClosing(WindowEvent e) {System.exit(0);}
               });

               frame.getContentPane().add(new LoginScreen(), BorderLayout.CENTER);
               frame.pack();
               frame.setVisible(true);
           }



}
```

UsersMotifForm.java

```
package appletPlay;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.*;

import com.jgoodies.forms.factories.*;
import com.jgoodies.forms.layout.*;

/**
 * @author darrell nabors
 *
 * this routine displays the data for the users favorite motifs
 *
 */
public class UsersMotifForm extends JPanel {

    static JFrame frame;        // set frame for display

            public String [] motifData = new String[9];     //array to hold the data from the favemotif routine



            public UsersMotifForm(String userchek) {  initComponents(userchek);  }

            private void initComponents(String userchek) {

        FaveMotif get1record = new FaveMotif();

                        motifData = get1record.getFave(userchek);       //  get the fave motif data according to the user name



                        // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
                        // Generated using JFormDesigner Evaluation license - darrell nabors

                        //populate fields with the favorite motif data
                        label1 = new JLabel();
                        label3 = new JLabel();
                        userID = new JTextField(motifData[0]);
                        label2 = new JLabel();
                        label4 = new JLabel();
                        motifID1 = new JTextField(motifData[1]);
                        label5 = new JLabel();
                        motifName1 = new JTextField(motifData[2]);
                        motifID2 = new JTextField(motifData[4]);
                        motifName2 = new JTextField(motifData[5]);
                        motifID3 = new JTextField(motifData[7]);
                        motifName3 = new JTextField(motifData[8]);

                        //======== this ========

                        // JFormDesigner evaluation mark
                        setBorder(new javax.swing.border.CompoundBorder(
```

```
                        new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
                                "2011 SoundByte", javax.swing.border.TitledBorder.CENTER,
                                javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog",
java.awt.Font.PLAIN, 12),
                                java.awt.Color.black), getBorder())); addPropertyChangeListener(new
java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e)
{if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

                setLayout(new FormLayout(
                        "8*(default, $lcgap), default",
                        "8*(default, $lgap), default"));

                //---- label1 ----
                label1.setText("Your Saved Motifs");
                label1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                add(label1, CC.xy(15, 3, CC.CENTER, CC.DEFAULT));

                //---- label3 ----
                label3.setText("User ID");
                label3.setFont(new Font("Century Gothic", Font.ITALIC, 13));
                add(label3, CC.xy(9, 5, CC.RIGHT, CC.DEFAULT));

                //---- userID ----
                userID.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                userID.setEditable(false);
                add(userID, CC.xy(11, 5));
                add(label2, CC.xy(15, 5));

                //---- label4 ----
                label4.setText("ProSite ID");
                label4.setFont(new Font("Century Gothic", Font.ITALIC, 13));
                add(label4, CC.xy(13, 7, CC.RIGHT, CC.DEFAULT));

                //---- motifID1 ----
                motifID1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                motifID1.setForeground(Color.blue);
                motifID1.setEditable(false);
                add(motifID1, CC.xy(15, 7));

                //---- label5 ----
                label5.setText("Motif Name");
                label5.setFont(new Font("Century Gothic", Font.ITALIC, 13));
                add(label5, CC.xy(13, 9, CC.RIGHT, CC.DEFAULT));

                //---- motifName1 ----
                motifName1.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                motifName1.setForeground(Color.blue);
                motifName1.setEditable(false);
                add(motifName1, CC.xy(15, 9));

                //---- motifID2 ----
                motifID2.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                motifID2.setForeground(Color.red);
                motifID2.setEditable(false);
                add(motifID2, CC.xy(15, 11));

                //---- motifName2 ----
                motifName2.setFont(new Font("Century Gothic", Font.PLAIN, 14));
```

```
                motifName2.setForeground(Color.red);
                motifName2.setEditable(false);
                add(motifName2, CC.xy(15, 13));

                //---- motifID3 ----
                motifID3.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                motifID3.setForeground(Color.darkGray);
                motifID3.setEditable(false);
                add(motifID3, CC.xy(15, 15));

                //---- motifName3 ----
                motifName3.setFont(new Font("Century Gothic", Font.PLAIN, 14));
                motifName3.setForeground(Color.darkGray);
                motifName3.setEditable(false);
                add(motifName3, CC.xy(15, 17));
                // JFormDesigner - End of component initialization  //GEN-END:initComponents
        }

        // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
        // Generated using JFormDesigner Evaluation license - darrell nabors
        private JLabel label1;
        private JLabel label3;
        private JTextField userID;
        private JLabel label2;
        private JLabel label4;
        private JTextField motifID1;
        private JLabel label5;
        private JTextField motifName1;
        private JTextField motifID2;
        private JTextField motifName2;
        private JTextField motifID3;
        private JTextField motifName3;
        // JFormDesigner - End of variables declaration  //GEN-END:variables


        // main program to display the form
        public  void showFaveForm(String usercheck)
         {
    frame = new JFrame("SoundByte Motif Database");
            frame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {System.exit(0);}
            });

            frame.getContentPane().add(new UsersMotifForm(usercheck), BorderLayout.CENTER);
            frame.pack();
            frame.setVisible(true);

         }


}
```

FaveMotif.java

```
package appletPlay;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


//  this routine connects to the database and retrieves the users favorite motif listings


public class FaveMotif {

            /**
             * @param args
             */


public String[] getFave(String userID)
{
                        Connection conn = null;


                        String[] motifData = new String[9];   //declare array to hold data from query
                        int k = 0;


                        //set variables 2 connect 2 database
       try
       {
         String userName = "root";
         String password = "";
         String url = "jdbc:mysql://127.0.0.1:3306/motifDB";
         Class.forName ("com.mysql.jdbc.Driver").newInstance ();
         conn = DriverManager.getConnection (url, userName, password);

         Statement stmt = conn.createStatement();


         // query to pull data from motif and users tables
         String favSrch = "select users_users, proSiteID, motifName from motifDB.motif JOIN motifDB.motif_has_users on
proSiteID = motif_proSiteID where users_users = \""+userID+"\";";


           System.out.println ("Database connection established");

           ResultSet rs2 = stmt.executeQuery(favSrch);  //run query


           k=0;
           while (rs2.next())                    // store data into the array
           {motifData[k]=rs2.getString(1);
            motifData[k+1]=rs2.getString(2);
            motifData[k+2]=rs2.getString(3);
            k=k+3;}
            rs2.close();
```

```
        }
        catch (Exception e) {System.err.println ("Cannot connect to database server!...do somethin'!"); }
        finally
        {
          if (conn != null)
          {   try  { conn.close ();  System.out.println ("Database connection terminated"); }
              catch (Exception e) { /* ignore close errors */ }
          }
        }

      return motifData;   // pass array back to the usersmotifform routine

            }

}
```

References

1. R. Elmasri, S. B. Navathe,“Fundamentals of Database Systems”, Sixth Edition, Addison-Wesley, 2009

2. “E-R Diagram for Online Bookstore”, http://www.docstoc.com/docs/4199320/E-R-Diagram-for-Online-Bookstore

3. “CRUD Matrices”, http://www.saintmarys.edu/~psmith/417act14.html

4. “A Simple Data Dictionary”, http://www.bcarter.com/tip039.htm

5. “bookstoreEnglishPic_2.jpg”, http://www.4shared.com/photo/1cRz2E9N/bookstoreEnglishPic_2.html

6. “MySQL 5.1 Reference Manual”, http://dev.mysql.com/doc/refman/5.1/en/

7. “FormDev - JFormDesigner - Java/Swing GUI Designer”, http://www.formdev.com/