ALGONQUIN COLLEGE

CST2335 GRAPHICAL INTERFACE PROGRAMMING

Week 7

Android SQLite

Topics

- Sql using ROOM
- Creating a Data Access Object (DAO)
- Using AlertDialog and Snackbar

Using ROOM Database in Android

- Previously Android used SQLiteDatabase and SQLiteOpenHelper classes for creating databases and performing CRUD operations.
- Room is a library for making it easy for running SQL commands
- It is not included in projects so we have to modify the **app/build.gradle** file

Setting up ROOM in Android

- Add these lines of code to the **dependencies { }** section of **app/build.gradle** file and press the '**Sync Now**' button afterwards

```
def room_version = "2.4.2"
implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
```

Create data storage objects

- Open ChatMessage.java and add the following code

```
@Entity
public class ChatMessage {

    @ColumnInfo(name="message")
    protected String message;

    @ColumnInfo(name="TimeSent")
    protected String timeSent;

    @ColumnInfo(name="SendOrReceive")
    protected int sendOrReceive;
}
```

```
    @PrimaryKey
    public int id;
```

Create data storage objects

- Add the **@Entity** annotation above the class to mark this as something that can go into a database
- Add **@ColumnInfo(name="xxxx")** to specify that this variable will go into a database column named xxxx
- Add a **@PrimaryKey public int id** field

Creating a Data Access Object (DAO)

- We need a DAO to handle inserting, updating, creating and deleting operations. To do this we create an interface and add **@DAO** annotation above the code below

```
public interface ChatMessageDAO {

    @Insert
    void insertMessage(ChatMessage m);

    @Query("Select * from ChatMessage")
    List<ChatMessage> getAllMessages();

    @Delete
    void deleteMessage(ChatMessage m);
}
```

Creating a Database on disk of the phone or tablet

- Create an abstract class called **MessageDatabase.java** that extends **RoomDatabase**
- The code below tells Room that this Database class is meant for storing ChatMessage objects, and uses the ChatMessageDAO class for querying data

```
@Database(entities = {ChatMessage.class}, version=1)
public abstract class MessageDatabase  extends RoomDatabase {
  public abstract ChatMessageDAO cmDAO();
}
```

Open a database

- Use the following code to open a database in **ChatRoom** class

```
MessageDatabase db = Room.databaseBuilder(getApplicationContext(), MessageDatabase.class, "database-name").build();
ChatMessageDAO mDAO = db.cmDAO();
List<ChatMessage> allMessages = mDAO.getAllMessages();
```

Setting an onClick listener for rows

- When a row is clicked in our MyRowHolder constructor we want to be able to show an alert window asking if you want to delete the row

```
class MyRowHolder extends RecyclerView.ViewHolder {
    TextView messageText;
    TextView timeText;

    public MyRowHolder(@NonNull View itemView) {
        super(itemView);

        itemView.setOnClickListener(clk ->{

            int position = getAbsoluteAdapterPosition();

        });

        messageText = itemView.findViewById(R.id.messageText);
        timeText = itemView.findViewById(R.id.timeText);
    }
}
```

Creating an AlertDialog

- Start by creating the alert dialog object inside the onClick of MyRowHolder constructor

```
AlertDialog.Builder builder = new AlertDialog.Builder( ChatRoom.this );
```

Creating an AlertDialog

- Start by creating the alert dialog object inside the onClick of MyRowHolder constructor

```
AlertDialog.Builder builder = new AlertDialog.Builder( ChatRoom.this );
```

Creating an AlertDialog - Set the message

- Do this by calling builder.setMessage()

```
builder.setMessage("Do you want to delete the message: " + messageText.getText());
```

Creating an AlertDialog - Set the title

- Do this by calling builder.setTitle()

```
builder.setTitle("Question:");
```

Creating an AlertDialog - Set the action buttons

- Do this by calling **builder.setPositiveButton()** and **builder.setNegativeButton()** respectively

```
builder.setNegativeButton( text: "No", (dialog, cl) -> { });
builder.setPositiveButton( text: "Yes", (dialog, cl) -> { });
```

Creating an AlertDialog - Code positive button

- Clicking the Yes button should remove a message from a row and delete it from the database.
- It should afterwards update the Adapter object

```
builder.setPositiveButton( text: "Yes", (dialog, cl)->{
    ChatMessage m = messages.get(position);
     mDAO.deleteMessage( m );
     messages.remove(position);
     adt.notifyItemRemoved(position);
});
```

Creating an AlertDialog - Show the alert dialog

- We finally show the alert dialog by chaining the following methods to our alert dialog object

```
builder.create().show();
```

Using a Snackbar

- It is similar to a Toast in the sense that, it can show a message for LENGTH_SHORT or LENGTH_LONG amount of time.

Snackbar.make(View v, CharSequence text, int duration).show();

Using a Snackbar

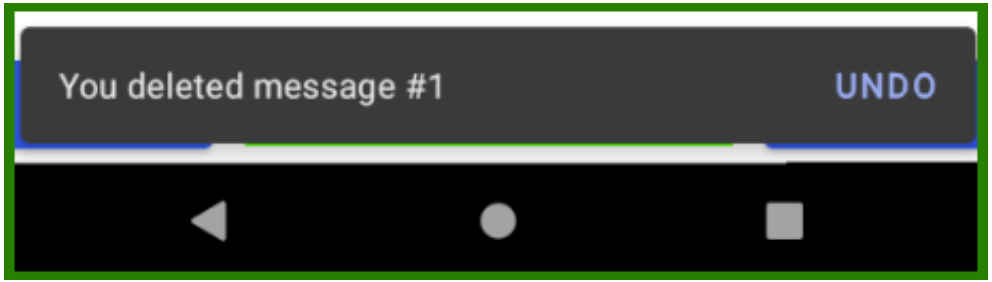
```
private class MyRowViews extends RecyclerView.ViewHolder{

    TextView messageText;
    TextView timeText;

    public MyRowViews(View itemView) {
        super(itemView);

        itemView.setOnClickListener( click -> {

            int position = getAbsoluteAdapterPosition();

            MyRowViews newRow = adt.onCreateViewHolder( parent: null, adt.getItemViewType(position));


            AlertDialog.Builder builder = new AlertDialog.Builder( context: ChatRoom.this );
            builder.setMessage( "Do you want to delete the message: " + messageText.getText())
                .setTitle("Question:")
                .setNegativeButton( text: "No", (dialog, cl)->{ })
                .setPositiveButton( text: "Yes", (dialog, cl) ->{

                    messages.remove(position);
                    adt.notifyItemRemoved(position);

                    Snackbar.make(messageText,  text: "You deleted message #"+ position, Snackbar.LENGTH_LONG).show();
            }).create().show();
        });
```

Using a Snackbar - optional button

- A Snackbar has an optional button, like the AlertDialog. On the Snackbar, call **setAction()** before the .show() function:

```
Snackbar.make(messageText,  text: "You deleted message #" + position, Snackbar.LENGTH_LONG)
        .setAction( text: "Undo", clk -> { })
        .show();
```

- It achieves the following effect:



Using a Snackbar - Code the Undo button

- With the following code, removedMessage stores the message before it's removed from the ArrayList so when you click undo it gets reinserted

```
.setPositiveButton( text: "Yes", (dialog, cl) -> {

    ChatMessage removedMessage = messages.get(position);
    messages.remove(position);
    adt.notifyItemRemoved(position);

    Snackbar.make(messageText,  text: "You deleted message #" + position, Snackbar.LENGTH_LONG)
            .setAction( text: "Undo", clk -> {

                messages.add(position, removedMessage);
                adt.notifyItemInserted(position);

            })
            .show();

})
```