UI Element – TimePicker

1. Create a new Android Application Project with the following attributes:
    a. Application Name: UITimePicker
    b. Icon: Clock clipart
    c. Activity Name: TimePickerActivity
    d. Layout Name: main
2. Create the xml file:
    a. Change the background of the screen to an image by following these steps:
        1. Right-click res folder then choose New>Folder, folder name is raw.
        2. Drag any image(jpg or png) from your computer into the raw folder.
        3. Click main.xml. Within the LinearLayout tag, type: android:background = "@raw/"
        4. Press ctrl+spacebar, the filename of your image should appear, then press enter.
    b. With the XML code below, drag UI elements needed in the app:

```
<TextView
    android:id="@+id/timeDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="24sp" />

<Button
    android:id="@+id/pickTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Change the time"
    android:textSize="24sp" />
```

    c. Save main.xml.

3. Write the following java source code in TimePickerActivity:
    a. This app uses the Calendar class in the utilities package of the standard edition. To implement the Calendar class, import java.util.Calendar.

b. Declare global variables to be used in various methods. The second part of the code is handling the TimePicker's event.

```
public class TimePickerActivity extends Activity {

    private TextView mTimeDisplay;
    private Button mPickTime;

    private int mHour;
    private int mMinute;

    static final int TIME_DIALOG_ID = 0;

    // the callback received when the user "sets" the time in the dialog
    private TimePickerDialog.OnTimeSetListener mTimeSetListener = new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            mHour = hourOfDay;
            mMinute = minute;
            updateDisplay();
        }
    };
```

c. The onCreate() method captures View elements from the XML file, handles click events of the button, gets the current time using the Calendar class and invokes the updateDisplay() method.

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // capture our View elements
    mTimeDisplay = (TextView) findViewById(R.id.timeDisplay);
    mPickTime = (Button) findViewById(R.id.pickTime);

    // add a click listener to the button
    mPickTime.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            showDialog(TIME_DIALOG_ID);
        }
    });

    // get the current time
    final Calendar c = Calendar.getInstance();
    mHour = c.get(Calendar.HOUR_OF_DAY);
    mMinute = c.get(Calendar.MINUTE);

    // display the current date
    updateDisplay();
}
```

d. User-defined methods: updateDisplay(), pad() and onCreateDialog() are created to provide an accurate TimePicker. The method updateDisplay() displays the modified time in the textview; the method pad() displays the string value of hour and minutes; and the method onCreateDialog() displays a TimePicker in a dialog box.

```
// updates the time we display in the TextView
private void updateDisplay() {
    mTimeDisplay.setText(new StringBuilder().append(pad(mHour)).append(":")
            .append(pad(mMinute)));
}

private static String pad(int c) {
    if (c >= 10)
        return String.valueOf(c);
    else
        return "0" + String.valueOf(c);
}

@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case TIME_DIALOG_ID:
        return new TimePickerDialog(this, mTimeSetListener, mHour, mMinute,
                false);
    }
    return null;
}
```

4. Save and run the program.