UI Element – DatePicker

1. Create a new Android Application Project with the following attributes:
    a. Application Name: UIDatePicker
    b. Icon: Calendar clipart
    c. Activity Name: DatePickerActivity
    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/dateDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:textColor=""
    android:textSize="24sp" />

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

    Provide the HTML code of color PINK to TextView's textColor and color BLUE to Button's background color.

    c. Save main.xml.

3. Write the following java source code in DatePickerActivity:

a. Like the TimePicker UI element, DatePicker uses the Calendar class in the utilities package of the standard edition either. 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 DatePicker's event.

```
private TextView mDateDisplay;
private Button mPickDate;
private int mYear;
private int mMonth;
private int mDay;

static final int DATE_DIALOG_ID = 0;

  // the callback received when the user "sets" the date in the dialog
private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener() {

            public void onDateSet(DatePicker view, int year,
                                  int monthOfYear, int dayOfMonth) {
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateDisplay();
            }
        };
```

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

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // capture our View elements
    mDateDisplay = (TextView) findViewById(R.id.dateDisplay);
    mPickDate = (Button) findViewById(R.id.pickDate);

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

    // get the current date
    final Calendar c = Calendar.getInstance();
    mYear = c.get(Calendar.YEAR);
    mMonth = c.get(Calendar.MONTH);
    mDay = c.get(Calendar.DAY_OF_MONTH);

    // display the current date (this method is below)
    updateDisplay();
}
```

d. User-defined methods: updateDisplay() and onCreateDialog() are created to provide an accurate DatePicker. The method updateDisplay() displays the modified date in the textview; and the method onCreateDialog() displays a DatePicker in a dialog box.

```
 // updates the date in the TextView
private void updateDisplay() {
    mDateDisplay.setText(
        new StringBuilder()
                // Month is 0 based so add 1
                .append(mMonth + 1).append("-")
                .append(mDay).append("-")
                .append(mYear).append(" "));
}

@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case DATE_DIALOG_ID:
        return new DatePickerDialog(this,
                    mDateSetListener,
                    mYear, mMonth, mDay);
    }
    return null;
}
```

4. Save and run the program.