

Level/Subject : **D3/Programming 6**
Topic : SQLite Database in Android apps

Week : 5
Activity : Dialogs in Android apps using Android Studio
Alocated time : 120 mins labs
Deliverables : Project folder
Due date : end of session

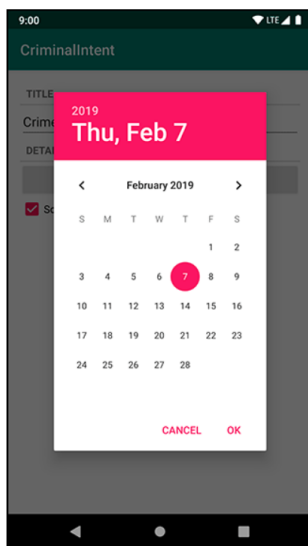
Competency :

Student expected to be able to create Android apps with Dialogs using Android Studio IDE.

Example Practice Task:

Create Android apps with Dialogs using Android Studio.

1. Dialogs demand attention and input from the user. They are useful for presenting a choice or important information. In this practice, you will add a dialog in which users can change the date of a crime. Pressing the date button in `CrimeFragment` will present this dialog

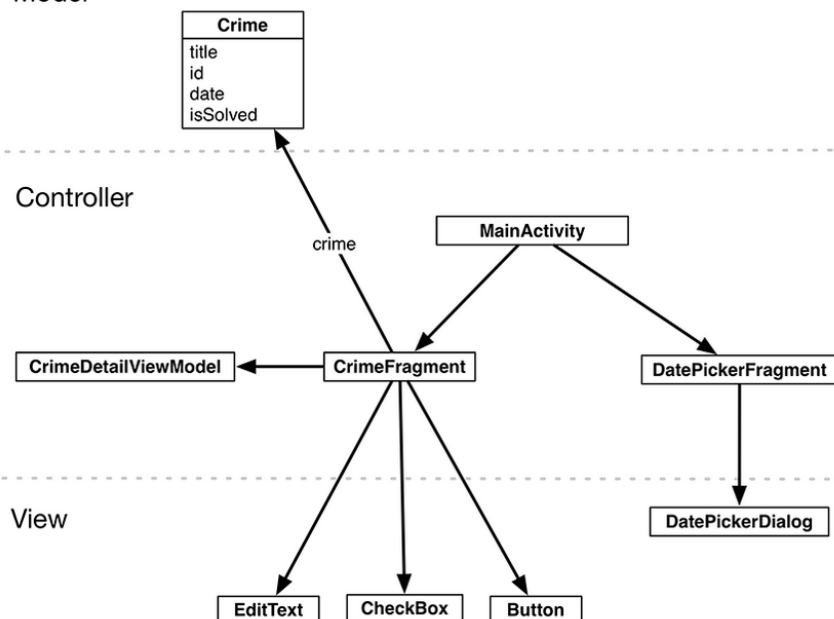


2. The dialog above is an instance of `DatePickerDialog`, a subclass of `AlertDialog`. `DatePickerDialog` displays a date selection prompt to the user and provides a listener interface you can implement to capture the selection. For creating more custom dialogs, `AlertDialog` is the all-purpose `Dialog` subclass that you will use most often.

Creating a DialogFragment

- When displaying a `DatePickerDialog`, it is a good idea to wrap it in an instance of `DialogFragment`, a subclass of `Fragment`. It is possible to display a `DatePickerDialog` without a `DialogFragment`, but it is not recommended. Having the `DatePickerDialog` managed by the `FragmentManager` gives you more options for presenting the dialog.
- In addition, a bare `DatePickerDialog` will vanish if the device is rotated. If the `DatePickerDialog` is wrapped in a fragment, then the dialog will be re-created and put back onscreen after rotation.
- For `CriminalIntent`, you are going to create a `DialogFragment` subclass named `DatePickerFragment`. Within `DatePickerFragment`, you will create and configure an instance of `DatePickerDialog`. `DatePickerFragment` will be hosted by `MainActivity`.

Model



View

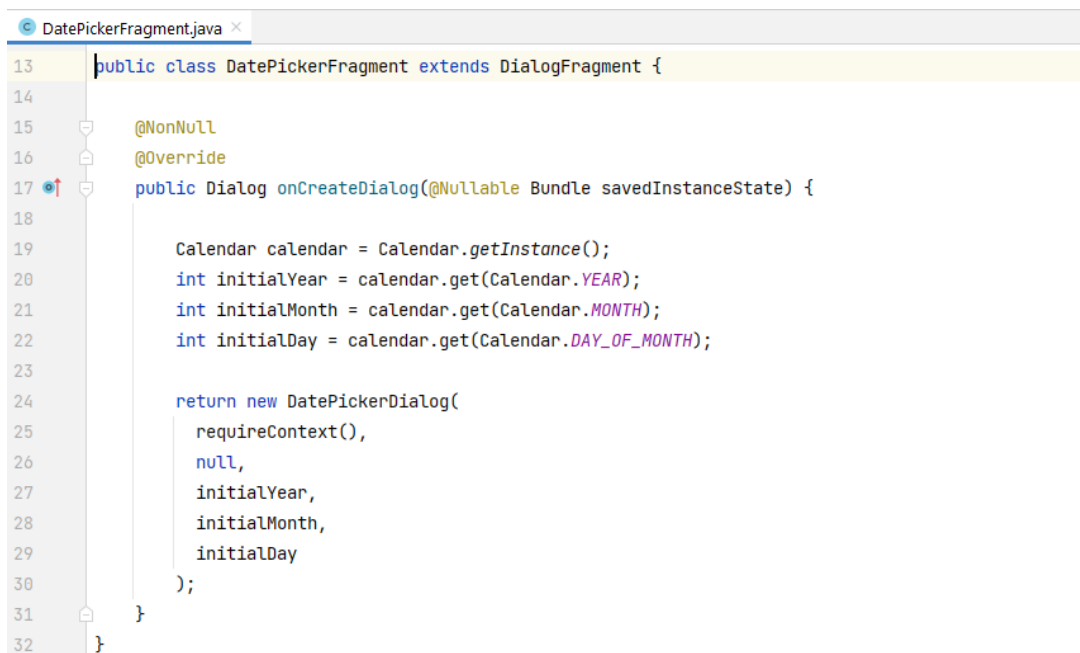
- Your first tasks are:
 - creating the `DatePickerFragment` class
 - building a `DatePickerFragment`
 - getting the dialog onscreen via the `FragmentManager`
- Later in this practice, you will pass the necessary data between `CrimeFragment` and `DatePickerFragment`.

8. Create a new class named `DatePickerFragment` and make its superclass `DialogFragment`. Be sure to choose the Jetpack version of `DialogFragment`: `androidx.fragment.app.DialogFragment`.

9. `DialogFragment` includes the following method:

```
public Dialog onCreateDialog(Bundle savedInstanceState)
```

10. The `FragmentManager` of the hosting activity calls this method as part of putting the `DialogFragment` onscreen.
11. In `DatePickerFragment.java`, add an implementation of `onCreateDialog(Bundle)` that builds a `DatePickerDialog` initialized with the current date

A screenshot of an IDE window titled 'DatePickerFragment.java'. The code shows a class 'DatePickerFragment' extending 'DialogFragment'. It has an '@Override' annotation and a 'public Dialog onCreateDialog(@Nullable Bundle savedInstanceState)' method. Inside the method, it gets a 'Calendar' instance, retrieves the current year, month, and day, and then returns a new 'DatePickerDialog' instance initialized with these values and the current context.

```
13 public class DatePickerFragment extends DialogFragment {
14
15     @NonNull
16     @Override
17     public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
18
19         Calendar calendar = Calendar.getInstance();
20         int initialYear = calendar.get(Calendar.YEAR);
21         int initialMonth = calendar.get(Calendar.MONTH);
22         int initialDay = calendar.get(Calendar.DAY_OF_MONTH);
23
24         return new DatePickerDialog(
25             requireContext(),
26             null,
27             initialYear,
28             initialMonth,
29             initialDay
30         );
31     }
32 }
```

12. The `DatePickerDialog` constructor takes in several parameters. The first is a context object, which is required to access the necessary resources for the view. The second parameter is for the date listener, which you will add later in this practice. The last three parameters are the year, month, and day that the date picker should be initialized to. Until you know the date of the crime, you can just initialize it to the current date.

Showing a DialogFragment

13. Like all fragments, instances of `DialogFragment` are managed by the `FragmentManager` of the hosting activity.
14. To get a `DialogFragment` added to the `FragmentManager` and put onscreen, you can call either of the following methods on the fragment instance:

```
public void show(FragmentManager manager, String tag)
public void show(FragmentTransaction transaction, String tag)
```

15. The string parameter uniquely identifies the `DialogFragment` in the `FragmentManager`'s list. Whether you use the `FragmentManager` or `FragmentTransaction` version is up to you. If you pass in a `FragmentTransaction`, you are responsible for creating and committing that transaction. If you pass in a `FragmentManager`, a transaction will automatically be created and committed for you.
16. Here, you will pass in a `FragmentManager`.
17. In `CrimeFragment`, add a constant for the `DatePickerFragment`'s tag.
18. Then, in `onCreateView(...)`, remove the code that disables the date button. Set a `View.OnClickListener` that shows a `DatePickerFragment` when the date button is pressed in `onCreateView()`.

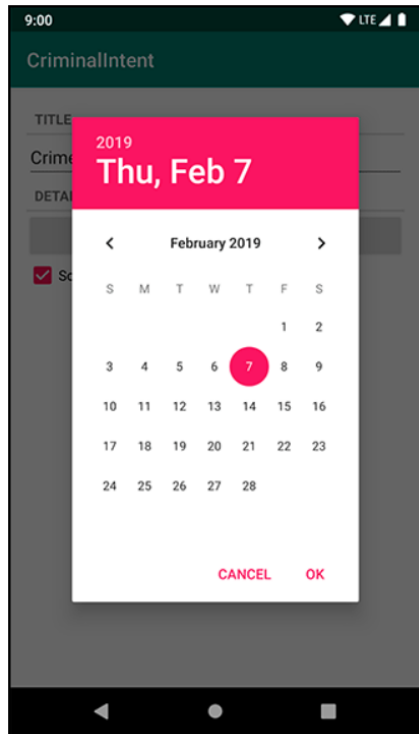
```

CrimeFragment.java x
23 public class CrimeFragment extends Fragment {
24     private static final String ARG_CRIME_ID = "crime_id";
25     private static final String TAG = "CrimeFragment";
26     private static final String REQUEST_DATE = "DialogDate";
27
...
87
88 @Override
89 public View onCreateView(LayoutInflater inflater, ViewGroup container,
90     Bundle savedInstanceState) {
91     View v = inflater.inflate(R.layout.fragment_crime, container, false);
92     mTitleField = v.findViewById(R.id.crime_title);
93     mTitleField.addTextChangedListener(new TextWatcher() {...});
109
110     mDateButton = v.findViewById(R.id.crime_date);
111     mDateButton.setText(mCrime.getDate().toString());
112     //mDateButton.setEnabled(false);
113     mDateButton.setOnClickListener(new View.OnClickListener() {
114         @Override
115         public void onClick(View view) {
116             FragmentManager manager = getChildFragmentManager();
117             DatePickerFragment dialog = new DatePickerFragment();
118             dialog.show(manager, REQUEST_DATE);
119         }
120     });
121
122     mSolvedCheckBox = v.findViewById(R.id.crime_solved);
123     mSolvedCheckBox.setOnCheckedChangeListener(
124         new CompoundButton.OnCheckedChangeListener() {
125             @Override
126             public void onCheckedChanged(CompoundButton buttonView,
127                 boolean isChecked) {
128                 mCrime.setSolved(isChecked);
129             }
130         });
131
132     return v;
133 }

```

19. DialogFragment's show(FragmentManager, String) requires a non-null value for the fragment manager argument. The Fragment.fragmentManager variable is nullable, so you cannot pass it directly to show(...). Instead, you use Fragment's getChildFragmentManager() method, whose return type is a non-null FragmentManager. If the fragment's fragmentManager variable is null when Fragment.getChildFragmentManager() is called, the method will throw an IllegalStateException stating that the fragment is not currently associated with a fragment manager.

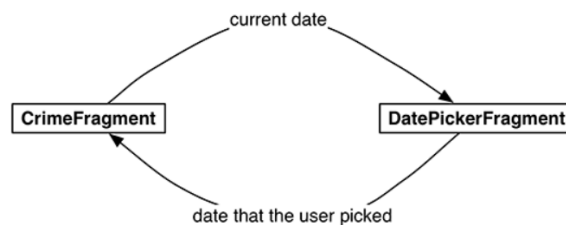
20. Run CriminalIntent and press the date button to see the dialog



21. Your dialog is onscreen and looks good. In the next section, you will wire it up to present the `Crime`'s date and allow the user to change it.

Passing Data Between Two Fragments

22. You have passed data between two activities using intent extras, passed data between a fragment and an activity using a callbacks interface, and passed data from an activity to a fragment using fragment arguments. Now you need to pass data between two fragments that are hosted by the same activity – `CrimeFragment` and `DatePickerFragment`



23. To get the `Crime`'s date to `DatePickerFragment`, you are going to write a `newInstance(Date)` method and make the `Date` an argument on the fragment.
24. To get the new date back to the `CrimeFragment` so that it can update the model layer and its own view, you will declare a callbacks interface method in `DatePickerFragment` that accepts the new date parameter, as shown



Passing data to DatePickerFragment

25. To get data into your DatePickerFragment, you are going to stash the date in DatePickerFragment's arguments bundle, where the DatePickerFragment can access it.
26. Creating and setting fragment arguments is typically done in a newInstance (...) method, as you saw before. In DatePickerFragment.java, add a newInstance(Date, String) method.

```

12 public class DatePickerFragment extends DialogFragment {
13
14     private static final String ARG_DATE = "date";
15     private static final String ARG_REQUEST_CODE = "requestCode";
16
17     @Override
18     public static DatePickerFragment newInstance(Date date, String requestCode){
19         Bundle args = new Bundle();
20         args.putSerializable(ARG_DATE, date);
21         args.putString(ARG_REQUEST_CODE, requestCode);
22         DatePickerFragment fragment = new DatePickerFragment();
23         fragment.setArguments(args);
24         return fragment;
25     }
26
27     @Override
28     public Dialog onCreateDialog(Bundle savedInstanceState) {
  
```

27. In CrimeFragment, remove the call to the DatePickerFragment constructor and replace it with a call to DatePickerFragment.newInstance(Date, String).

```

110 public View onCreateView(LayoutInflater inflater, ViewGroup container,
111                          Bundle savedInstanceState) {
112     View v = inflater.inflate(R.layout.fragment_crime, container, attachToRoot: false);
113
114     mTitleField = v.findViewById(R.id.crime_title);
115     mTitleField.addTextChangedListener(new TextWatcher() {...});
116
117     mDateButton = v.findViewById(R.id.crime_date);
118     mDateButton.setText(mCrime.getDate().toString());
119     //mDateButton.setEnabled(false);
120     mDateButton.setOnClickListener(new View.OnClickListener() {
121         @Override
122         public void onClick(View v) {
123             FragmentManager manager = getChildFragmentManager();
124             //DatePickerFragment dialog = new DatePickerFragment();
125             DatePickerFragment dialog = DatePickerFragment.newInstance(mCrime.getDate(),
126                             REQUEST_DATE);
127             dialog.show(manager, REQUEST_DATE);
128         }
129     });
130
131     mSolvedCheckBox = v.findViewById(R.id.crime_solved);
132     mSolvedCheckBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
133         @Override
134         public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
135             mCrime.setSolved(isChecked);
136         }
137     });
138
139     return v;
140 }

```

28. DatePickerFragment needs to initialize the DatePickerDialog using the information held in the Date. However, initializing the DatePickerDialog requires Ints for the month, day, and year. Date is more of a timestamp and cannot provide Ints like this directly.
29. To get the Ints you need, you provide the Date to the Calendar object. Then you can retrieve the required information from the Calendar.
30. In onCreateDialog(Bundle), get the Date from the arguments and use it and the Calendar to initialize the DatePickerDialog.


```

c DatePickerFragment.java x
26      @Override
27      public Dialog onCreateDialog(Bundle savedInstanceState) {
28          Date date = (Date) getArguments().getSerializable(ARG_DATE);
29
30          Calendar calendar = Calendar.getInstance();
31          calendar.setTime(date);
32          int initialYear = calendar.get(Calendar.YEAR);
33          int initialMonth = calendar.get(Calendar.MONTH);
34          int initialDay = calendar.get(Calendar.DAY_OF_MONTH);
35
36          return new DatePickerDialog(
37              requireContext(),
38              null,
39              initialYear,
40              initialMonth,
41              initialDay);
42      }

```

31. Now CrimeFragment is successfully telling DatePickerFragment what date to show. You can run CriminalIntent and make sure that the correct date of the crime is showing in the Dialog.

Returning data to CrimeFragment

32. To have CrimeFragment receive the date back from DatePickerFragment, you need a way to keep track of the relationship between the two fragments.
33. With activities, you call `startActivityForResult(...)`, and the `ActivityManager` keeps track of the parent-child activity relationship. When the child activity dies, the `ActivityManager` knows which activity should receive the result.

Setting a FragmentResultListener

34. You can create a similar connection by making CrimeFragment the target fragment of DatePickerFragment. This connection is automatically re-established after both CrimeFragment and DatePickerFragment are destroyed and re-created by the OS. To create this relationship, you implement the following FragmentResultListener interface.

```
CrimeFragment.java
112
113     mDateButton = v.findViewById(R.id.crime_date);
114     mDateButton.setText(mCrime.getDate().toString());
115     //mDateButton.setEnabled(false);
116     mDateButton.setOnClickListener(new View.OnClickListener() {
117         @Override
118         public void onClick(View v) {
119             FragmentManager manager = getChildFragmentManager();
120             manager.setFragmentResultListener(
121                 REQUEST_DATE,
122                 getViewLifecycleOwner(),
123                 new FragmentResultListener() {
124                     @Override
125                     public void onFragmentResult(String requestKey, Bundle result) {
126                         //handling result
127                     }
128                 }
129             );
130             //DatePickerFragment dialog = new DatePickerFragment();
131             DatePickerFragment dialog = DatePickerFragment.newInstance(
132                 mCrime.getDate(), REQUEST_DATE
133             );
134             dialog.show(manager, REQUEST_DATE);
135         }
136     });
```

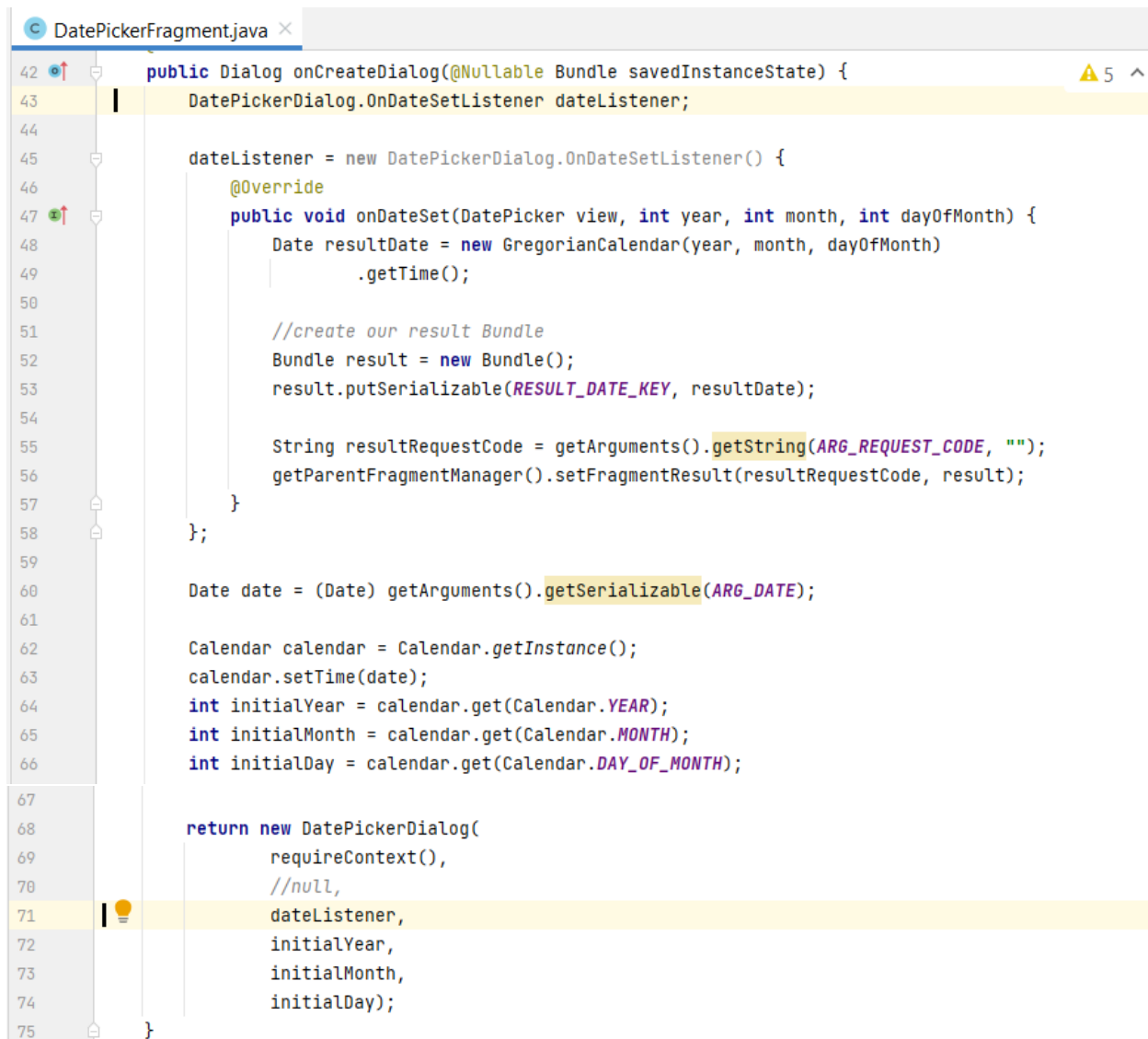
35. Now that you have a connection between CrimeFragment and DatePickerFragment, you need to send the date back to CrimeFragment.

```
DatePickerFragment.java
12     public class DatePickerFragment extends DialogFragment {
13
14         private static final String ARG_DATE = "date";
15         private static final String ARG_REQUEST_CODE = "requestCode";
16         private static final String RESULT_DATE_KEY = "resultDateKey";
17
18         @Override
19         public static Date getSelectedDate(Bundle result){
20             Date date = (Date) result.getSerializable(RESULT_DATE_KEY);
21             return date;
22         }
23     }
```

36. Next, implement the `FragmentManagerListener` in `CrimeFragment`, set the date on the crime property and update the UI.

```
CrimeFragment.java
113     mDateButton = v.findViewById(R.id.crime_date);
114     mDateButton.setText(mCrime.getDate().toString());
115     //mDateButton.setEnabled(false);
116     mDateButton.setOnClickListener(new View.OnClickListener() {
117         @Override
118         public void onClick(View v) {
119             FragmentManager manager = getChildFragmentManager();
120             manager.setFragmentManagerListener(
121                 REQUEST_DATE,
122                 getViewLifecycleOwner(),
123                 new FragmentResultListener() {
124                     @Override
125                     public void onFragmentResult(String requestKey, Bundle result) {
126                         //handling result
127                         if (requestKey.equals(REQUEST_DATE)){
128                             if (result == null){
129                                 return;
130                             }
131                             mCrime.setDate(DatePickerFragment.getSelectedDate(result));
132                             updateUI();
133                         }
134                     }
135                 }
136             );
137     });
138     //DatePickerFragment dialog = new DatePickerFragment();
```

37. Now that `CrimeFragment` can respond to new dates, `DatePickerFragment` needs to send the new date when the user selects one. In `DatePickerFragment`, add a listener to the `DatePickerDialog` that sends the date back to `CrimeFragment`



```
42 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
43     DatePickerDialog.OnDateSetListener dateListener;
44
45     dateListener = new DatePickerDialog.OnDateSetListener() {
46         @Override
47         public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
48             Date resultDate = new GregorianCalendar(year, month, dayOfMonth)
49                 .getTime();
50
51             //create our result Bundle
52             Bundle result = new Bundle();
53             result.putSerializable(RESULT_DATE_KEY, resultDate);
54
55             String resultRequestCode = getArguments().getString(ARG_REQUEST_CODE, "");
56             getParentFragmentManager().setFragmentResult(resultRequestCode, result);
57         }
58     };
59
60     Date date = (Date) getArguments().getSerializable(ARG_DATE);
61
62     Calendar calendar = Calendar.getInstance();
63     calendar.setTime(date);
64     int initialYear = calendar.get(Calendar.YEAR);
65     int initialMonth = calendar.get(Calendar.MONTH);
66     int initialDay = calendar.get(Calendar.DAY_OF_MONTH);
67
68     return new DatePickerDialog(
69         requireContext(),
70         //null,
71         dateListener,
72         initialYear,
73         initialMonth,
74         initialDay);
75 }
```

38. The `OnDateSetListener` is used to receive the date the user selects. The first parameter is for the `DatePicker` the result is coming from.
39. The selected date is provided in year, month, and day format, but you need a `Date` to send back to `CrimeFragment`. You pass these values to the `GregorianCalendar` and access the `time` property to get a `Date` object.
40. Once you have the date, it needs to be sent back to `CrimeFragment`. The `targetFragment` property stores the fragment instance that started your `DatePickerFragment`.
41. Now the circle is complete.

42. Run `CriminalIntent` to ensure that you can, in fact, control the dates.

43. Change the date of a `Crime` and confirm that the new date appears in `CrimeFragment`'s view. Then return to the list of crimes and check the `Crime`'s date to ensure that the model layer was updated.

Task : More Dialogs

Write another dialog fragment named `TimePickerFragment` that allows the user to select what time of day the crime occurred using a `TimePicker` widget. Add another button to `CrimeFragment` to display a `TimePickerFragment`.

#####

Notes:

1. Create folder `PRG6_M5_P2`.
2. Zip the folder and submit it to the server.

Bibliography:

- Marsicano, et. al., “Android Programming – The Big Nerd Ranch”, 5th Ed, 2022, Pearson Technology.