Dialogs

based on Communicating with Fragments | Android Developers (2022)

Communicating with Fragments

The Fragment library provides
two options for communication:

1. A shared ViewModel
2. The Fragment Result API

*Some of these approaches still employ fragment arguments, so fragment arguments are *not* obsolete!*

Sharing Data Using a ViewModel

- Share data with the host activity
- Share data between fragments in the same activity
- Share data between a parent and child fragment

In Criminal Intent, we do have two activities that share the same host, but always at different times (we swap them out)

In our case, it is reasonable for each fragment to have its own view-model (so the view-model is not shared)

Therefore, we pass information from one fragment to another using fragment arguments.

Sharing Data Using a ViewModel

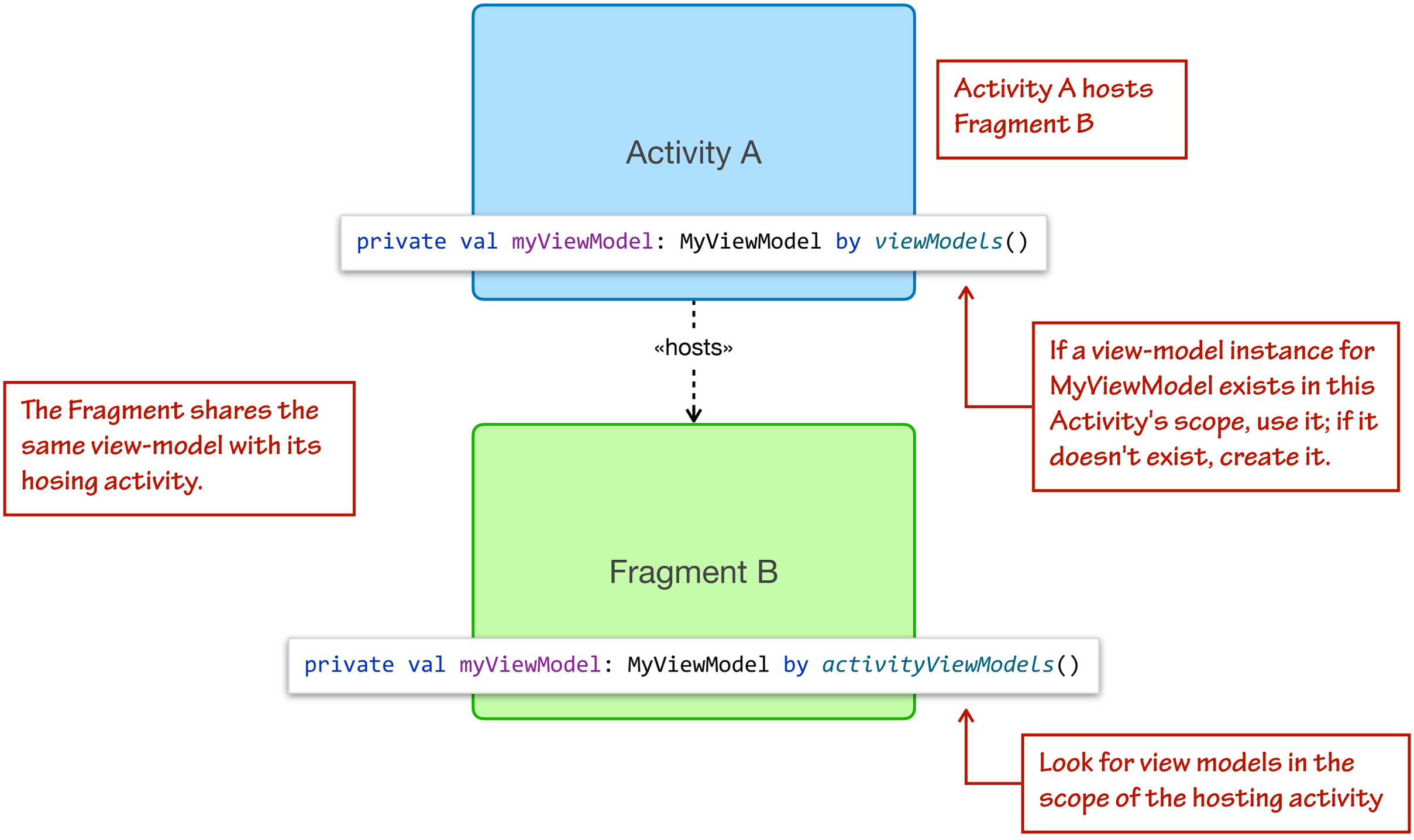
Key Detail

Within a specific scope, there can only one instance of each *type* of ViewModel.

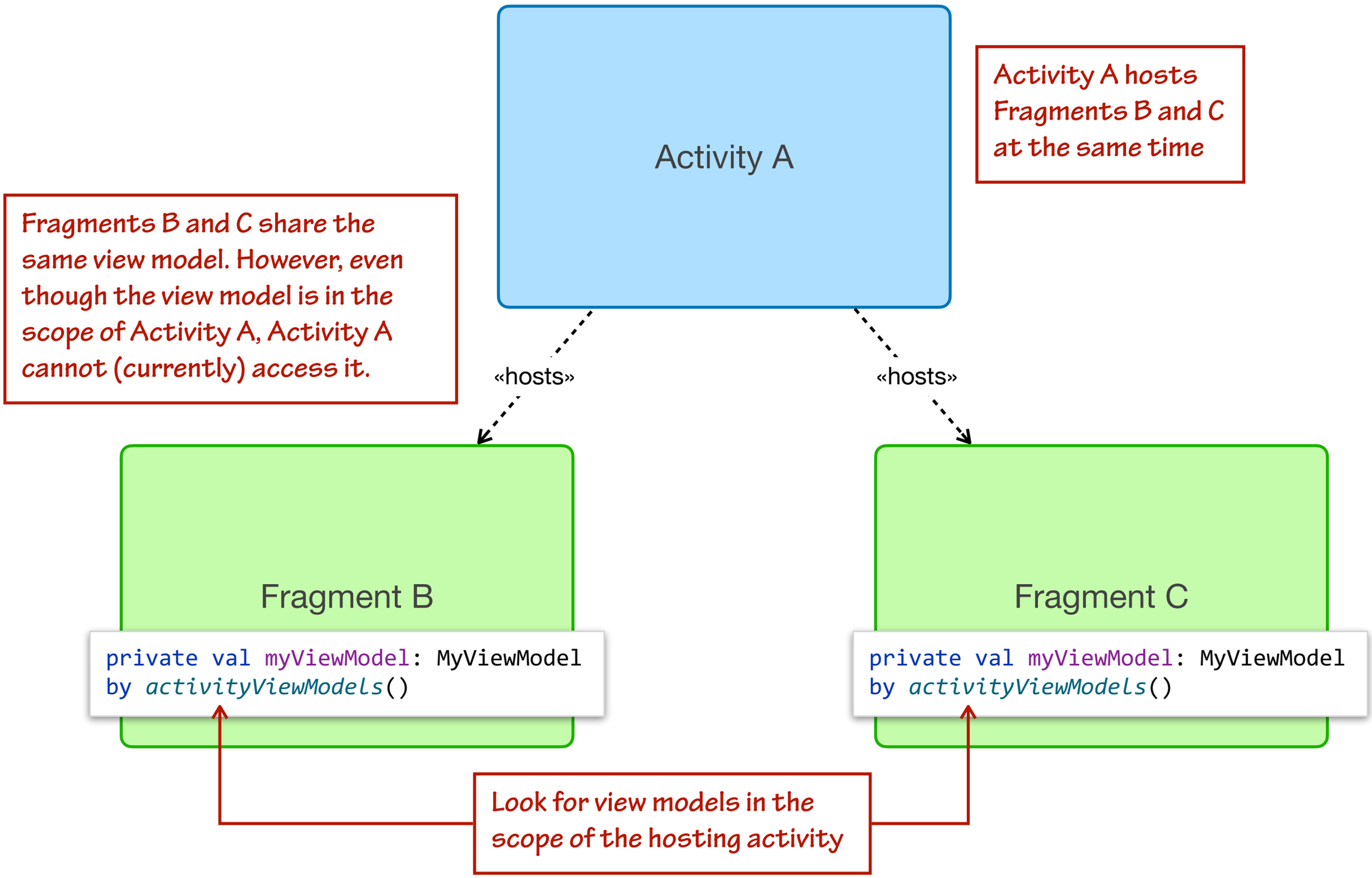
Nothing prohibits an Activity from having multiple view-models. For example, an activity can have one view-model of type MyViewModel and one view-model of type YourViewModel.

However, the same Activity cannot have two different instances of type MyViewModel.

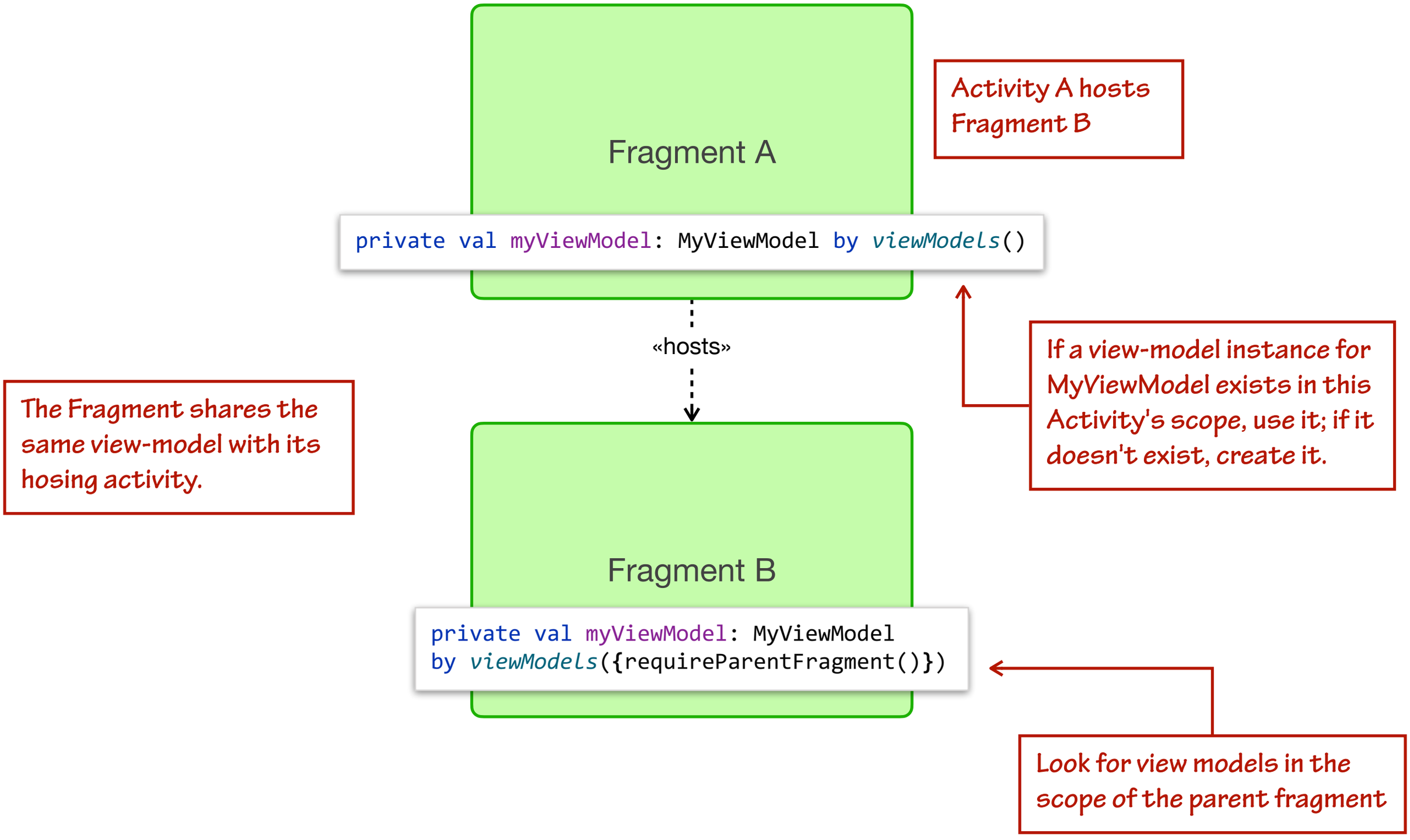
Share Data with the Host Activity



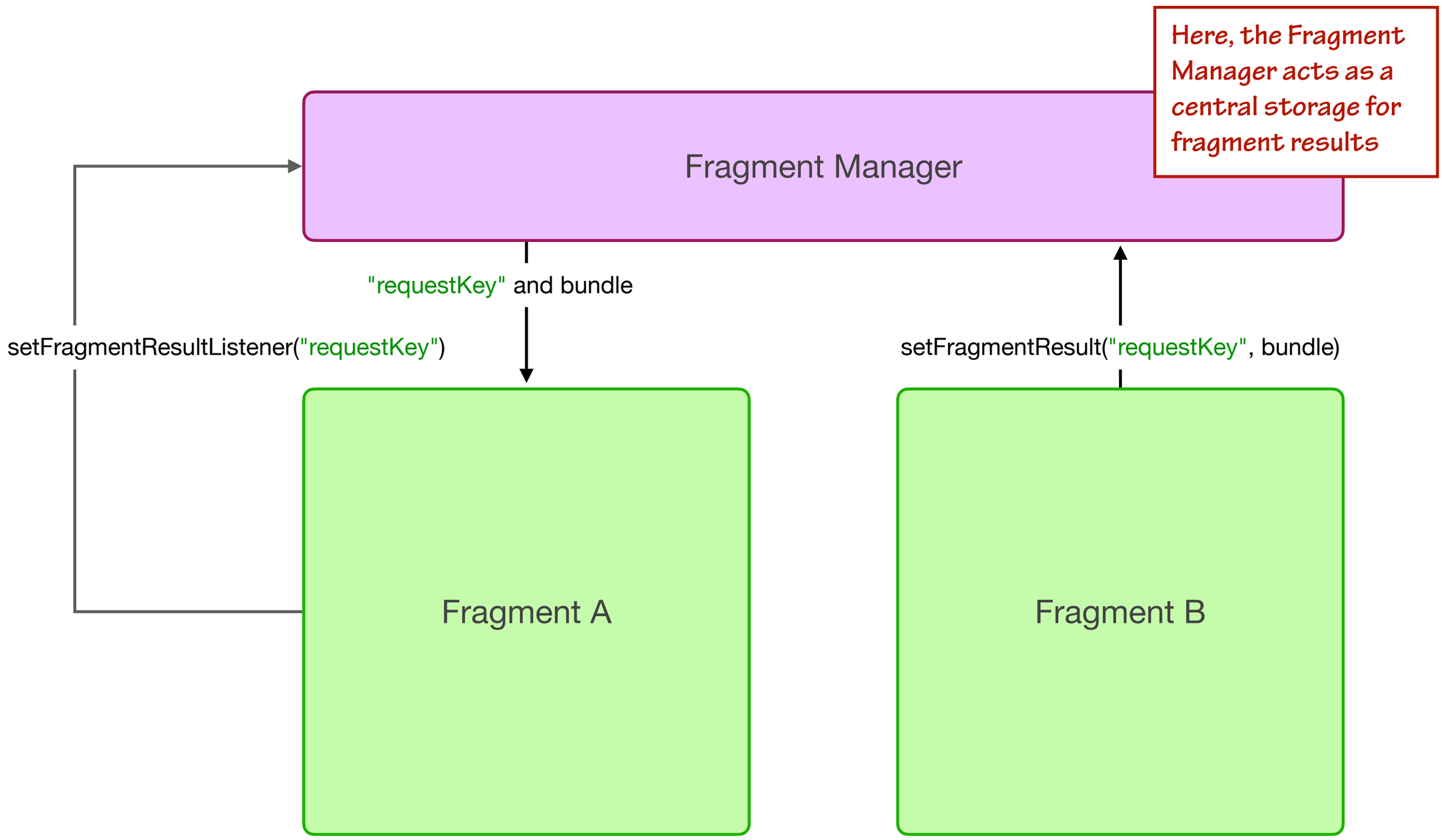
Share Data with the Host Activity



Share Data between Parent and Child Fragments



Pass Results between Fragments



DatePicker Dialog

- DatePickerFragment
- CrimeDetailFragment

BNR Approach is Deprecated

Chapter 13 of BNR describes an approach to the data-picker dialog that involves setTargetFragment. This approach is deprecated. We will use the approach of passing results between fragments described on the Android Developer website.

```
private const val ARG_ORIGINAL_DATE = "original_date"
private const val ARG_REQUEST_KEY = "request_key"

class DatePickerFragment : DialogFragment() {

    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {

        ...

    }

    companion object {

        fun newInstance(date: Date, requestKey: String): DatePickerFragment {

        ...

    }
}
```

Bundle: If an activity or fragment is being re-initialized after previously being shut down then this Bundle contains the data it most recently supplied in onSaveInstanceState(Bundle).

— Android Developers

Returns a Dialog.

These are passed in by CrimeDetailFragment. They become the fragment arguments used by the DatePickerFragment.

The constants above are the keys

```
override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {

    val dateListener = DatePickerDialog.OnDateSetListener {
            _: DatePicker, year: Int, month: Int, day: Int ->
        val resultDate : Date = GregorianCalendar(year, month, day).time

        parentFragmentManager.setFragmentResult(
            arguments?.getString(ARG_REQUEST_KEY).toString(),
            bundleOf(CrimeDetailFragment.ARG_NEW_DATE to resultDate)
        )
    }
    val date = arguments?.getSerializable(ARG_ORIGINAL_DATE) as Date
    val calendar = Calendar.getInstance()
    calendar.time = date
    val initialYear = calendar.get(Calendar.YEAR)
    val initialMonth = calendar.get(Calendar.MONTH)
    val initialDay = calendar.get(Calendar.DAY_OF_MONTH)
    return DatePickerDialog(
        requireContext(),
        dateListener,
        initialYear,
        initialMonth,
        initialDay
    )
}
```

This is the code that runs when you pick a new date. A result-date is created and passed (via a bundle) to the parentFragmentManager.

The rest of the code uses the data passed in (original date) to set up the dialog

```
    companion object {
        fun newInstance(date: Date, requestKey: String): DatePickerFragment {
            val args = Bundle().apply {
                putSerializable(ARG_ORIGINAL_DATE, date)
                putString(ARG_REQUEST_KEY, requestKey)
            }
            return DatePickerFragment().apply {
                arguments = args
            }
        }
    }
```

This code takes the arguments from CrimeDetailFragment and puts them into the fragment arguments.

The onStart lifecycle function of CrimeDetailFragment

When the user clicks the date button, the date picker dialog is invoke. It required the current selected date, and it requires the request-key, which is used to send data to the fragment manager.

```
private const val REQUEST_KEY = "request_key"

class CrimeDetailFragment : Fragment() {
    ...

    override fun onStart() {
        super.onStart()
        ...

        ui.crimeDate.setOnClickListener {
            DatePickerFragment.newInstance(crime.date, REQUEST_KEY)
                .show(parentFragmentManager, REQUEST_KEY)
        }

        parentFragmentManager.setFragmentResultListener(
            REQUEST_KEY,
            viewLifecycleOwner)
        { _, bundle ->
            crime.date = bundle.getSerializable(ARG_NEW_DATE) as Date
            updateUI()
        }
    }
```

We need to listen for changes to data associated with the request key in the fragment manager. When we are notified of those changes, we set the new date and update the UI

DreamCatcher Dialog

- AddReflectionDialog

You need to create dialog_add_reflection.xml
This simple layout contains an EditText view

okListener responds to clicking "OK"

This listener takes params we don't need

```
class AddReflectionDialog : DialogFragment() {
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        val ui = DialogAddReflectionBinding.inflate(LayoutInflater.from(context))
        val okListener = DialogInterface.OnClickListener { _, _ ->
            parentFragmentManager.setFragmentResult(REQUEST_KEY_ADD_REFLECTION,
                bundleOf(Pair(BUNDLE_KEY_REFLECTION_TEXT,
                                ui.reflectionText.text.toString()))
            )
        }

        return AlertDialog.Builder(requireContext())
            .setView(ui.root)
            .setTitle("Add Reflection")
            .setPositiveButton(android.R.string.ok, okListener)
            .setNegativeButton(android.R.string.cancel, null)
            .create()
    }
}
```

Define these constants in DreamDetailFragment

The simple AlertDialog takes care of some basic functionality for us

The date dialog in criminal intent had a new-instance method that allowed us to pass in the current date

This dialog is simpler because we're adding something rather than updating