SIMPLE APPLICATION DATA

23

23

Simple Application Data

- Save simple data as name-value pairs
- Two scenarios:
  - User preferences (`SharedPreferences`)
  - Application UI state between calls to subactivities
    - `onPause()`
    - `onSaveInstanceState (Bundle outState)`
    - `onDestroy()`
    - `onCreate(Bundle inState)`
    - `onRestoreInstanceState(Bundle inState)`
    - `onStart()`
    - `onResume()`

24

24

User Preferences

```
public static final String MY_PREFS = "mySharedPreferences";
public static final String USERNAME = “username”;
public static final String PASSWORD = “password”;

protected void savePreferences(String chatUsername, char[] password) {
  // Create or retrieve the shared preference object.
  int mode = Activity.MODE_PRIVATE;
  SharedPreferences mySharedPreferences =
              getSharedPreferences(MY_PREFS, mode);

  // Retrieve an editor to modify the shared preferences.
  SharedPreferences.Editor editor = mySharedPreferences.edit();

  // Store new primitive types in the shared preferences object.
  editor.putString(USERNAME, chatUsername);
  editor.putString(PASSWORD, new String(password));

  // Commit the changes.
  editor.commit();
}
```

25

25

User Preferences

```
public static final String MY_PREFS = "mySharedPreferences";
public static final String USERNAME = “username”;
public static final String PASSWORD = “password”;

public void loadPreferences() {
  // Restore preferences
  int mode = Activity.MODE_PRIVATE;
  SharedPreferences mySharedPreferences =
              getSharedPreferences(MY_PREFS, mode);

  String username = mySharedPreferences.getString(USERNAME, null);

  char[] password =
              mySharedPreferences.getString(PASSWORD, "").toCharArray();
}
```

26

26

User Preferences

- Preferences
  - Cached in-memory prefs object
  - Shared by app components
  - Saved in app file space
- Concurrency and reliability
  - Saving is atomic
  - No concurrency control i.e. no locking
  - No transactions i.e. keep prefs file small

27

27

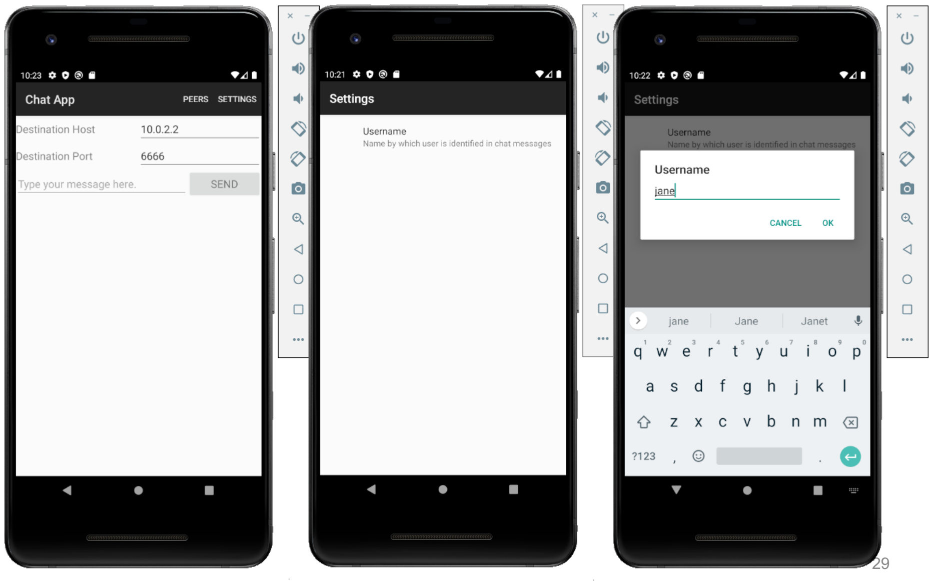
User Preferences

- Classes
  - `Preferences`: just one file
  - `SharedPreferences`: multiple files
  - `PreferenceActivity`: UI for setting preferences
- Modes
  - `MODE_PRIVATE`: only visible to app
  - ~~`MODE_MULTI_PROCESS`: for multi-process app~~
    - ~~Always check file for updates~~

28

28

Editing Preferences



Destination Host 10.0.2.2

Destination Port 6666

Type your message here.

Settings

Username
Name by which user is identified in chat messages

Username
jane

29

29

Using Preferences Library

- App Gradle dependencies:

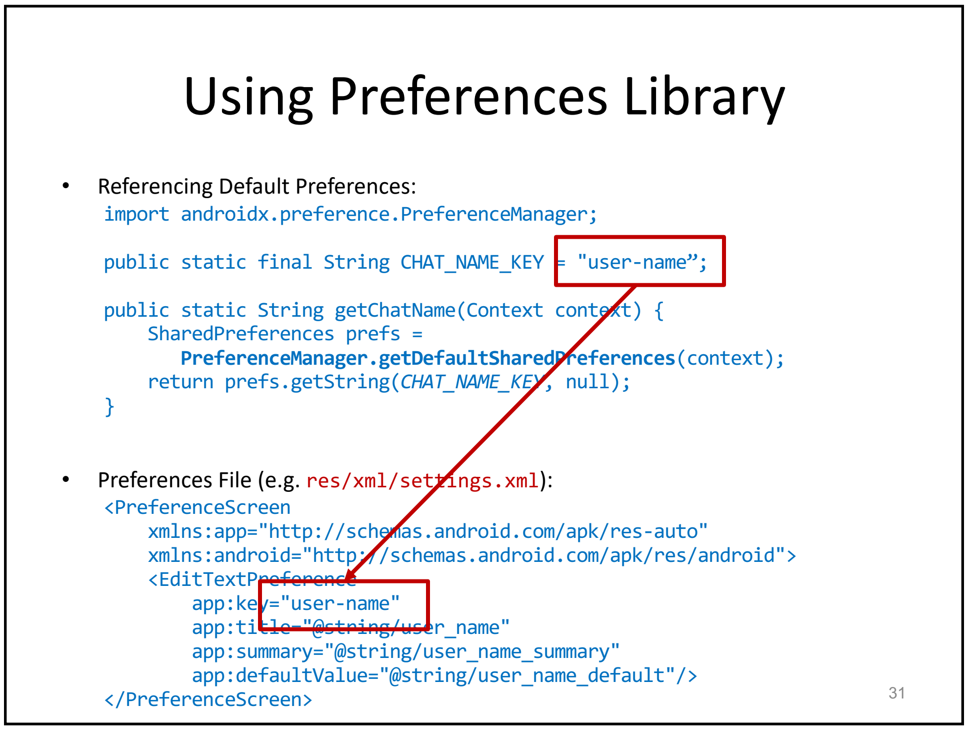
```
dependencies {
   ...
   implementation "androidx.preference:preference:1.1.0”
   ...
}
```

- Preferences File (e.g. `res/xml/settings.xml`):

```
<PreferenceScreen
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference
        app:key="user-name"
        app:title="@string/user_name"
        app:summary="@string/user_name_summary"
        app:defaultValue="@string/user_name_default"/>
</PreferenceScreen>
```

30

30



Using Preferences Library

- Referencing Default Preferences:

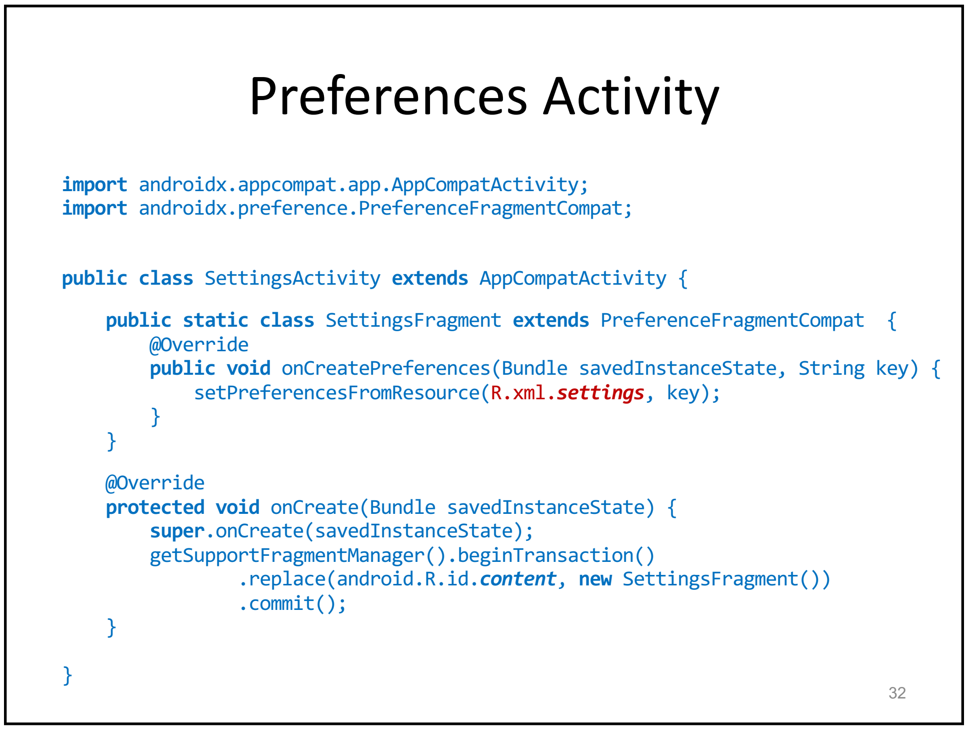
```
import androidx.preference.PreferenceManager;

public static final String CHAT_NAME_KEY = "user-name”;

public static String getChatName(Context context) {
    SharedPreferences prefs =
       PreferenceManager.getDefaultSharedPreferences(context);
    return prefs.getString(CHAT_NAME_KEY, null);
}
```

- Preferences File (e.g. `res/xml/settings.xml`):

```
<PreferenceScreen
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference
        app:key="user-name"
        app:title="@string/user_name"
        app:summary="@string/user_name_summary"
        app:defaultValue="@string/user_name_default"/>
</PreferenceScreen>
```

31

31



Preferences Activity

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.preference.PreferenceFragmentCompat;


public class SettingsActivity extends AppCompatActivity {

    public static class SettingsFragment extends PreferenceFragmentCompat  {
        @Override
        public void onCreatePreferences(Bundle savedInstanceState, String key) {
            setPreferencesFromResource(R.xml.settings, key);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportFragmentManager().beginTransaction()
                .replace(android.R.id.content, new SettingsFragment())
                .commit();
    }

}
```

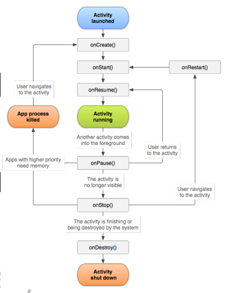
32

32

```
<activity
    android:name=".activities.SettingsActivity"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
    android:label="@string/title_activity_settings" />
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.preference.PreferenceFragmentCompat;


public class SettingsActivity extends AppCompatActivity {

    public static class SettingsFragment extends PreferenceFragmentCompat  {
        @Override
        public void onCreatePreferences(Bundle savedInstanceState, String key) {
            setPreferencesFromResource(R.xml.settings, key);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportFragmentManager().beginTransaction()
                .replace(android.R.id.content, new SettingsFragment())
                .commit();
    }

}
```

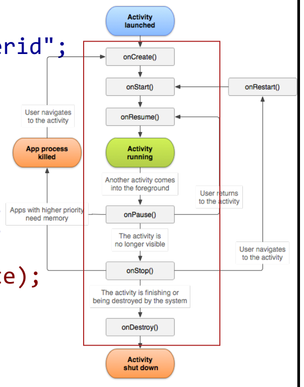
Simple Application Da


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

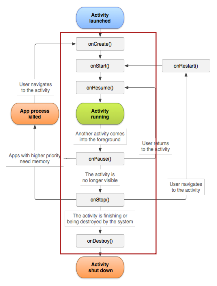
- Save simple data as name-value pai
- Two scenarios:
  - User preferences (SharedPreference
  - Application UI state between calls to subactivities
    - onPause()
    - onSaveInstanceState (Bundle outState)
    - onDestroy()
    - onCreate(Bundle inState)
    - ~~onRestoreInstanceState(Bundle inState)~~
    - onStart()
    - onResume()

Application UI State

```
// Save UI state while activity is not active
// (i.e. UI state for a single user session)

private static final String USERID_KEY = ”userid";

private String loggedInUser;

@Override
public void onSaveInstanceState(
                 Bundle savedInstanceState) {

  super.onSaveInstanceState(savedInstanceState);

  savedInstanceState.putString(
             USERID_KEY,
             loggedInUser);
}
```


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Application UI State


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

```
// Save UI state while activity is not active
// (i.e. UI state for a single user session)

private static final String USERID_KEY = ”userid";

private String loggedInUser;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

   loggedInUser = savedInstanceState.getString(USERID_KEY);
}
```