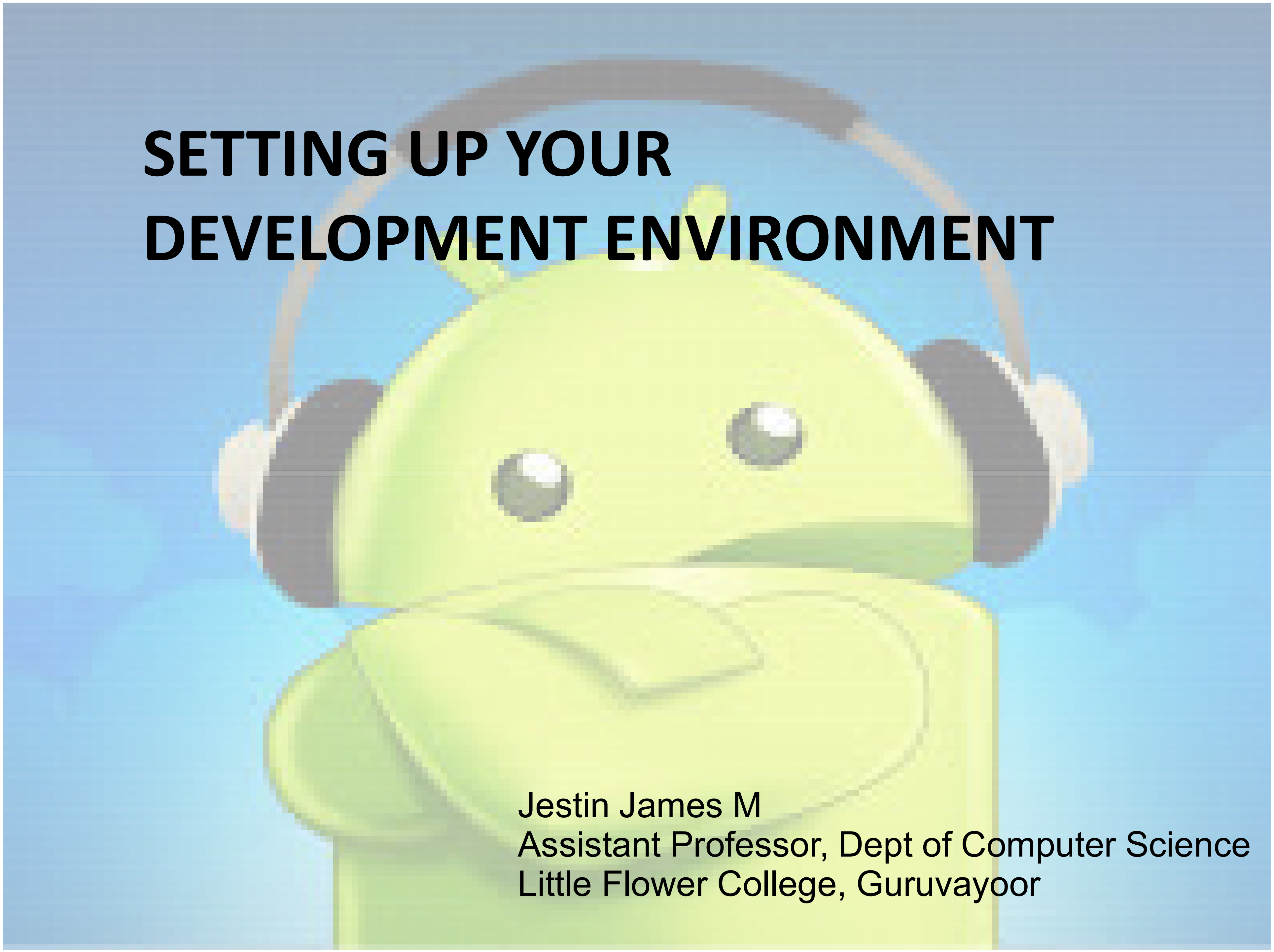


SETTING UP YOUR DEVELOPMENT ENVIRONMENT

Jestin James M
Assistant Professor, Dept of Computer Science
Little Flower College, Guruvayoor

Setting Up Your Development Environment

- The Android SDK requires JDK 5 or JDK 6 (the examples use JDK 6) and Eclipse 3.5 or higher
- The Android SDK is compatible with Windows (Windows XP, Windows Vista, and Windows 7),
- Mac OS X (Intel only), and Linux (Intel only).
- In terms of hardware, you need an Intel machine, the more powerful the better

Setting Up Your Development Environment

- Android Development Tools (ADT).
- ADT is an Eclipse plug-in that supports building Android applications with the Eclipse IDE.
- The Android SDK is made up of two main parts:
- The tools and the packages

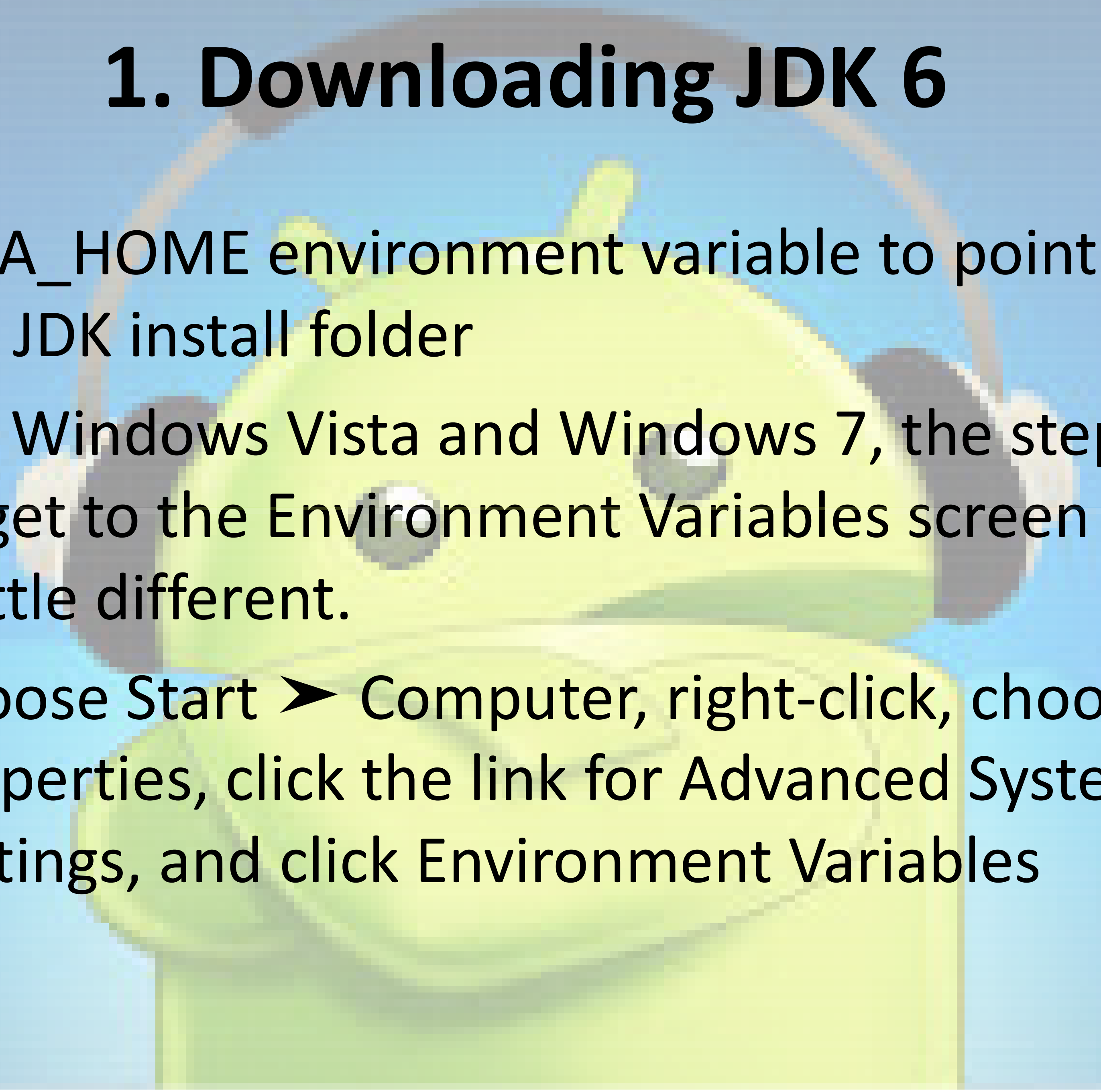
Setting Up Your Development Environment

- When you first install the SDK, all you get are the base tools.
- The packages are the files specific to a particular version of Android (called a *platform) or a particular add-on to a platform*

Setting Up Your Environment
1. Downloading JDK 6

- The first thing you need is the Java SE Development Kit.
- The Android SDK requires JDK 5 or higher; we developed the examples using JDK 6.
- For Windows, download JDK 6 from the Oracle web site (www.oracle.com/technetwork/java/javase/downloads/index.html) and install it

1. Downloading JDK 6



- JAVA_HOME environment variable to point to the JDK install folder
- For Windows Vista and Windows 7, the steps to get to the Environment Variables screen are a little different.
- Choose Start ➤ Computer, right-click, choose Properties, click the link for Advanced System Settings, and click Environment Variables

1. Downloading JDK 6

- click New to add the variable or Edit to modify it if it already exists.
- The value of JAVA_HOME is something like C:\Program Files\Java\jdk1.6.0_27.

2. Downloading Eclipse 3.6

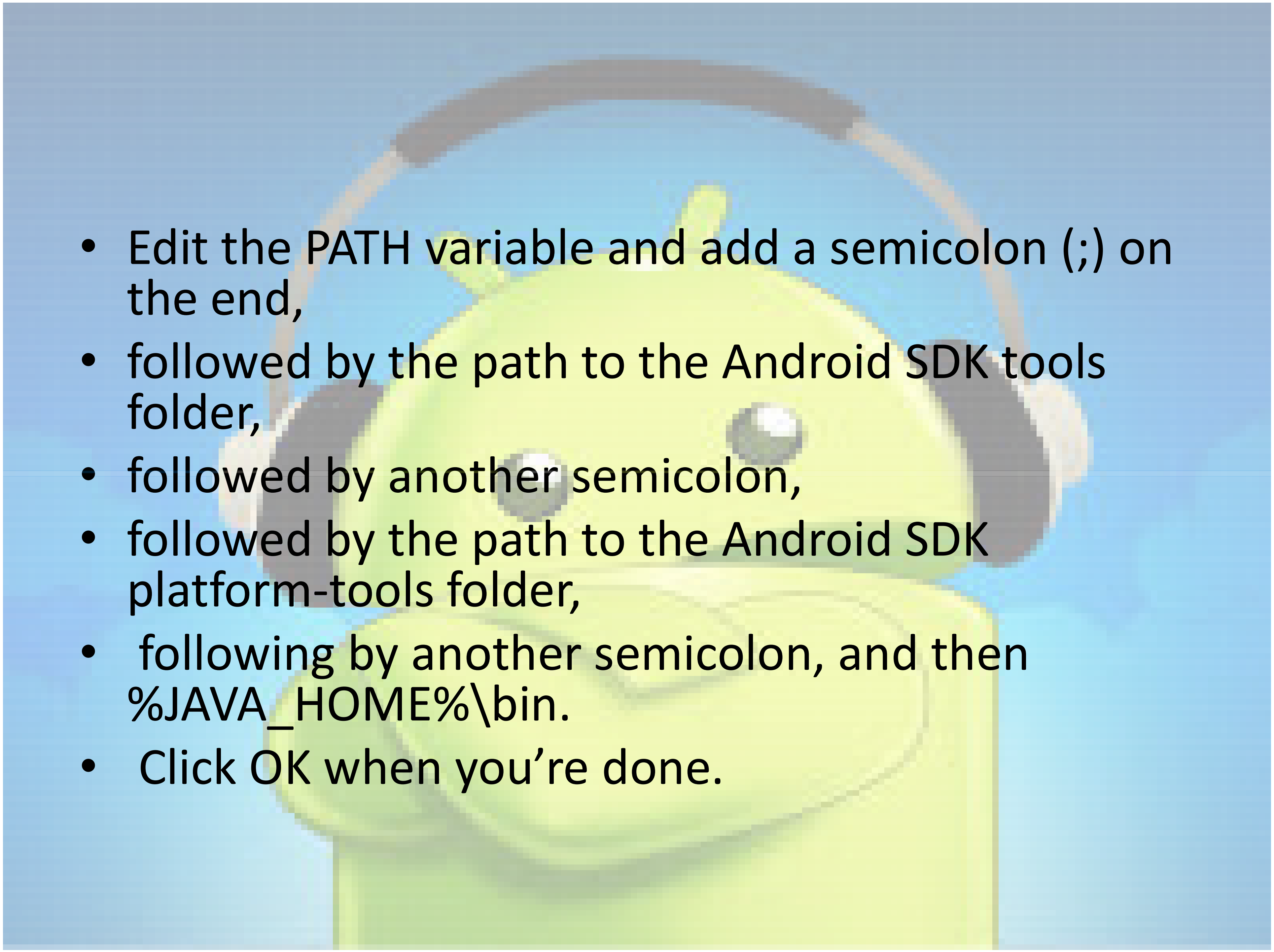
- download the Eclipse IDE for Java Developers
- You can download all versions of Eclipse from www.eclipse.org/downloads/.
- When you first start up Eclipse, it asks you for a location for the workspace.
- To make things easy, you can choose a simple location such as C:\android or a directory under your home directory.

3. Downloading the Android SDK

- You can download the Android SDK from http://developer.android.com/sdk

4. Updating Your PATH Environment Variable

- The Android SDK comes with a tools directory that you want to have in your PATH.
- You also need in your PATH the platform-tools directory you just installed



- Edit the PATH variable and add a semicolon (;) on the end,
- followed by the path to the Android SDK tools folder,
- followed by another semicolon,
- followed by the path to the Android SDK platform-tools folder,
- following by another semicolon, and then %JAVA_HOME%\bin.
- Click OK when you’re done.

5. Installing Android Development Tools (ADT)

- To get started, launch the Eclipse IDE and follow these steps:
- **1. Select Help ➤ Install New Software.**
- **2. Select the Work With field, type in** https://dl-ssl.google.com/android/eclipse/ and press Enter.

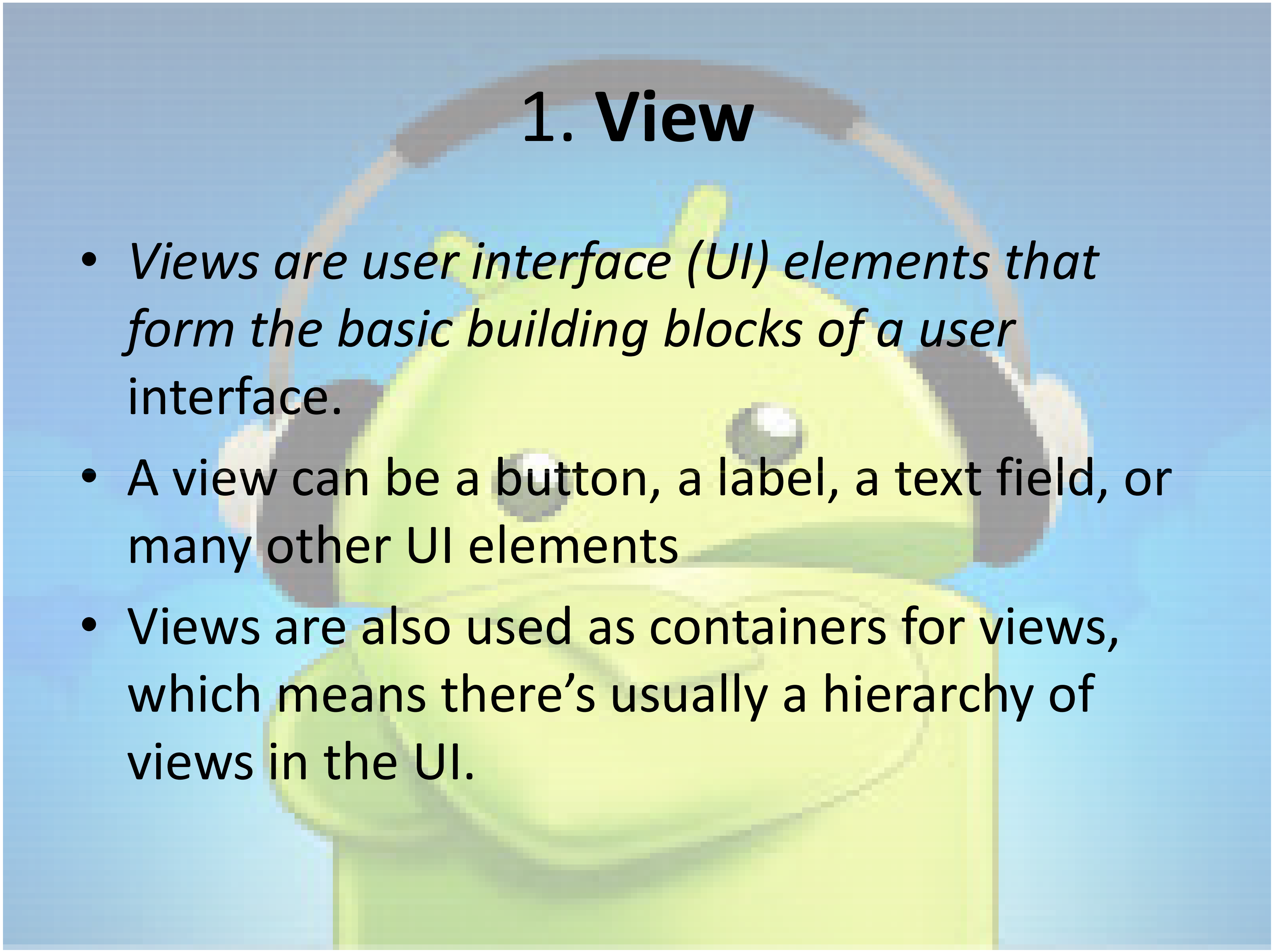
5. Installing Android Development Tools (ADT)

- **3. You should see an entry named Developer Tools with four child nodes:**
- Android DDMS, Android Development Tools, Android Hierarchy Viewer, and Android Traceview. Select the parent node Developer Tools, make sure the child nodes are also selected, and click the Next button.
- The versions you see may be newer than these, and that’s okay

5. Installing Android Development Tools (ADT)

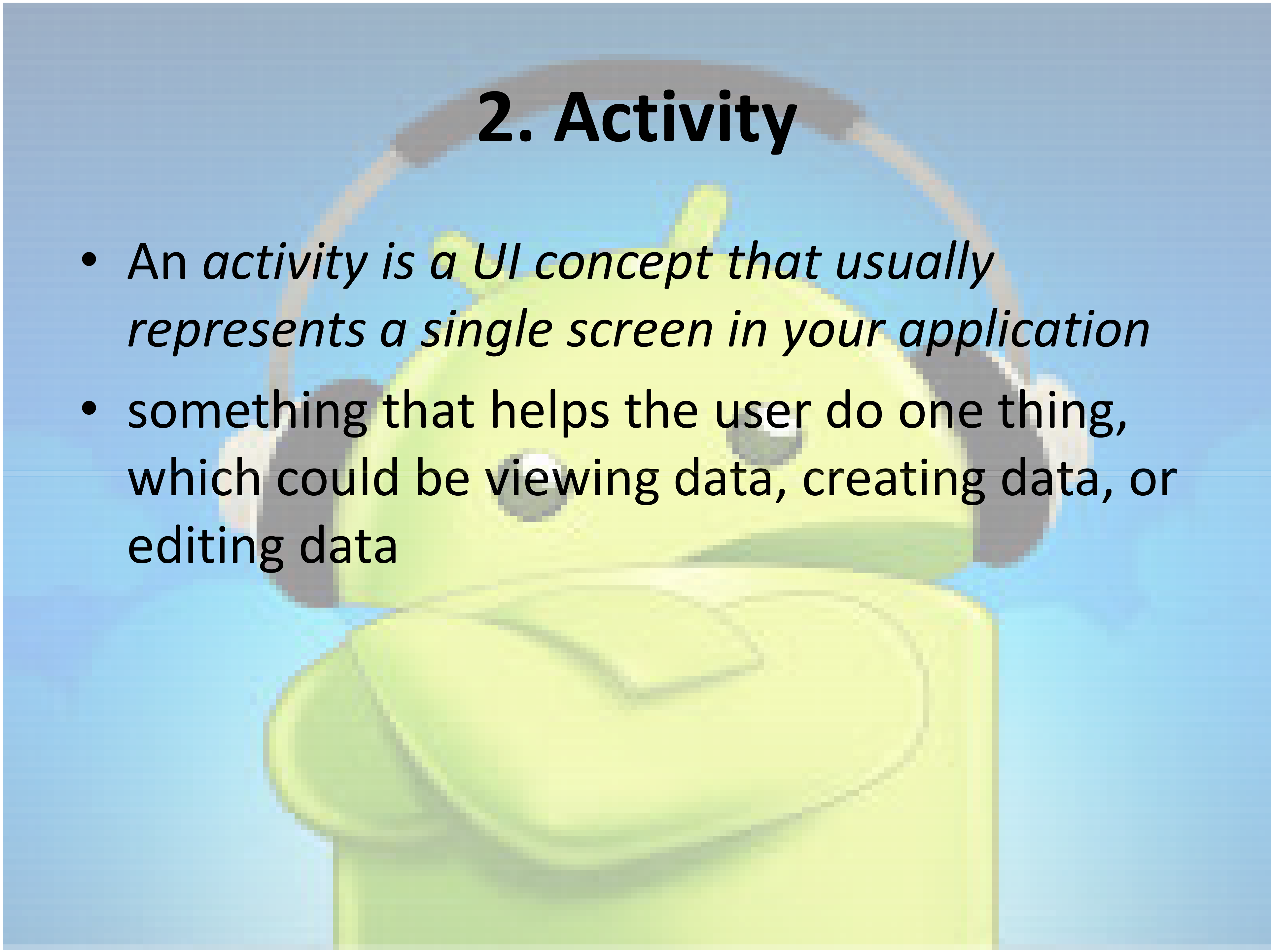
- **4. Eclipse asks you to verify the tools to install. Click Next**
- **5. You're asked to review the licenses for ADT as well as for the tools** required to install ADT. Review the licenses, click "I accept," and then click the Finish button.

LEARNING THE FUNDAMENTAL COMPONENTS



1. **View**

- *Views are user interface (UI) elements that form the basic building blocks of a user* interface.
- A view can be a button, a label, a text field, or many other UI elements
- Views are also used as containers for views, which means there’s usually a hierarchy of views in the UI.



2. Activity

- An *activity is a UI concept that usually represents a single screen in your application*
- something that helps the user do one thing, which could be viewing data, creating data, or editing data

3. Fragment

- When a screen is large, it becomes difficult to manage all of its functionality in a single activity.
- *Fragments are like sub-activities, and an activity can display one or more* fragments on the screen at the same time.
- When a screen is small, an activity is more likely to contain just one fragment, and that fragment can be the same one used within larger screens.

4. Intent

An *intent generically defines an "intention" to do some work*

- Broadcast a message.
- Start a service.
- Launch an activity.
- Display a web page or a list of contacts.
- Dial a phone number or answer a phone call

4. Intent

- Intents can be explicit or implicit.
- to display a URL, the system decides what component will fulfill the intention.
- Intents loosely couple the action and action handler.

5. Content Provider

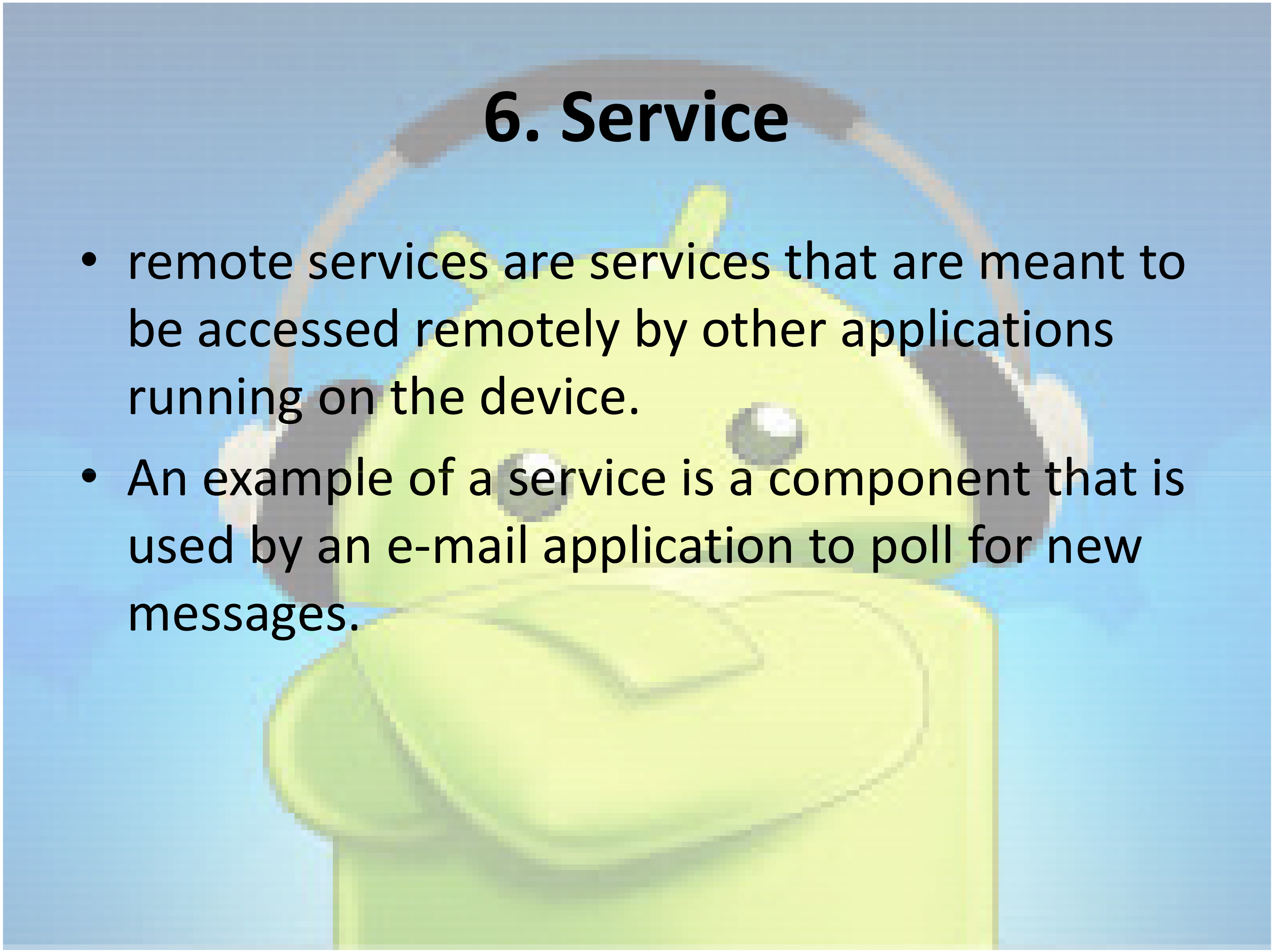
- Data sharing
- Android provide standard mechanism for applications to share data
- Through content providers, you can expose your data and have your applications use data from other applications.

6. Service

- *Services in Android resemble services you see in Windows or other platforms*
- Android defines two types of services: local services and remote services
- Local services are components that are only accessible by the application that is hosting the service

6. Service

- remote services are services that are meant to be accessed remotely by other applications running on the device.
- An example of a service is a component that is used by an e-mail application to poll for new messages.



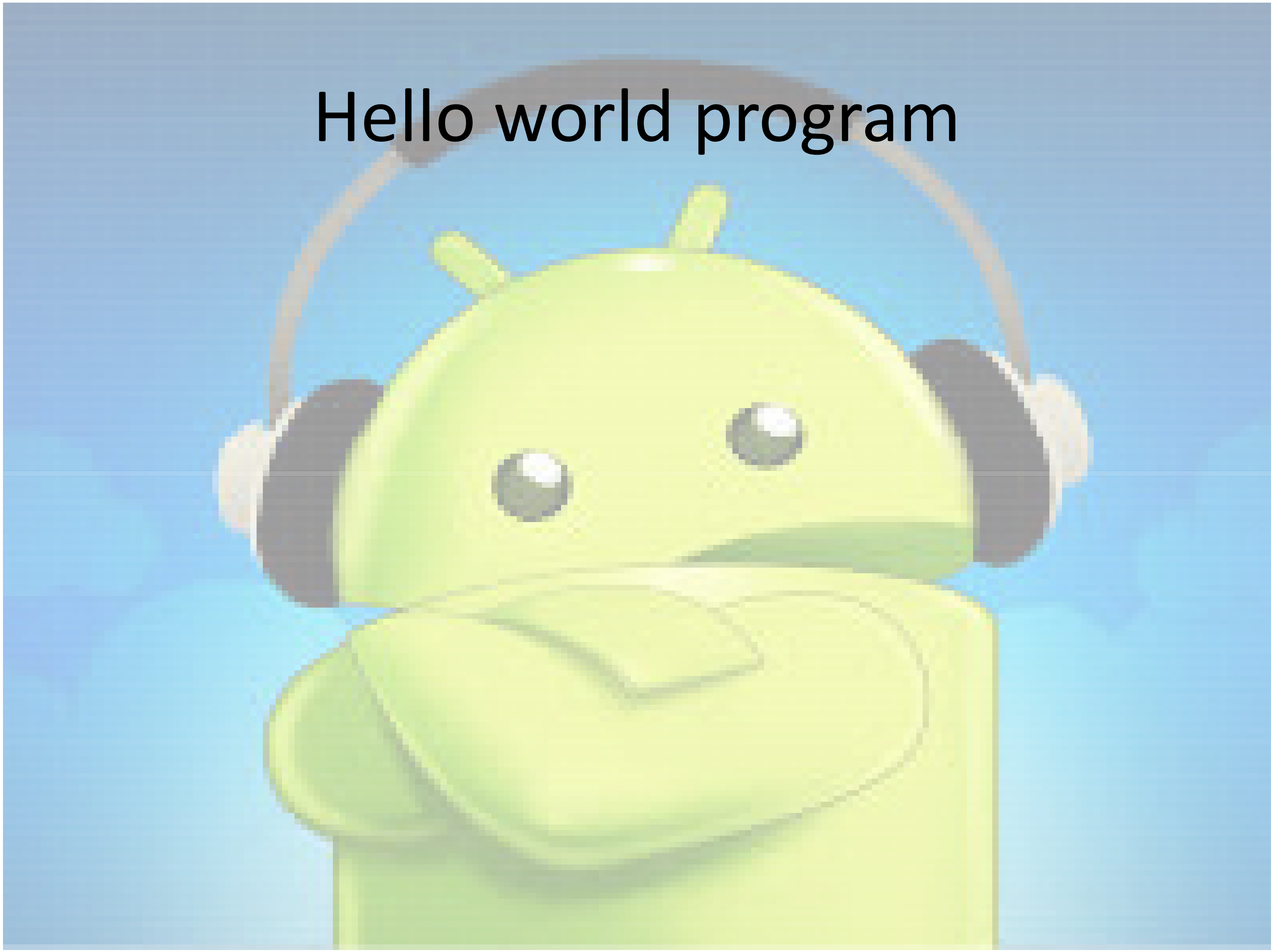
7. AndroidManifest.xml

- defines the contents and behavior of your application.
- example, it lists your application’s activities and services, along with the permissions and features the application needs to run.

8. Android Virtual Devices

- An Android Virtual Device (AVD) allows developers to test their applications without hooking up an actual Android device
- AVDs can be created in various configurations to emulate different types of real devices.

Hello world program

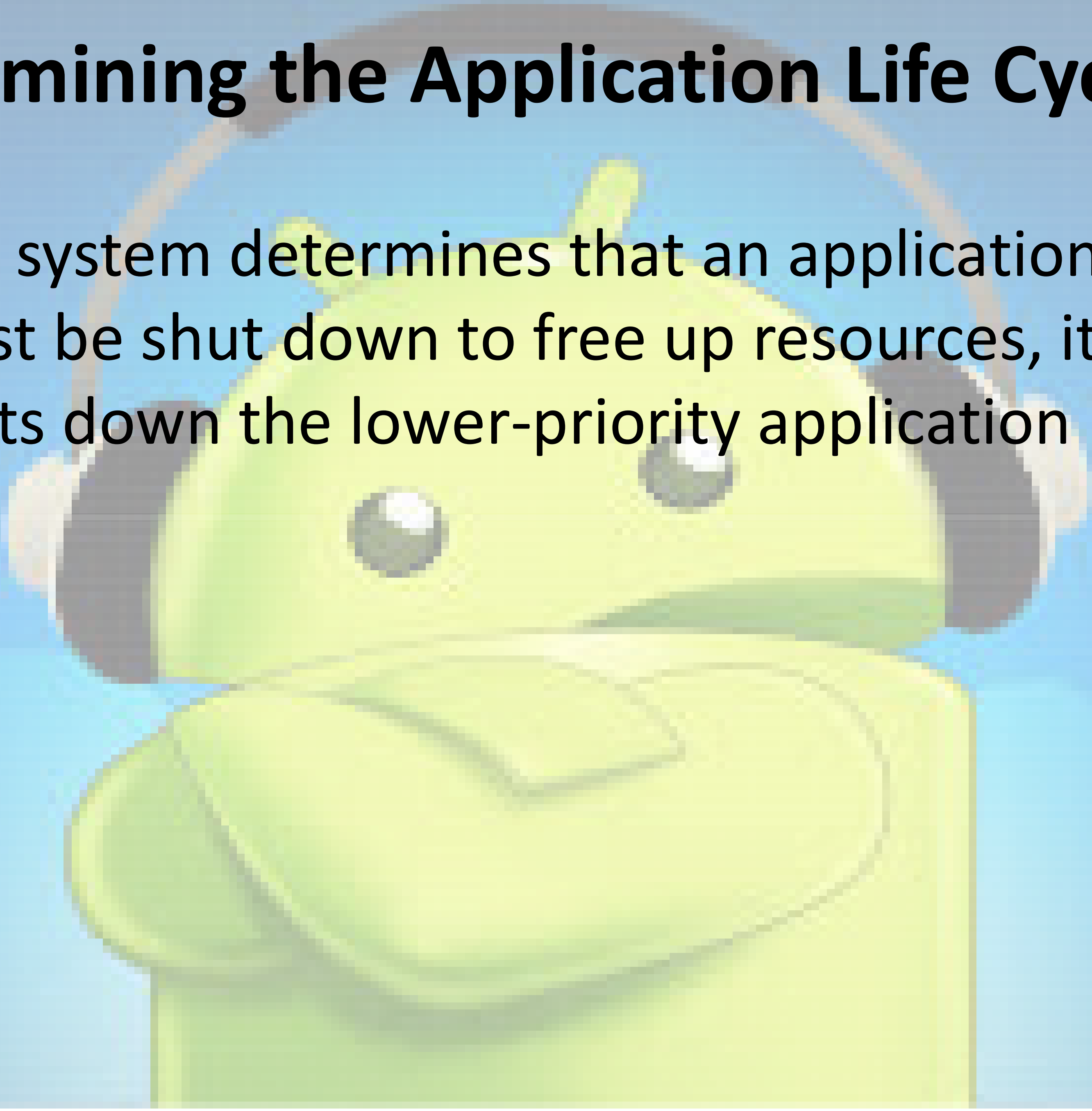


Examining the Application Life Cycle

- The life cycle of an Android application is strictly managed by the system
- A user may want to launch a web browser,
- but the system ultimately decides whether to start the application
- If the user is currently working with an activity, the system gives high priority to that application.

Examining the Application Life Cycle

- The system determines that an application must be shut down to free up resources, it shuts down the lower-priority application



J2EE

- J2EE apps are loosely managed by the container they run in.
- a J2EE container can remove an application from memory if it sits idle for a predetermined time period
- J2EE container usually has sufficient resources to run lots of applications at the same time
- With Android, resources are more limited, so Android must have more control and power over applications

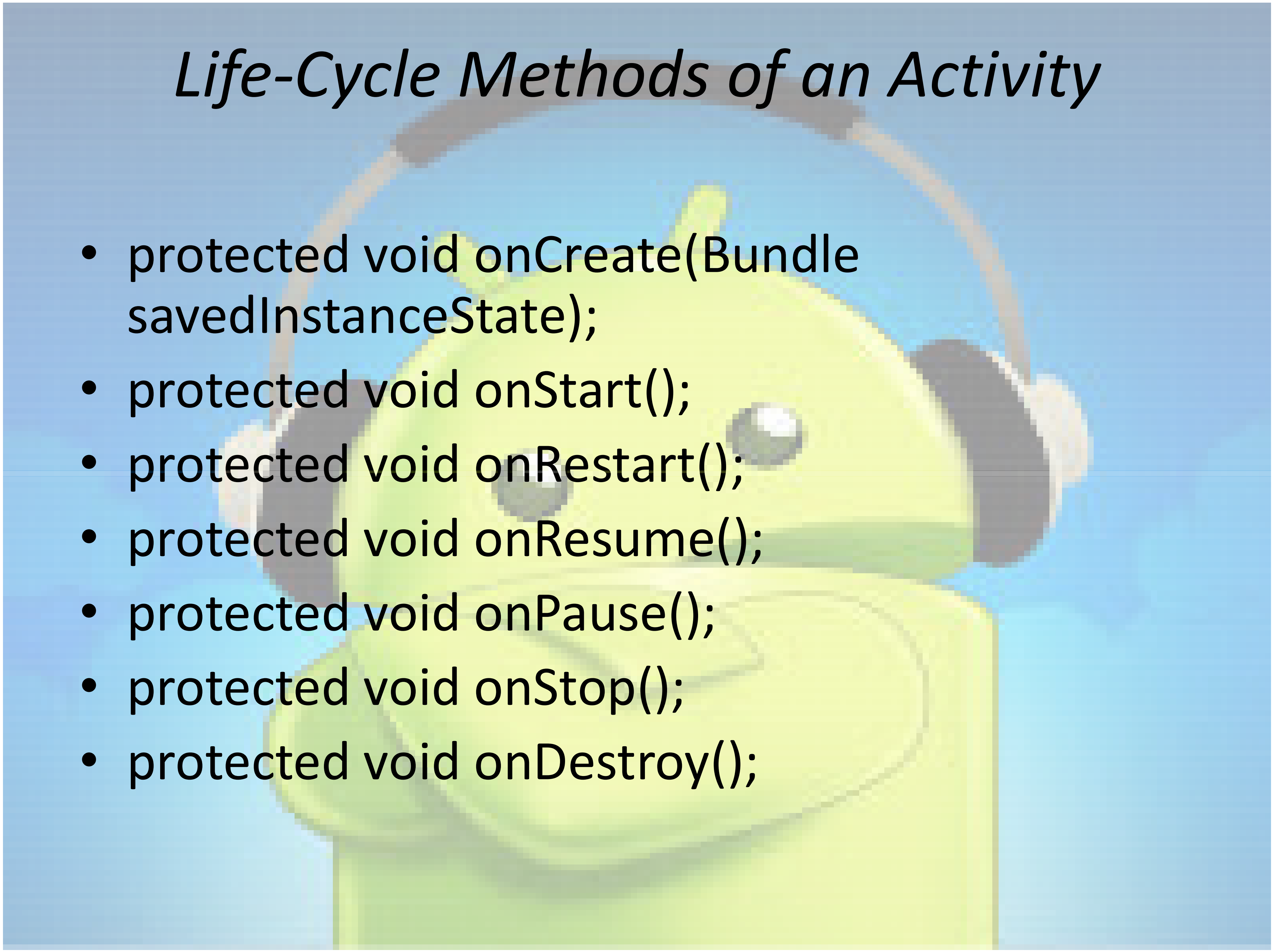
Examining the Application Life Cycle

- Android runs each application in a separate process,
- each of which hosts its own virtual machine.
- This provides a protected-memory environment.
- By isolating applications to an individual process, the system can control which application deserves higher priority.
- For example, a background process that's doing a CPU-intensive task can't block an incoming phone call.

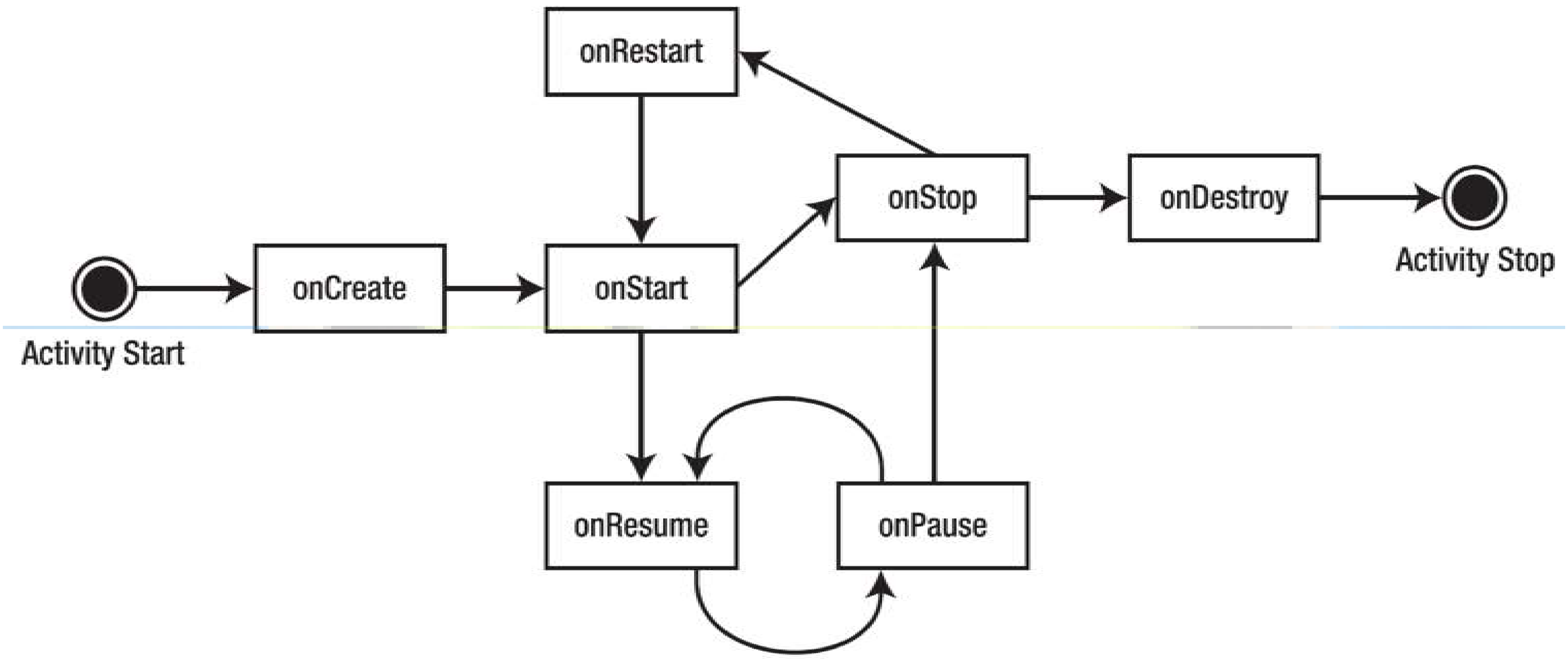
Examining the Application Life Cycle

- The concept of application life cycle is logical,
- Example : A user is talking to someone on the phone and needs to open an e-mail message to answer a question
- In the background, however, the system is saving and restoring application state.
- For instance, when the user clicks the link in the e-mail message
- the system saves metadata on the running e-mail message activity before starting the browser-application activity to launch a URL

Life-Cycle Methods of an Activity



- protected void onCreate(Bundle savedInstanceState);
- protected void onStart();
- protected void onRestart();
- protected void onResume();
- protected void onPause();
- protected void onStop();
- protected void onDestroy();


onRestart
Activity Start
onCreate
onStart
onStop
onDestroy
Activity Stop
onResume
onPause