GPCET
Pioneering Innovative Education

G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC with 'A' Grade of UGC, Approved by AICTE, New Delhi

Permanently Affiliated to JNTUA, Ananthapuramu

(Recognized by UGC under 2(f) and 12(B) & ISO 9001:2008 Certified Institution)

Nandikotkur Road, Venkayapalli, Kurnool – 518452

Department of Computer Science and Engineering

Bridge Course
On
MOBILE APPLICATION DEVELOPMENT

By

D.JAYANARAYANA REDDY

GPCET
Pioneering Innovative Education

Table of Contents

JAVA – BASICS

1.Java Basics introduction:

Object - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

 Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

 Methods - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

 Instance Variables - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

First Java Program

```
public class MyFirstJavaProgram {
/* This is my first java program.
* This will print 'Hello World' as the output
*/
public static void main(String []args) {
System.out.println("Hello World"); // prints Hello World
}
}
```

Basic Syntax

Case Sensitivity - Java is case sensitive, which means identifier **Hello**and **hello** would have different meaning in Java.

 Class Names - For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example: *class MyFirstJavaClass*

 Method Names - All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

Example: *public void myMethodName()*

 Program File Name - Name of the program file should exactly match the class name. When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile). **Example**: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*

 public static void main(String args[]) - Java program processing starts from the main() method which is a mandatory part of every Java program.

2.Java Identifiers

In Java, there are several points to remember about identifiers. They are as follows:

- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value.
- Examples of illegal identifiers: 123abc, -salary.

Java Modifiers

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers:

- **Access Modifiers:** default, public , protected, private
- **Non-access Modifiers:** final, abstract, strictfp

Java Variables

Following are the types of variables in Java:

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static Variables)

Classes in Java

A class is a blueprint from which individual objects are created.

Following is a sample of a class.

```
public class Dog{
String breed;
int ageC
String color;
void barking(){
}
void hungry(){
}
void sleeping(){
}
}
```

A class can contain any of the following variable types.

- **Local variables:** Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

 Instance variables: Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

 Class variables: Class variables are variables declared within a class, outside any method, with the static keyword.

Singleton Class:

The Singleton's purpose is to control object creation, limiting the number of objects to only one. Since there is only one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static fields. Singletons often control access to resources, such as database connections or sockets.

XML BASICS

3.XML INTRODUCTION:

Stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.

What is Markup?

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. So, what exactly is a markup language? Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

simple syntax rules:

```
<?xml version="1.0"?>
<contact-info>
<name>Tanmay Patil</name>
<company>TutorialsPoint</company>
<phone>(011) 123-4567</phone>
</contact-info>
```

XML Declaration

The XML document can optionally have an XML declaration. It is written as follows:

<?xml version="1.0" encoding="UTF-8"?>

Syntax Rules for XML Declaration

 The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.

 If the document contains XML declaration, then it strictly needs to be the first statement of the XML document.

 The XML declaration strictly needs be the first statement in the XML document.

 An HTTP protocol can override the value of *encoding* that you put in the XML declaration.

Syntax Rules for Tags and Elements

Element Syntax: Each XML-element needs to be closed either with start or with end elements as shown below:

<element>....</element>

or in simple-cases, just this way:

<element/>

4.XML Attributes

An **attribute** specifies a single property for the element, using a name/value pair. An XMLelement can have one or more attributes.

Syntax Rules for XML Attributes

 Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and *href* are considered two different XML attributes.

Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute ***b*** is specified twice:

....

Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax:

....

XML Document Example

A simple document is shown in the following example:

```
<?xml version="1.0"?>
<contact-info>
<name>Tanmay Patil</name>
<company>TutorialsPoint</company>
<phone>(011) 123-4567</phone>
</contact-info>
```

5.XML Declaration Examples

Following are few examples of XML declarations:

XML declaration with no parameters:

<?xml >

XML declaration with version definition:

<?xml version="1.0">

XML declaration with all parameters defined:

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

XML declaration with all parameters defined in single quotes:

<?xml version='1.0' encoding='iso-8859-1' standalone='no' ?>

XML tags as follows:

Start Tag

The beginning of every non-empty XML element is marked by a start-tag. Following is an example of start-tag:

<address>

End Tag

Every element that has a start tag should end with an end-tag. Following is an example of end-tag:

</address>

Note, that the end tags include a solidus ("/") before the name of an element.

Empty Tag

The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as follows:

A start-tag immediately followed by an end-tag as shown below:

<hr></hr>

A complete empty-element tag is as shown below:

<hr />

Empty-element tags may be used for any element which has no content.

ANDROID BASICS

6.ANDROID INTRODUCTION:

ADK:

Android Development Kit, What people use to develop anything for the Android such as APPs and ROM's

adb:

Android Debug Bridge, a command-line debugging application included with the SDK. It provides tools to browse the device, copy tools on the device, & forward ports for debugging. If you are developing in Eclipse using the ADT Plugin, adb is integrated into your development environment.

AOSP:

Android Open System Project, usually you will see this term when referring to a program or ROM. This will mean that the program or ROM was taken from Google & does not contain any modifications done by the phone Manufacturer or the phone service provider. This is Android the way Google intended.

BootLoader:

In literal terms, boot loader is code that is executed before any Operating System starts to run. The concept of boot loaders is universal to virtually all Operating systems that includes operating systems on your PC, laptop, Smartphone, & other such devices. Boot loaders basically package the instructions to boot operating system kernel & most of them also have their own debugging or modification environment. As the boot-loader kicks off before any piece of software on your device, it makes it extremely processor specific & every motherboard has its own bootloader.

Boot Loop:

Simply means something is preventing the phone from completing it's boot cycle & is stuck between the boot animation & the unlock screen, creating a looped animation

Brick or Bricked:

Jargon for a completely unrecoverable device, (no more than a brick or paperweight)
Bug or Software Bug: An Error or flaw in software that produces a failure or unexpected/unwanted result. Typically created from incorrect code, this is why some ROMs are better & smoother running than others because developers have taken the time to input "perfect" code (read put in a lot of hours & hard.

Busybox:

A single multicall binary that packages the functionality of most widely used standard Unix tools, BusyBox provides a fairly complete environment for any small or embedded system.

COMPCACHE:

(compressed caching) is, in short, virtual swap, setting aside a certain percentage (usually 25%) of your RAM as 'compressed' swap. Compcache compresses the data that would normally go to swap, then moves it back into RAM, and reverses the process when moving it out of the 'compressed' swap.

Dalvik Cache:

A program cache area for the program Dalvik. Dalvik is a java based virtual machine that is the basis for running your programs (the ones that have the .apk extension). In order to make access times faster (because there's not JIT (just in time) compiler installed by default), the dalvik-cache is the result of dalvik doing a optimization of the running program. It's similar to the prefetch files in Windows.

DDMS:

Dalvik Debug Monitor Service, a GUI debugging application included with the SDK. It provides screen capture, log dump, and process examination capabilities.

Deep Sleep: A state when the CPU is off, display dark, device is waiting for external input.

De-odex:

Apk files have respective odexes that devs use to supposedly save space. Deodexing means you convert it back to a .dex file & put it back inside the apk. This allows you to easily replace files (not having to worry about odexes), but the main point was to deodex services.jar so that you can change all text to different colors (such as the clock color to white) & to deodex services.jar, you need to deodex everything.

Dev. or Developer:

An individual that creates, or alters a file in such a manner as to advance the program

Drawable:

A compiled visual resource that can be used as a background, title, or other part of the screen. A drawable is typically loaded into another UI element, for example as a background image. A drawable is not able to receive events, but does assign various other properties such as "state" and scheduling, to enable subclasses such as animation objects or image libraries.

Flash - Flashing :

The process of applying a firmware image (or ROM) to a device. It generally entails a very specific order of steps. Failing to complete any one of these steps properly may result in soft/hard) bricking

Basic4android:

It is a commercial product similar to Simple.It is inspired by Microsoft Visual Basic 6 and Microsoft Visual Studio. It makes android programming much simpler for regular Visual Basic programmers who find coding in Java difficult.

Corona:

Corona lets developers build graphic applications by using its integrated Lua language, which is layered on top of C++/OpenGL. The SDK uses a subscription-based purchase model, without requiring any per-application royalties and imposes no branding requirements**.**

Delphi:

Delphi can also be used for creating Android application in the Object Pascal language. The latest release is Delphi 10 Seattle, developed by Embarcadero.User interfaces are developed using the cross-platform GUI framework Firemonkey.

HyperNext:

HyperNext Android Creator (HAC) is a software development system aimed at beginner programmers that can help them create their own Android apps without knowing Java and the Android SDK. It is based on HyperCard that treated software as a stack of cards with only one card being visible at any one time and so is well suited to mobile phone applications that have only one window visible at a time. HyperNext Android Creator's main programming language is simply called HyperNext and is loosely based on Hypercard's HyperTalk language.

Kivy:

Kivy is an open source Python library for developing multitouch application software with a natural user interface (NUI) for a wide selection of devices. Kivy provides the possibility of maintaining a single application for numerous operating systems ("*code once, run everywhere*"). Kivy has a custom-built deployment tool for deploying mobile applications called *Buildozer*, which is available only for Linux.

Lazarus:

The Lazarus IDE may be used to develop Android applications using Object Pascal (and other Pascal dialects), based on the Free Pascal compiler starting from version 2.7.1.

Processing:

The Processing environment, which also uses the Java language, has supported an Android mode since version 1.5; integration with device camera and sensors is possible using the Ketai library.

Qt for Android:

Qt for Android enables Qt 5 applications to run on devices with Android v2.3.3 (API level 10) or later.[41] Qt is a cross-platform application framework which can target platforms such as Android, Linux, iOS, Sailfish OS and Windows. Qt application development is done in standard C++ and QML, requiring both the Android NDK and SDK.[42] Qt Creator is the integrated development environment provided with the Qt Framework for multi-platform application development.

Xamarin:

With a C# shared codebase, developers can use Xamarin to write native iOS, Android, and Windows apps with native user interfaces and share code across multiple platforms. Over 1 million developers use Xamarin's products in more than 120 countries around the world as of May 2015

7.App components:

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.
There are four different types of app components:

- Activities.
- Services.
- Broadcast receivers.
- Content providers.

Activities

An *activity* is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others.

Services

A *service* is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

Broadcast receivers

A *broadcast receiver* is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running.

Content providers

A *content provider* manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it.

ContentResolver

The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the ContentResolver object.

8.Declaring components

```
<?xml  version="1.0" encoding="utf-8"?>
<manifest ... >
   <application android:icon="@drawable/app_icon.png" ... >
      <activity android:name="com.example.project.ExampleActivity"
            android:label="@string/example_label" ... >

</activity>
      ...
   </application>
</manifest>
```

S.no	Directory	Resource
1	res/animator/	XML files that define property animations.
2	res/anim/	XML files for tween animations.
3	res/color/	XML files that define colors and color state list resource.
4	res/drawable/	Bitmap files (.png, .jpg, .gif) or XML files that define drawable resource subtypes.
5	res/layout/	XML files that define a user interface layout.
6	res/menu/	XML files that define application menus.
7	res/raw/	arbitrary files in their raw format (binary or text files)
8	res/values/	XML files that contain simple values, such as strings, integers, and colors. It

		is recommended to use filename conventions for particular types of resources: arrays.xml (typed arrays), colors.xml (color values), dimens.xml (dimension values), strings.xml (string values), styles.xml (styles).

9.Intents:

Intents displays notification messages to the user from within the Android enabled device. It can be used to alert the user of a particular state that occurred. Users can be made to respond to intents.

Activities from Services.

Activities can be closed, or terminated anytime the user wishes. On the other hand, services are designed to run behind the scenes, and can act independently. Most services run continuously, regardless of whether there are certain or no activities being executed.

Items important in every Android project

These are the essential items that are present each time an Android project is created:

– AndroidManifest.xml

– build.xml

– bin/

– src/

– res/

– assets/

importance of XML-based layouts

The use of XML-based layouts provides a consistent and somewhat standard means of setting GUI definition format. In common practice, layout details are placed in XML files while other items are placed in source files.

Containers:

Containers, as the name itself implies, holds objects and widgets together, depending on which specific items are needed and in what particular arrangement that is wanted. Containers may hold labels, fields, buttons, or even child containers, as examples.

Orientation:

Orientation, which can be set using setOrientation(), dictates if the LinearLayout is represented as a row or as a column. Values are set as either HORIZONTAL or VERTICAL.

Importance of Android in the mobile market:

Developers can write and register apps that will specifically run under the Android environment. This means that every mobile device that is Android enabled will be able to support and run these apps. With the growing popularity of Android mobile devices, developers can take advantage of this trend by

creating and uploading their apps on the Android Market for distribution to anyone who wants to download it.

Disadvantages of Android:

Given that Android is an open-source platform, and the fact that different Android operating systems have been released on different mobile devices, there's no clear cut policy to how applications can adapt with various OS versions and upgrades. One app that runs on this particular version of Android OS may or may not run on another version. Another disadvantage is that since mobile devices such as phones and tabs come in different sizes and forms, it poses a challenge for developers to create apps that can adjust correctly to the right screen size and other varying features and specs.

Four essential states of an activity:

– Active – if the activity is at the foreground

– Paused – if the activity is at the background and still visible

– Stopped – if the activity is not visible and therefore is hidden or obscured by another activity

– Destroyed – when the activity process is killed or completed terminated

ANR:

ANR is short for Application Not Responding. This is actually a dialog that appears to the user whenever an application have been unresponsive for a long period of time.

escape characters :

Escape characters are preceded by double backslashes. For example, a newline character is created using '\\n'

settings permissions in app development:

Permissions allow certain restrictions to be imposed primarily to protect data and code. Without these, codes could be compromised, resulting to defects in functionality.

Intent filter:

Because every component needs to indicate which intents they can respond to, intent filters are used to filter out intents that these components are willing to receive. One or more intent filters are possible, depending on the services and activities that is going to make use of it.

10.Three key loops when monitoring an activity:

– Entire lifetime – activity happens between onCreate and onDestroy

– Visible lifetime – activity happens between onStart and onStop

– Foreground lifetime – activity happens between onResume and onPause

OnStop() method :

A call to onStop method happens when an activity is no longer visible to the user, either because another activity has taken over or if in front of that activity.

Different states of a process:

There are 4 possible states:

– foreground activity

– visible activity

– background activity

– empty process.