TOPS Technologies

ANDROID

CONTENTS

INTRODUCTION OF PROGRAMMING

What is Program?

- Sequence of instructions written for specific purpose
- Like program to find out factorial of number
- Can be written in higher level programming languages that human can understand
- Those programs are translated to computer understandable languages
- Task will be done by special tool called compiler
- Compiler will convert higher level language code to lower language code like binary code
- Most prior programming language was 'C'
- Rules & format that should be followed while writing program are called syntax
- Consider program to print "Hello!" on screen

```
void main()
{
            printf("Hello!");
}
```

- Void main(), function and entry point of compilation
- Part within two curly brackets, is body of function
- Printf(), function to print sequence of characters on screen
  Any programming language having three part
  - Data types
  - Keywords
  - Operators
- Data types are like int, float, double
- Keyword are like printf, main, if, else
- The words that are pre-defined for compiler are keyword
- Keywords can not be used to define variable
- We can define variable of data type
  - Like int a; then 'a' will hold value of type int and is called variable
- Programs can have different type of statements like conditional statements and looping statements
- Conditional statements are for checking some condition

CONDITIONAL STATEMENT

- Conditional statements controls the sequence of statements, depending on the condition
- Relational operators allow comparing two values.
  1. == is equal to
  2. != not equal to
  3. < less than
  4. > greater than
  5. <= less than or equal to
  6. >= greater than or equal to

Simple "if" statement

It execute if condition is TRUE

Syntax:

```
if(condition)     {
        Statement1;
        Statement n;
}
```

Example:

```
main()
{
      int a,b;
      printf("Enter a,b values:");
      scanf("%d %d",&a,&b);
      if(a>b) {
            printf("\n a is greater than b"); }
}
```

Output

Enter a,b values:20 10
a is greater than b

If-else Statement

It execute IF condition is TRUE.IF condition is FLASE it execute ELSE part

Syntax:

```
if(condition){
      Statement1; .....
      Statement n;
}else{
      Statement1;       .....
      Statement n;
}
```

Example:

```
main()
{
            int a,b;
            printf("Enter a,b values:");
            scanf("%d %d",&a,&b);
            if(a>b) {
            printf("\n a is greater than b"); }
            else {
            printf("\nb is greater than b"); }
}
```

Output

Enter a,b values:10 20
b is greater than a

If-else if statement

It execute IF condition is TRUE.IF condition is FLASE it checks ELSE IF part .ELSE IF is true then execute ELSE IF PART. This is also false it goes to ELSE part.

Syntax:

```
if(condition) {
       Statementn;
}else if(condition){
      Statementn;
}else{
            Statement n;
}
```

Example:

```
main()
{
      int a,b;
      printf("Enter a,b values:");
      scanf("%d %d",&a,&b);
      if(a>b) {
      printf("\n a is greater than b"); }
      else if(b>a) {
      printf("\nb is greater than b"); }
      else {
      printf("\n a is equal to b"); }
      }
```

Output

Enter a,b values:10 10

a is equal to b

Nested if statement

To check one condition within another.

Take care of brace brackets within the conditions.

Synatax:

```
if(condition){
      if(condition){
            Statement n;
      }
}else{
      Statement n;
}
```

Example:

```
main()
      {
            int a,b;
            printf("\n Enter a and b values:");
            scanf("%d %d ",&a,&b);
            if(a>b) {
                  if((a!=0) && (b!=0)) {
                        printf("\na and b both are +ve and a >b); }
                  else {
                        printf("\n a is greater than b only") ; } }
            else {
                  printf(" \na is less than b"); }
      }
```

Output

Enter a and b values:30 20

a and b both are +ve and a> b

Switch case

The switch statement is much like a nested if .. else statement. switch statement can be slightly more efficient and easier to read.

Syntax :

```
switch( expression ) {
      case constant-expression1:
            statements1; break;
```

```
                        case constant-expression2:
                              statements2; break
                        default :
                              statements4; break;
                  }
```

Example:

```
      main() {
            char Grade = 'B';
            switch( Grade )
            {
                  case 'A' :
                        printf( "Excellent\n" );
                        break;
                  case 'B' :
                        printf( "Good\n" );
                        break;
                  case 'C' :
                        printf( "OK\n" );
                        break;
                  case 'D' :
                        printf( "Mmmmm....\n" );
                        break;
                  default :
                        printf( "What is your grade anyway?" );
                        break;
            }
}
```

Loops

- Loops provide a way to repeat commands and control how many times they are repeated.
- C provides a number of looping way.

while loop

- A while statement is like a repeating if statement.
- Like an If statement, if the test condition is true: the statements get executed.
- The difference is that after the statements have been executed, the test condition is checked again.
- If it is still true the statements get executed again.
- This cycle repeats until the test condition evaluates to false.
- Basic syntax of while loop is as follows:
  while (expression) {
  Single statement or Block of statements;
  }
- Will check for expression, until its true when it gets false execution will be stop

```
main()
{
      int i = 5;
      while ( i > 0 ) {
            printf("Hello %d\n", i );
            i = i -1;
}
}
```

Output

Hello 5

Hello 4

```
Hello 3
Hello 2
Hello 1
```

do … while

- It is just like a while loop except that the test condition is checked at the end of the loop rather than the start.
- This has the effect that the content of the loop are always executed at least once.
- Basic syntax of do...while loop is as follows:

```
do
{
      Single statement or Block of statements;
} while(expression);
```

Example:

```
main()
{
            int i = 5;
            do{
                  printf("Hello %d\n", i );
                  i = i -1; }
            while ( i > 0 );
      }
```

```
Output
Hello 5
Hello 4
Hello 3
Hello 2
Hello 1
```

For loop

- for loop is similar to while, it's just written differently. for statements are often used to proccess lists such a range of numbers:
- Basic syntax of for loop is as follows:

```
for( initialization; condition; increment)
{
      //Single statement or Block of statements;
}
```

Example:

```
main()
{
      int i; int j = 5;
      for( i = 0; i <= j; i ++ ) {
printf("Hello %d\n", i    );
      }
}
```

```
Output
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
```

Break & Continue

- C provides two commands to control how we loop:

- break -- exit form loop or switch.
- continue -- skip 1 iteration of loop.
- Break is used with switch case

```
main()
      {     int i; int j = 5;
            for( i = 0; i <= j; i ++ )
            {   if( i == 3 ) {
                        continue;
                  }
                  printf("Hello %d\n", i );
            }
      }
```

Output

Hello 0
Hello 1
Hello 2
Hello 4
Hello 5

OOPS

- Class – group of data members & member functions
- Like person can be class having data members height and weight and member functions as get_details() and put_details() to manipulate on details
- Class is nothing until you create it's object
- Object – instantiates class allocates memory

OOPS Fundamentals

- Access to data members & member functions can be done using object only (if they are not static!)
- OOPS features are
  - Encapsulation
  - Data hiding
  - Data reusability
  - Overloading (polymorphism)
  - Overriding

Encapsulation

- making one group of data members & member functions
- Can be done through class
- Then group of data members & Member functions will be available just by creating object.

Data Hiding

- can be done through access modifiers
- Access modifiers are private, public, protected and internal
- Private members or member function won't be available outside class
- Public – available all over in program outside class also
- Protected – members that are available in class as well as in it's child class
- Private for another class
- Protected access modifier comes only when inheritance is in picture
- Internal is used with assembly creation

```
class employee {
private:
      char empname[50];
       int empno;
```

```
      public:
            void getvalue()   {
                    cout<<"INPUT Employee Name:";
                    cin>>empname;
                    cout<<"INPUT Employee Number:";
                    cin>>empno;     }
       void displayvalue()       {
                        cout<<"Employee Name:"<<empname<<endl;
                         cout<<"Employee Number:"<<empno<<endl;     }
};
main()
{
      employee e1;
e1.getvalue();
e1.displayvalue();
}
```

Overloading

- taking different output of one method or operator based on parameters and return types
- Like add() method performs addition and add(int a,int b) performs addition of 'a' and 'b' passed when calling
- Also, + operator performs addition of two numbers as well as concatenation of strings

```
class arith
      {
      public:
      void calc(int num1)     {
      cout<<"Square of a given number: " <<num1*num1 <<endl;
      }
       void calc(int num1, int num2 ){
      cout<<"Product of two whole numbers: " <<num1*num2 <<endl;
      }
};
int main() //begin of main function {
    arith a;
    a.calc(5);
    a.calc(6,7);
}
```

Output

Square of given number : 25

Product of two whole numbers : 42

Data Reusability(inheritance)

- helps in saving developers time
- You can use already created class to crate new one
- Called inheritance
- Already existing class is base class and new created is derived class
- Base class members can be available in derived class and to access them create object of derived class
- Like from parent to child

```
class CPolygon
      {
            protected:
```

```
                  int width, height;
            public:
            void set_values (int a, int b)      {
                   width=a; height=b;
            }
      };
class CRectangle: public CPolygon
{
            public: int area () {
             return (width * height);
            }
};
class CTriangle: public CPolygon
{
            public: int area () {
return (width * height / 2);
            }
 };
int main ()
{
            CRectangle rect;
            CTriangle trgl;
rect.set_values(4,5);
            trgl.set_values (4,5);
            cout << rect.area() << endl;
            cout << trgl.area() << endl;
            return 0;
}
```

Overriding

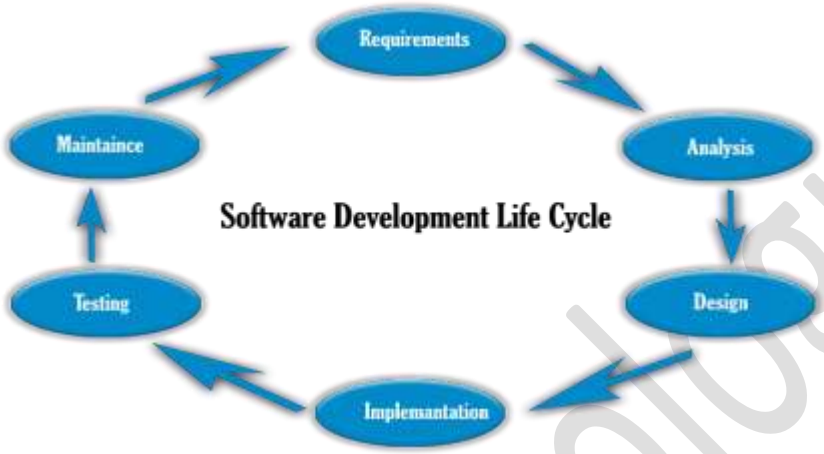
- In C++, overriding is a concept used in inheritance which involves a base class implementation of a method.
- Then in a subclass, you would make another implementation of the method. This is overriding. Here is a simple example.

```
class Base
{
public:
      virtual void DoSomething() {x = x + 5;}
private:
      int x;
};
class Derived : public Base
{
public:
      virtual void DoSomething() { y = y + 5; Base::DoSomething(); }
private:
      int y;
};
```

SDLC PROCESS

- For project development rules & regulation need to be followed for best quality output at defined time limit
- Rules are Software Development Life Cycle – SDLC
- It's part of software engineering

- Six rules to be followed…
- Requirement Gathering
- Analysis & SRS
- Designing
- Implementation (Coding)
- Testing
- Maintenance



Requirement collection

- Phase of collecting requirements from client
- Will be done by business analyst of company
- He will create questioner, in which put answers from client

Analysis & SRS (Software Requirement Specification)

- Collected requirements will be analyzed for time limit, budget and market trade
- Will be filtered and SRS will be created as result
- Will be discussed with client
- Based on requirements users, their activities and flow of data, modules will be defined
- Will be done by system analyst
- Based on diagrams all will be done
- Main diagrams are use-case, DFD and flow charts

Designing

- The Design document should reference what you are going to build to meet the requirements, and not how it can include pseudo code but shouldn't contain actual code functionality.
- Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary.
- These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input. At this phase the test plans are developed.

Implementation (Coding)

- To launch the coding phase, develop a shell program that is then put under some form of version control.
- This phase includes the set up of a development environment, and use of an enhanced editor for syntax checking.

Testing

- Each developer insures that their code runs without warnings or errors and produces the expected results.

- The code is tested at various levels in software testing. Unit, system and user acceptance tasting's are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much if any iteration occurs.
- **Types of testing**: Defect testing, Path testing, Data set testing, Unit testing, System testing, Integration testing, Black box testing, White box testing, Regression testing, Automation testing, User acceptance testing, Performance testing, etc.

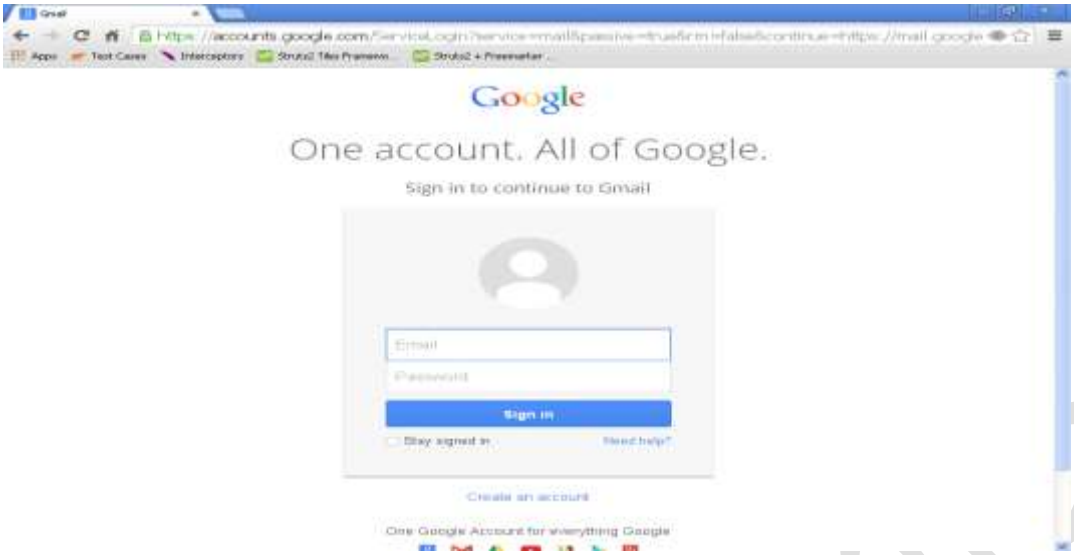
Maintenance

- User's guides and training are developed to reflect any new functionality and changes which need to be identified to the production staff.
- Any changes needed to operations and/or maintenance need to be addressed.
- Every run in production needs to be verified. Any problems with production need to be addressed immediately.
- A Change Request system may be set up to allow for feedback for enhancements.

APPLICATION TYPES

Web Application	Web applications are a whole different beast. Web applications are dynamic and ever-changing. Web applications rely on you to interact with them, whether by contributing content (YouTube, Twitter, Facebook). Like everything on the web, Web applications are built with HTML, CSS & JavaScript, but they also use programming languages like PHP, Ruby, or Python, and frameworks like Rails,Django, and CakePHP. Web applications almost always use databases, and because of that they are called dynamic.
Website	Websites are static, meaning they are not updated, at least not all that often. Websites are simple, single page sites or marketing websites. Websites are built using HTML, CSS, and maybe a little bit of JavaScript. No programming language is required, and neither is a database.
Desktop applications	These applicatios have traditionally been limited by the hardware on which they are run. They must be developed for and installed on a particular operating system, and may have strict hardware requirements that must be met to ensure that they function correctly. Updates to the applications must be applied by the user directly to their installation, and may require hardware upgrades or other changes in order to work.

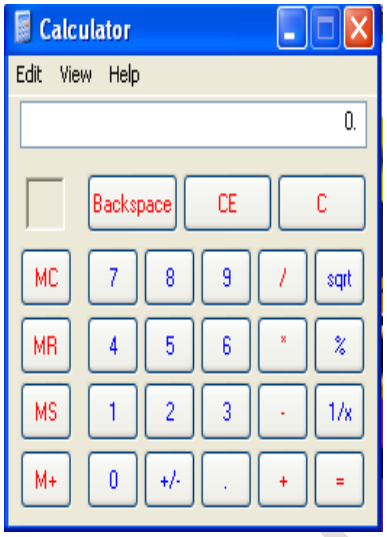
Web Application


Google
One account. All of Google.
Sign in to continue to Gmail
Sign in

Website


www.tops-int.com
TOPS TECHNOLOGIES
Home
Training
Outsourcing
Placements
About Us
Training Progr
Android Training
iPhone Training
PHP Training
SEO Training
Java Training
Asp.NET Training
Software Testing
Web Designing Traini
Silverlight Training
Joomla Training
Magento Training
Request a Call Back
LIVE CHAT
OFFLINE

Desktop Application



Mobile Application

DATABASE[MYSQL]

DBMS

- Database is convenient tool for managing data.
- A database is collection of coherent and meaningful data.
- Advantages of database
  - Redundancy can be reduced.
  - Inconsistency can be reduced.
  - Data can be shared among single and multiple users
  - Standards can be made and can be followed
  - Common security mechanism can be set

❖ RDBMS

- RDBMS – relational database management system
- In RDBMS, all data are stored in tables and relationship between data tables are stored in another tables.
- RDBMS was provided by Dr.E.F.Codd, he has given 12 rules for standard RDBMS. (which has not been met in any database system yet!)

Queries Type

- All SQL Commands are divided into four part based on functionality
  - DDL
  - DML
  - DQL
  - DCL

DDL Statement

- DDL
  - DDL is data definition language.
  - It is used to define structure of database and tables.
  - We can create, modify or delete structure of tables.

DDL Statement	Description	syntax	example
Create	Create command is used to create structure of table.	create table <table_name> (<columnName1><datatype>(<size>));	create table Person_Master(Name nvarchar(50),Age numeric(18,0));
Alter	Once you have defined structure, to modify that Alter command is used.	alter table <tableName>**add**(<NewColumnName><dataType>(<Size>),<NewColumnName><datatype>(<size>)..);	alter table Person_Master add Address nvarchar(50);
		alter table <tableName>**alter** (<columnName><DataType><newSize>));	alter table Person_Master alter column Name varchar(30);
Truncate	Truncate will empty table completely. It will drop table first and then will recreate structure of table.	truncate table <tableName>;	truncate table Person_Master
Drop	This command will discard whole table.	drop table <tableName>;	drop table Person_Master;

DML Statement

- DML is data manipulation language. It is used to manipulate data inside table.

Insert	This command is used to insert records inside table.	insert into <tableName>(columnName1,columnName2,..columnNamen)values(value1,value2,..valuen);	insert into Person_Master(Name,Age,Address)values('Name1',21,'Address1');
Update	This command is used to modify data inside table.	update <tableName> set <columnName1>=value1,<columnName2>=value2..<columnNamen=valuen;	update Person_Master set Name='name11';
Delete	This command is used to delete records from table.	delete from <tableName>;	delete from Person_Master;

DQL Statement

- DQL stands for **Data Query Language.**
- DQL Command is : Select

Select	This will select 'n' columns from table.	select <columnName1>,<columnName2>,..<ColumnNamen> from <tableName>;	select Name,Age,Address from Person_Master;
	To select all records from database,	select * from <tableName>;	select * from Person_Master;

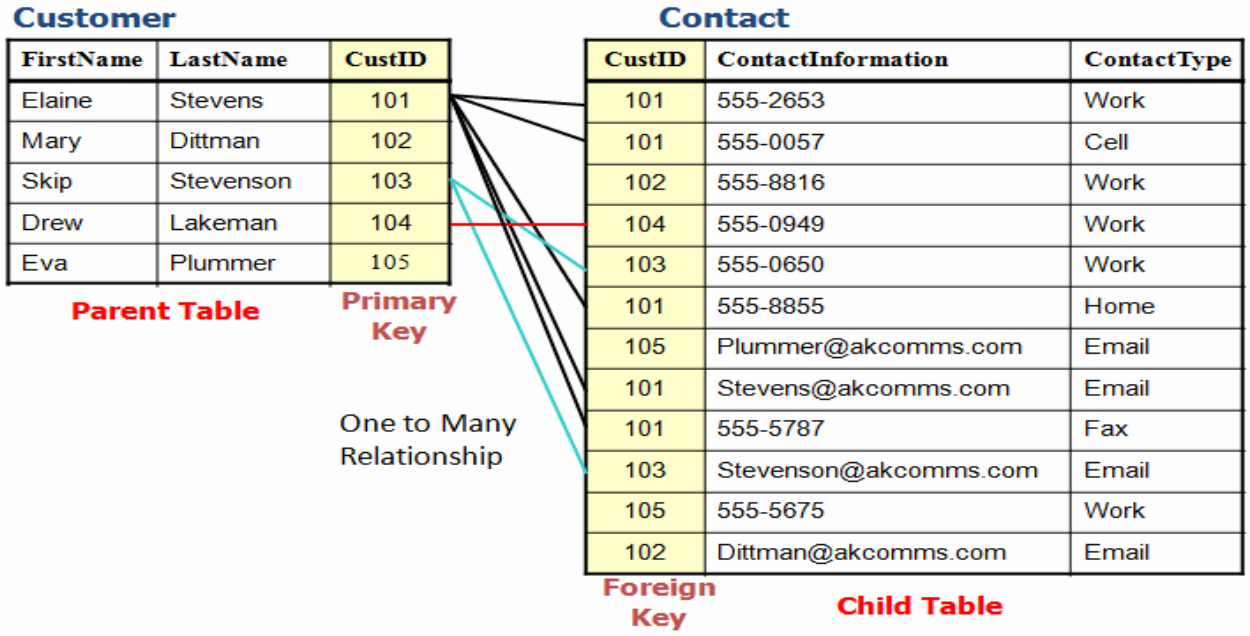
DCL Statement

Commit	Commit is saving of result of all actions or queries.	commit;	START TRANSACTION ; delete from Student_Master where id =102 Commit	Than after execute rollback command it will not show deleted record;
Rollback	Once any transaction has been done, to undo that transaction, rollback can be used.	rollback;	START TRANSACTION delete from Student_Master where id =102 rollback	Deleted record will be roll backed(i.e undo)

Constraints

- To avoid duplication, we need to define constraint on data.
- For example, all students will have unique roll number.
- Like
  - primary key,
  - foreign key,

- unique key etc.



- Primary keys and foreign keys are two types of constraints that can be used to enforce data integrity in SQL Server tables. These are important database objects.
- Primary Key can be defined while creating a table with Create Table command or it can be added with the Alter table command.

Primary key	Foreign key
Primary key uniquely identify a record in the table.	Foreign key is a field in the table that is primary key in another table.
Primary Key can't accept null values.	Foreign key can accept multiple null value.
We can have only one Primary key in a table.	We can have more than one foreign key in a table.
CREATE TABLE Department (DeptID int PRIMARY KEY, Name varchar (50) NOT NULL, Address varchar(100) NULL)	CREATE TABLE Employee (EmpID int PRIMARY KEY, Name varchar (50) NOT NULL, Salary int NULL, DeptID int FOREIGN KEY REFERENCES Department(DeptID))

Primary Key	Unique Key
A primary key *cannot allow* null values. (You cannot define a primary key on columns that allow nulls.)	A unique key *can allow* null values. (You can define a unique key on columns that allow nulls.)
Each table can have *at most one* primary key.	Each table can have *multiple* unique keys.

On some RDBMS a primary key automatically generates a *clustered* table index by default.	On some RDBMS a unique key automatically generates a *non-clustered* table index by default.
CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), PRIMARY KEY (P_Id))	CREATE TABLE Person_Master (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName))

Normalization

- Normalization is the process where a database is designed in a way that removes redundancies, and increases the clarity in organizing data in a database.
- Through normalization, the collection of data in single table will be distributed
- over several tables with specific relationship between them.

Benefits of database normalization in MySQL

- Reduced usage of storage space by intelligently categorizing data is one of Reduced usage of storage space by intelligently categorizing data is one of the many benefits database normalization lends to MySQL.
- It aids in better, faster, stronger searches as it entails fewer entities to scan in comparison with the earlier searches based on mixed entities.
- Data integrity is improved through database normalization as it splits all the data into individual entities yet building strong linkages with the related data.

❖ **Normalize your data easily**

- The following example will illustrate how database normalization helps achieve a good design in MySQL. The table below presents data that needs to be captured in the database.

Title	Author	Bio	ISBN	Subject	Pages	Publisher
Beginning MySQL Database Design and Optimization	Chad Russell, Jon Stephens	Chad Russell is a programmer and system administrator who owns his own internet hosting company. Jon Stephens is a member of the MySQL AB documentation team.	90593324	MySQL Database Design	520	Apress

- In the example shown above, a lot of storage space will be wasted if any one criterion (author or publisher) is considered as the identification key.
- Database normalization, thus, is essential. It is a step by step process that cannot be carried out haphazardly. The following steps will help in attaining database normalization in MySQL.

❖ **Step 1: Create first normal form (1NF)**

- The database normalization process involves getting data to conform to progressive normal forms, and a higher level of database normalization cannot be achieved unless the previous levels have been satisfied. First normal form is the basic level of database normalization.

- One method for bringing a table into 1NF is to separate the entities contained in the table into separate tables. In our case this would result in Book, Author, Subject, and Publisher tables.

- **Book table**

ISBN	Title	Pages
1590593324	Beginning MySQL Database Design and Optimization	520

- **Author table**

Author_ID	First Name	Last Name
1	Chad	Russell
2	Jon	Stephens
3	Mike	Hilyer

- **Subject table**

Subject _ID	Last_name
1	Russell
2	Stephens

- **Publisher Table**

Publisher_ID	Name	Address	City	State	Zip
1	Apress	2 580, Ninth street, station 219	Berkeley	California	94710

❖ **Step 2: Define relationships**

- Three types of relations can be established:
- One-to-(Zero or)-one
- One-to-(Zero or)-many
- Many-to-many
  - The book's table may have many to many relations with the Author's table.
  - Author's table may have many books and a book may have more than one author.
  - The Book's table may have many to many relations with the Subject table.
  - The books may fit in many subjects and the subjects may have many books.
  - Many-to-many relations have to be presented by "link" tables

Book Author table:

ISBN	Subject_ID
1590593324	1
1590593324	2

Book_Subject table:

ISBN	Subject_ID
1590593324	1
1590593324	2

- One-to-many in our example will be Books to Publisher.
- Each book has only one Publisher but one Publisher may have many books.

- We can achieve 'one-to-many' relationships with a foreign key.

ISBN	Title	Pages	Publisher_ID
1590593324	Beginning MySQL Database Design and Optimization	520	1

- A **foreign key** is a mechanism in database management systems (DBMS) that defines relations and creates constraints between data segments.
- It is not possible to review what is not related to the specific book.
- It is not possible to have a book without an author or publisher.
- The foreign key is introduced in the table that represents the "many", pointing to the primary key on the "one" table.
- Since the "Book" table represents the many portion of the one-to-many relationship, the primary key value of the Publisher as in a Publisher_ID column as a foreign key is added.

❖ **Step 3: Make second normal form (2NF)**

- Second normal form (2NF) cuts down the tautological/superfluous data in a table by selecting it, putting it in new tables and by establishing relations amongst them. In database normalization, 2NF is about the relations between the composite key columns and non-key columns.
- That means the non-key columns have to depend on the whole composite key.
- This table does not comply with the 2NF:

ISBN	Reviewer ID	Summary	Reviewer_URL
1590593324	3	A great book!	http://www.openwin.org

❖ **Step 4: Third Normal Form (3NF)**

- This requires that all columns depend directly on the primary key. Tables violate the 3NF when one column depends on another column which in turn depends on the primary key. (A transitive dependency).
- In the publisher table, the City and State are actually dependent on the zip code not the Publisher_ID

Publisher_ID	Name	Address	City	State	Zip
1	Apress	2580, Ninth street, station 219	Berkeley	California	94710

- To comply with 3NF we have to move these outside the publisher's table:

Zip	City	State
94710	Berkeley	California

- Through the process of database normalization we bring our schema's tables into conformance with progressive normal forms.
- As a result the tables each represent a single entity (a book, an author, a subject, etc) and we benefit from decreased redundancy, fewer anomalies and improved efficiency.

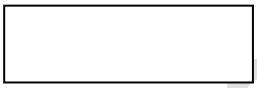
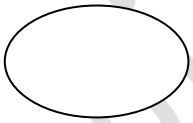
Joins

- Join, Inner Join, Cross Join, Outer Join
- OUTER Join further divides into Left Join, Right Join and Full Join.

Join	• A JOIN command joins/combines rows from more than 1 table. • Two tables that want to join together need to have a common field, like in the following example, department_id is common.	**SELECT** [COL-1], [COL-2] [, ... COL-N] **FROM** [TABLE-1] , [TABLE-2] **WHERE** [TABLE-1].[COLUMN NAME 1] = [TABLE 2].[COLUMN NAME 2]
example	SELECT employee.employee_id, first_name FROM employee , department WHERE employee.department_id = department.department_id	
Inner Join	INNER JOIN or JOIN does the same job.	
example	SELECT first_name,department.department_id FROM employee INNER JOIN department ON department.department_id=employee.department_id	
CROSS JOIN	CROSS JOIN is a query that returns all the records where each row from the table-1 is matched with each row from the table-2.	
example	SELECT employee.employee_id, first_name FROM employee CROSS JOIN department.	
FULL OUTER JOIN:	A FULL OUTER JOIN combines the results of both left and right outer joins and returns all (matched or unmatched) rows from the tables on both sides of the table. FULL JOINs does not work on MySQL but you can emulate it (Use Union)	SELECT [COL-1], [COL-2] [, ... COL-N] FROM [TABLE-1] FULL OUTER JOIN [TABLE-2] ON [TABLE-1].[COLUMN NAME 1] = [TABLE 2].[COLUMN NAME 2]]
example	SELECT first_name,department.department_id FROM employee full outer JOIN department ON department.department_id=employee.department_id	

DFD

- DFD – Data Flow Diagrams
- Graphical representation of flow of data inside application can also be used for visualization and data processing

- DFD elements are..
  - External Entity
  - Process
  - Data Flow
  - Data Store
- External entity
  - Can be user or external system that performs some process or activity in project
  - Symbolized with rectangle
  - Like, we have entity 'admin' then symbol will be 
- Process
  - Work or action taken on incoming data to produce output
  - Each process must have input and output
  - Symbolized as 

Data Flow

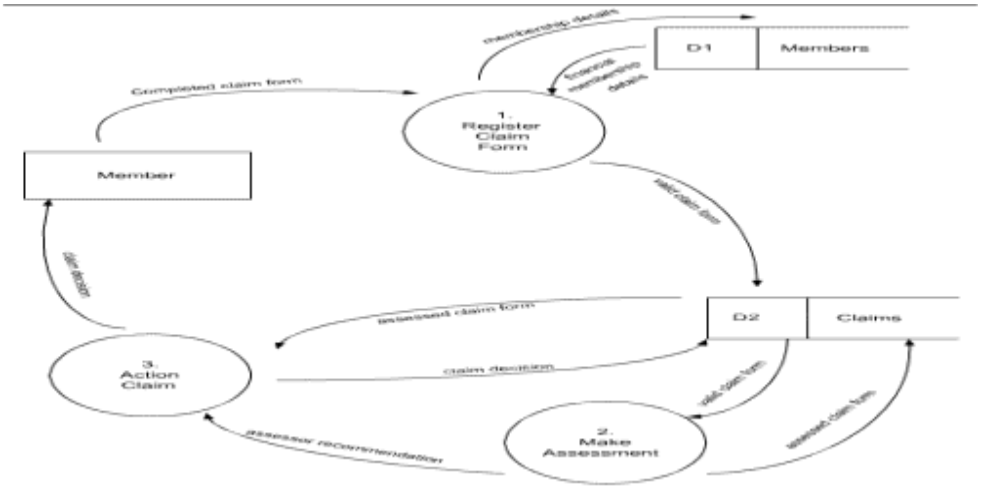
- Can be used to show input and output of data
- Should be named uniquely and don't include word 'data'
- Names can be 'payment', 'order', 'complaint' etc
- Symbolized as 

Data Store

- Can be used to show database tables
- Only process may connect data stores
- There can be two or more process sharing same data store
- Symbolized as 

DFD Rules

- 6 rules to follow
- Consider data flow diagram as given below

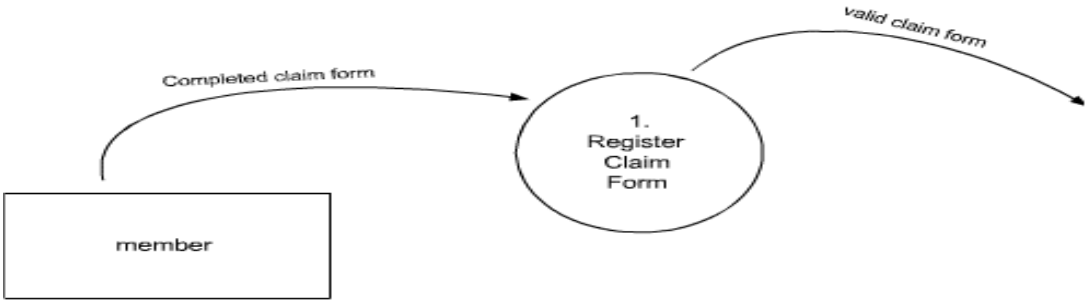


Rule 1

Each process must have data flowing into it and coming out from it

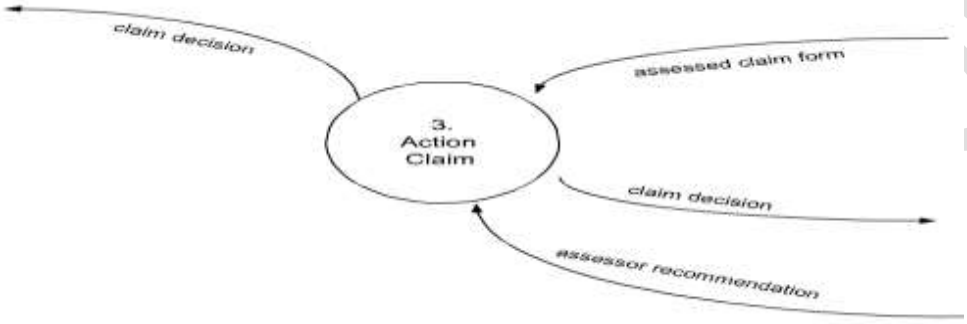
Rule 2

Each data store must have data going inside and data coming outside



Rule 3

A data flow out of a process should have some relevance to one or more of the data flows into a process

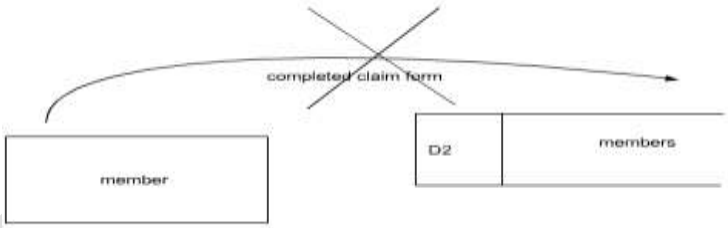


In process 3 all data flows are connected to process claim.

The claim decision can not be made until the claim form has been submitted and the assessor makes a recommendation

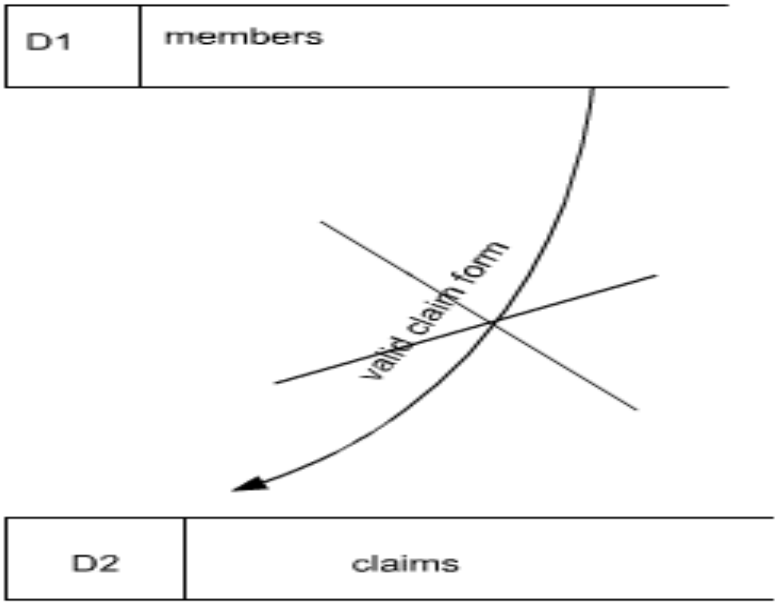
Rule 4

Data stored in system must go through a process



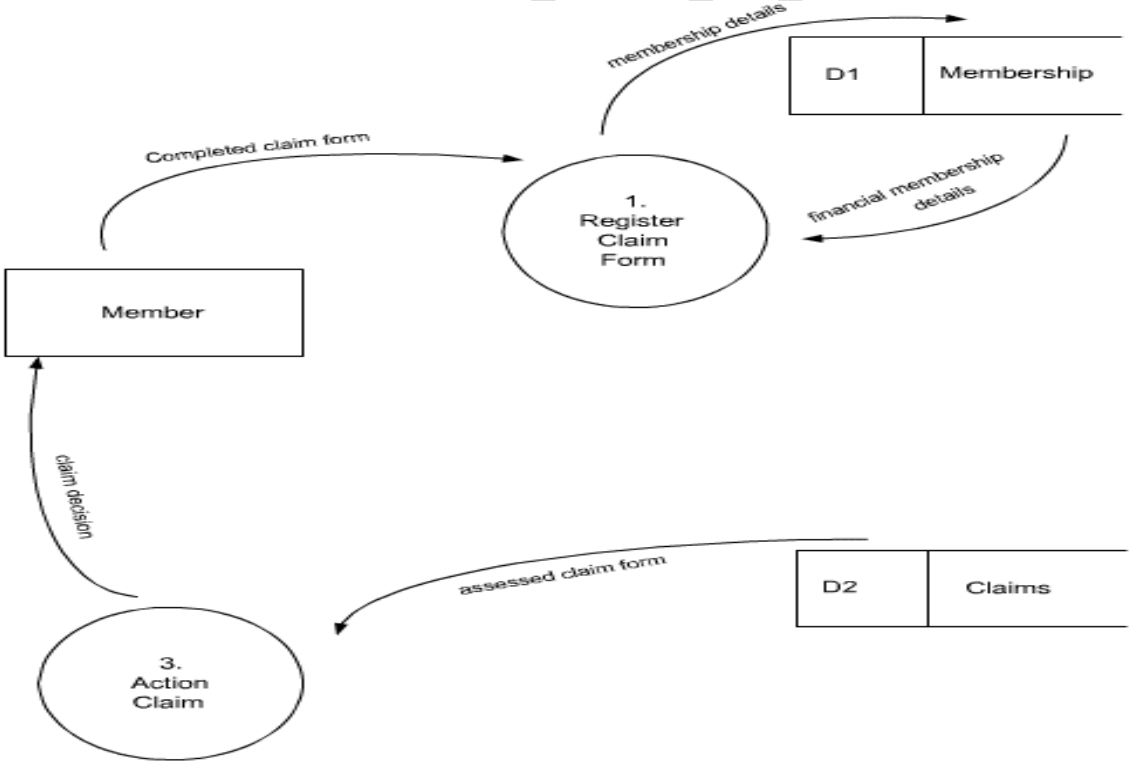
Rule 5

Two data stores can't communicate with each other unless process is involved in between



Rule 6

- The Process in DFD must be linked to either another process or a data store.
- A process can't be exist by itself, unconnected to rest of the system



Context Level

- DFD level-0
- It's also context level DFD
- Context level diagrams show all external entities.
- They do not show any data stores.

- The context diagram always has only one process labelled 0.

DFD Level-1(or 2)

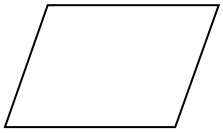
- Include all entities and data stores that are directly connected by data flow to the one process you are breaking down show all other data stores that are shared by the processes in this breakdown
- Like login process will linked to users & database in further leveling

FLOW CHARTS

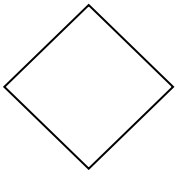
- Used to show algorithm or process
- Can give step by step solution to the problem
- The first flow chart was made by John Von Newman in 1945
- Pictorial view of process
- Helpful for beginner and programmers
- Flowcharts are generally drawn in the early stages of formulating computer solutions.
- Flowcharts facilitate communication between programmers and business people.
- These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems.
- Once the flowchart is drawn, it becomes easy to write the program in any high level language.
- Often we see how flowcharts are helpful in explaining the program to others.
- Hence, it is correct to say that a flowchart is a must for the better documentation of a complex program.
- Symbols are..
    - Start Or End
    - Show starting or ending of any flow chart
    - Symbolized as

    - Process
    - Defines a process like defining variables or initializing variable or performing any computation
    - Symbolized as

    - Input or Output
    - Used when user have to get or initialize any variable
    - Like get num1 and num2 from user
    - Symbolized as



    - Decision Making
    - For checking condition this symbols can be used
    - Like if num1 is greater than num2
    - Can be symbolized as



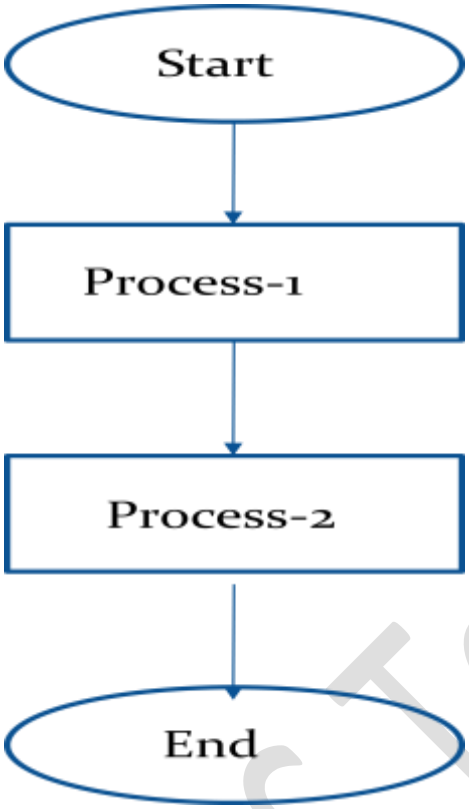
Flow lines

- Flow lines
- Lines showing flow of data and process

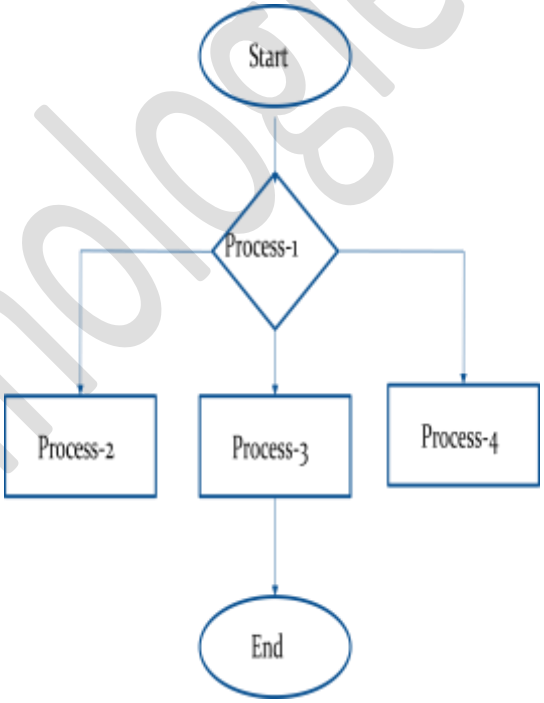
- Showing flow of instructions also
- Can be symbolized as ⟶

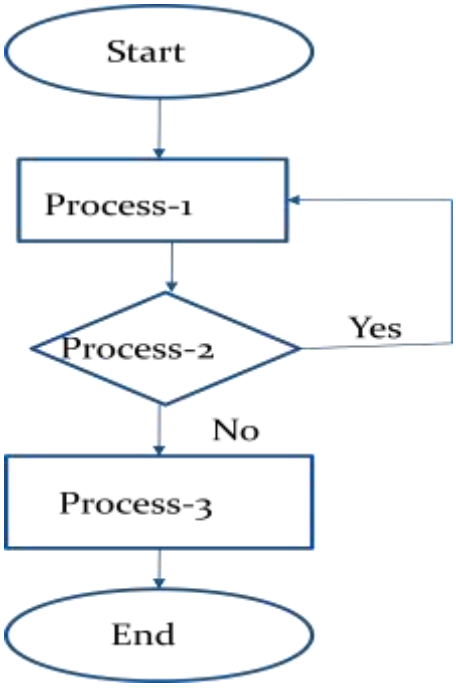
- Any program can be in three format
  - Linear or sequence
  - Branching
  - Looping
- Following are notations can be used to show this format

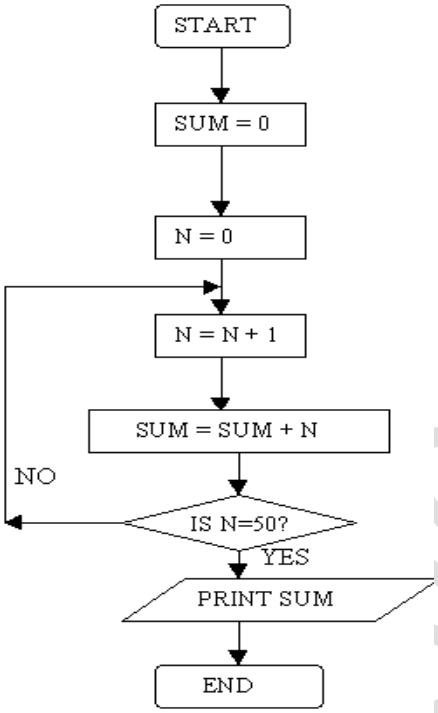
Linear Program Structure



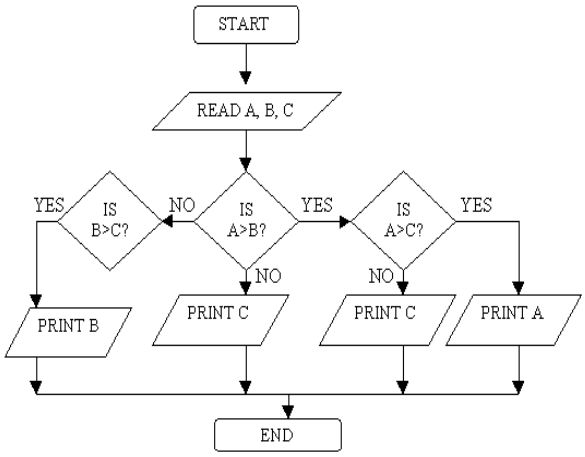
Branching Program Structure..



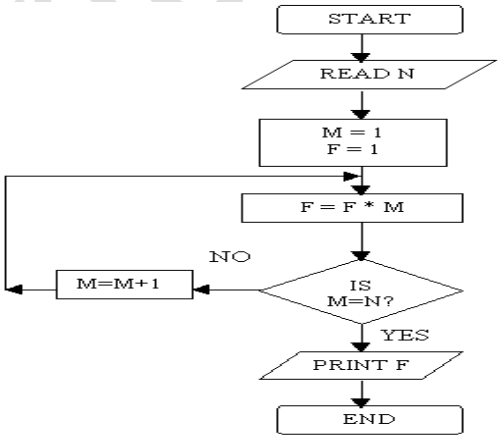




Looping Program Structure Sum of 1-50 numbers..



Max among three numbers…



Factorial of number

INTRODUCTION TO ANDROID

Android Facts and Figures

- The first Android-powered phone was launched on October 2008, T-Mobile G1
- By the end of 2009 over 20 Android-compatible handsets had been launched or announced in more than 26 countries on 32 different carrier networks.
- End of 2010 Android had become the world's leading smartphone platform
- 1.5 million Android devices activited every day, YES EVERY DAY
- From 2015 Google's mobile platform now has 1.4 billion 30 days active user globally.
- There Number of App available in the play store 1 million in July 2013 and was most recently Placed at 1.6 million apps in July 2015
- Estimated number of applications downloaded from Google Play in 2015 93billion

History

- **Android** is a software stack for mobile devices that includes an operating system, middleware and key applications

- Android is a Linux-based operating system designed primarily for touchscreen mobile devices
- Initially developed by Android Inc.
- Android, Inc. was founded in Palo Alto, California in October 2003 by
    - Andy Rubin (co-founder of Danger)
    - Rich Miner (co-founder of Wildfire Communications, Inc.)
    - Nick Sears (once VP at T-Mobile)
    - Chris White (headed design and interface development at WebTV)
- Purchesed by "Google" in 2005
- Android was unveiled on 5th November, 2007 to the world along with the founding of the "Open Handset Alliance"
- Android Code as Open Source, written primarily in a customized version of "Java"

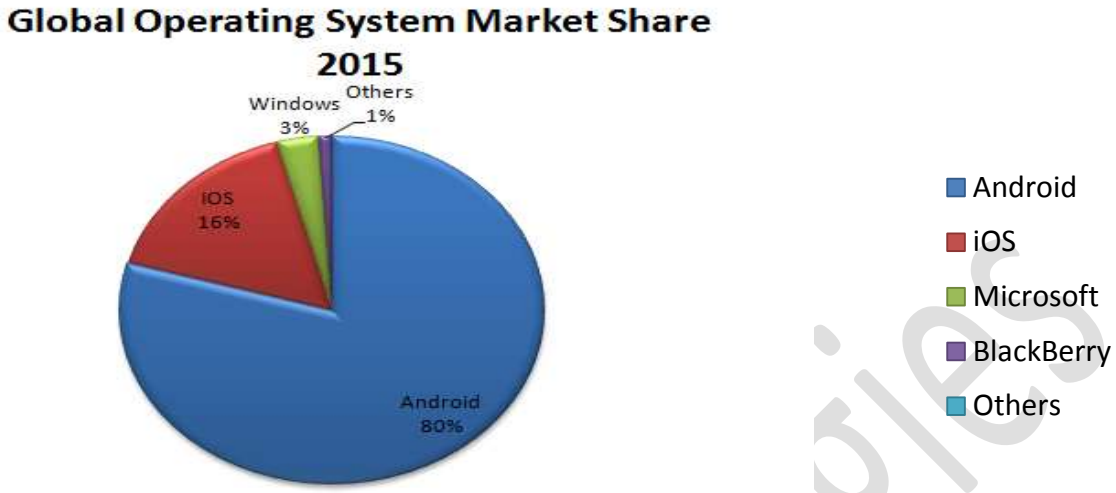
Open Handset Allience

The Open Handset Alliance (OHA) is a collection of more than 84 technology companies, including hardware manufacturers, mobile carriers, and software developers. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm. In their own words, the OHA represents the following:
A commitment to openness, a shared vision for the future, and concrete plans to make the vision a reality. To accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. (http://www .openhandsetalliance.com/)
The OHA hopes to deliver a better mobile software experience for consumers by providing the platform needed for innovative mobile development at a faster rate and with higher quality than existing platforms, without licensing fees for either software developers or handset manufacturers.

Android Devices

The open and customizable nature of Android allows it to be used on other electronics aside from smart phones and tablets, including laptops and netbooks, smartbookssmart TVs (Google TV) and cameras (Nikon Coolpix S800c and Galaxy Camera). In addition, the Android operating system has seen applications on smart glasses (Google Glass), wristwatches, headphones, car CD and DVD players, mirrors, portable media playersand landlines and Voice over IP phones. Ouya, a video game console running Android, became one of the most successfulKickstarter campaigns, crowdfunding US$8.5m for its development, and was later

followed by other Android-based consoles, such as Nvidia's Project Shield—an Android device in a video game controller form factor.



Android OS Architecture

Linux Kernel Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.

❑**Libraries** Running on top of the kernel, Android includes various C/C++ core libraries such as libc and SSL, as well as:

- ❑A media library for playback of audio and video media
- ❑A Surface manager to provide display management
- ❑Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
- SQLite for native database support
- SSL and WebKit for integrated web browser and Internet security

❑**Android Run Time** what makes an Android phone an Android phone rather than a mobile Linux implementation is the Android run time. Including the core libraries and the Dalvik virtual machine, the Android run time is the engine that powers your applications and, along with the libraries, forms the basis for the application framework.

❑**Core Libraries** While Android development is done in Java, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android-specific libraries.

❑**Dalvik Virtual Machine** Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances effi ciently. It relies on the Linux kernel for threading and low-level memory management.

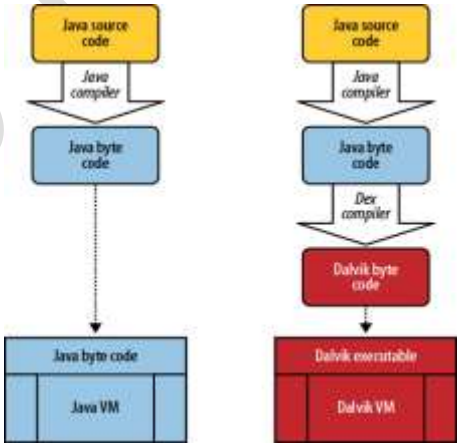
❑**Application Framework** The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.

❑**Application Layer** All applications, both native and third party, are built on the application layer using the same API libraries. The application layer runs within the Android run time using the classes and services made available from the application framework.

DEVELOPMENT WITH ANDROID

Delvik Virtual Machine

- One of the key elements of Android is the Dalvik virtual machine. Rather than use a traditional Java virtual machine (VM) such as Java ME (Java Mobile Edition), Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.
- The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management.
- All Android hardware and system service access is managed using Dalvik as a middle tier. By using a VM to host application execution, developers have an abstraction layer that ensures they never have to worry about a particular hardware implementation.
- The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory footprint. You create.dex executables by transforming Java language compiled classes using the tools supplied within the SDK.



What Comes in the Box

The Android software development kit (SDK) includes everything you need to start developing, testing, and debugging Android applications. Included in the SDK download are:

❑**The Android APIs** The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries used at Google to create native Android applications.

❑**Development Tools** to turn Android source code into executable Android applications, the SDK includes several development tools that let you compile and debug your applications.

❑**The Android Emulator** The Android Emulator is a fully interactive Android device emulator featuring several alternative skins. Using the emulator, you can see how your applications will look and behave on a real Android device.

❑**Full Documentation** The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them. In addition to the code documentation, Android's reference documentation explains how to get started and gives detailed explanations of the fundamentals behind Android development.

❑**Sample Code** The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available using Android, as well as simple programs that highlight how to use individual API features.

❑**Online Support** Despite its relative youth, Android has generated a vibrant developer community. The Google Groups at http://code.google.com/android/groups are active forums of Android developers with regular input from the Android development team at Google.

Setup Development Environment

What You Need to Begin

- **SYSTEM REQUIREMENTS**
  - Operating Systems
    - Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
    - Mac OS X 10.5.8 or later (x86 only)
    - Linux (tested on Ubuntu Linux, Lucid Lynx)
  - Android StudioIDE
    - Android Studio (Version 1.3 and above)
    - JDK required for Java Support
      JDK 7 (JRE alone is not sufficient)
- To setup java in your computer you must download java from the oracle site here is the download link
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- *If you already have a JDK installed, make sure that it meets the version requirements listed above, and note that the Java runtime environment (JRE) is not sufficient.*
- *Downloading and Installing the SDK*
  http://developer.android.com/sdk/index.html
- Download android-sdk_r<Version>-windows.zip file

ZIP file containing the API libraries, developer tools, documentation, and several sample applications and API demos that highlight the use of particular API features. Install it by unzipping the SDK into a SDK in C drive.

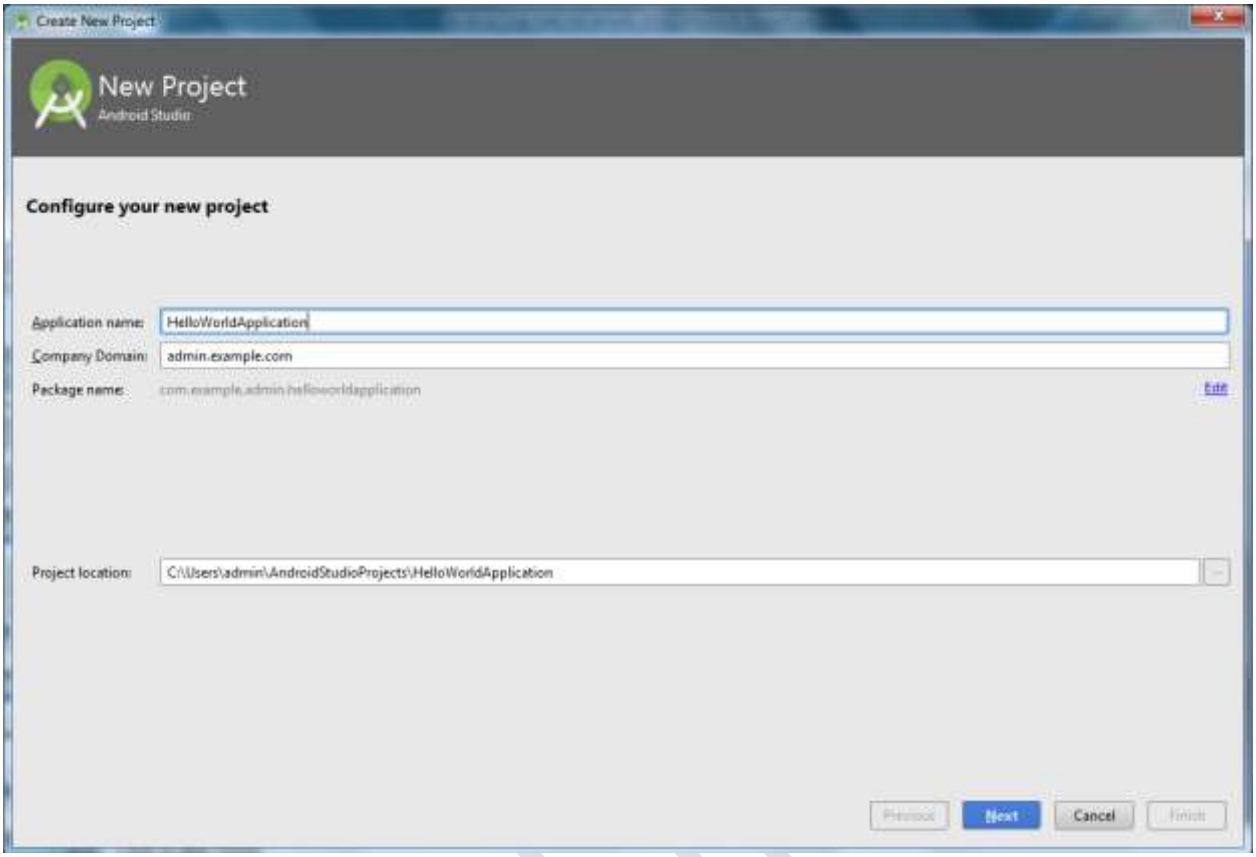
Developing with Android Studio

- First Download Android Studio from **http://developer.android.com/sdk/index.html**
- Download and Unzip this file in to your directory (C:\).
- Double-click on the file C:\android-studio\bin\studio.exe (for 32-bit machine) or C:\android-studio\bin\studio64.exe (for 64-bit machine).

- Click Ok and then click
- You can select Standard for making a normal installation where the sdk will be downloaded in it and be placed in the default location
- If you select Custom setup you will be able to select the theme of android studio and give location for downloading android or if any sdk source is existing then the same sdk source location can be specified.

Creating Hello World

- After you've created an AVD, the next step is to start a new Android project in Eclipse.
- **From Android Studio, select Start a new Android Studio project**
- **Fill in the project details as shown in below Example**

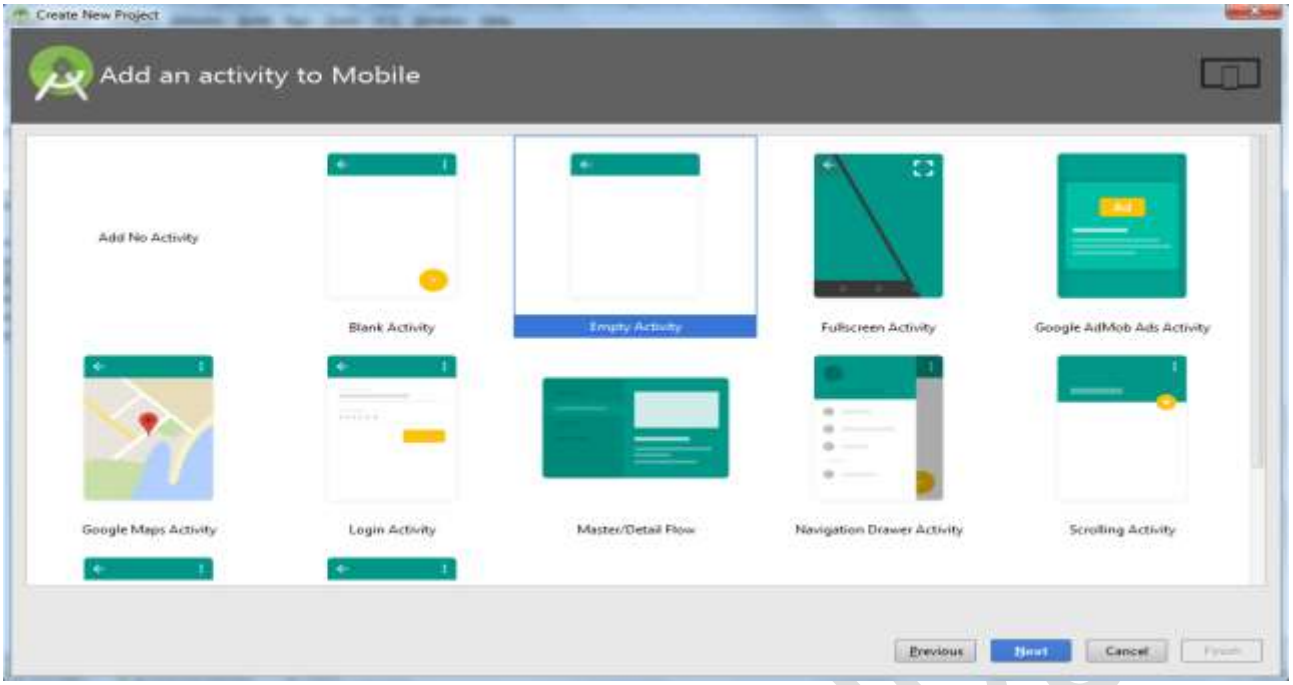


Here is a description of each field:

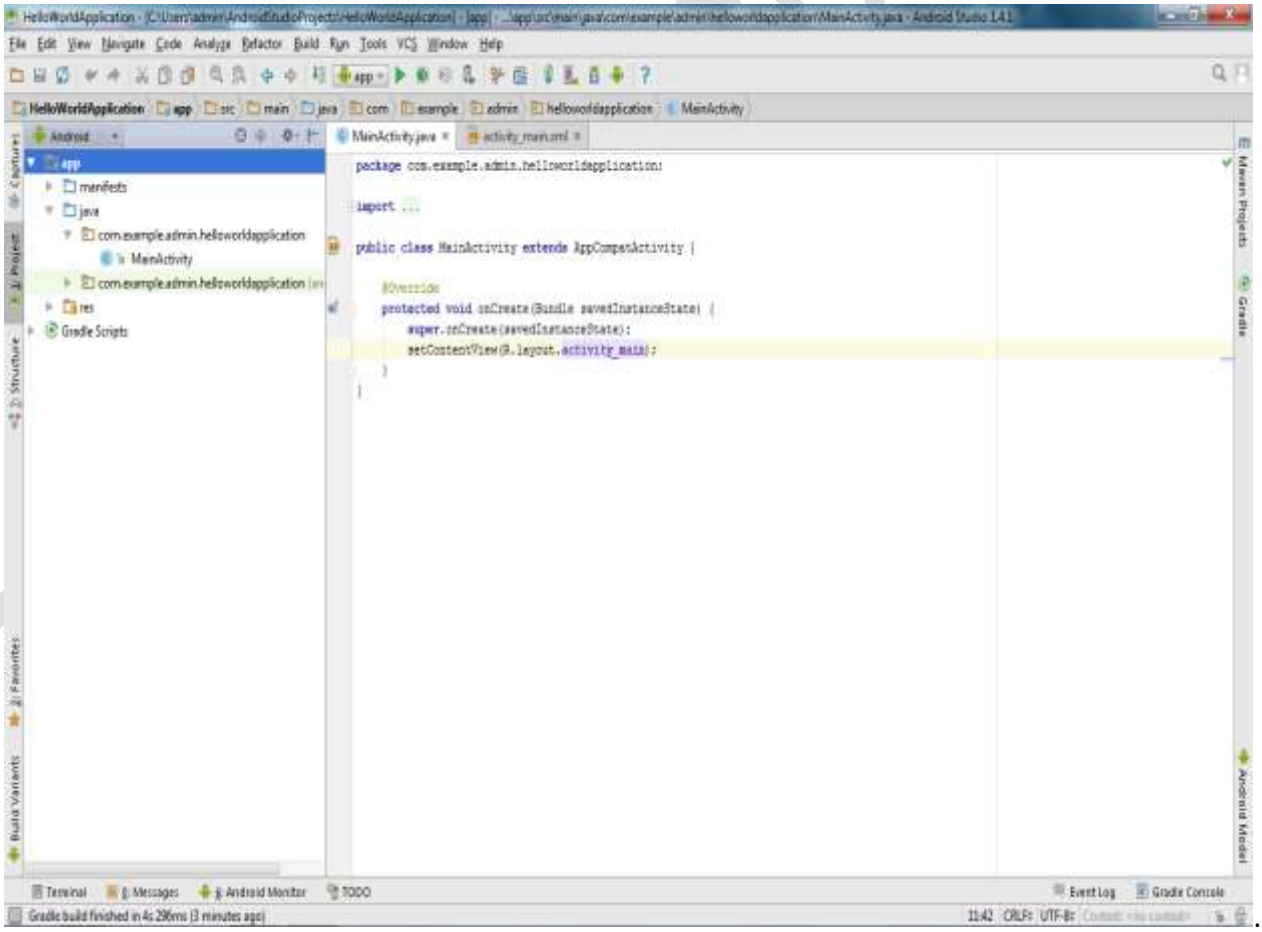
- **Application Name: :** This is the title for your application — the name that will appear on the Android device
- **Package Name:** This is the package namespace (following the same rules as for packages in the Java programming language) that you want all your source code to reside under. This also sets the package name under which the stub Activity will be generated.
- **Compony Domain:** The company domain name is used to uniquely identify your app. Thus, even if two different companies create Android apps with the same name, you can still see which is which because their company domain names will be different. If you have a company, type in the company's domain name in reverse (e.g. com.andro). If you have no company, you can just choose a fictive company domain name.
- **Min SDK Version:** This value specifies the minimum API Level required by your application. For more information, see Android API Levels.
- **Other fields::** The checkbox for "Use default location" allows you to change the location on disk where the project's files will be generated and stored. "Build Target" is the platform target that your application will be compiled against (this should be selected automatically, based on your Min SDK Version).Click on Next Button

After that open following Window

Select 'EmptyActivity' and press **Next**

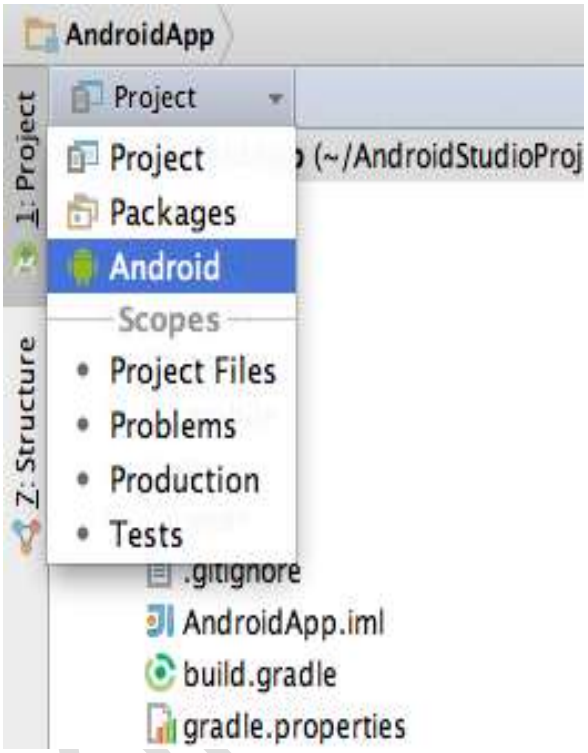


- **You will see the following files created automatically by the SDK.**

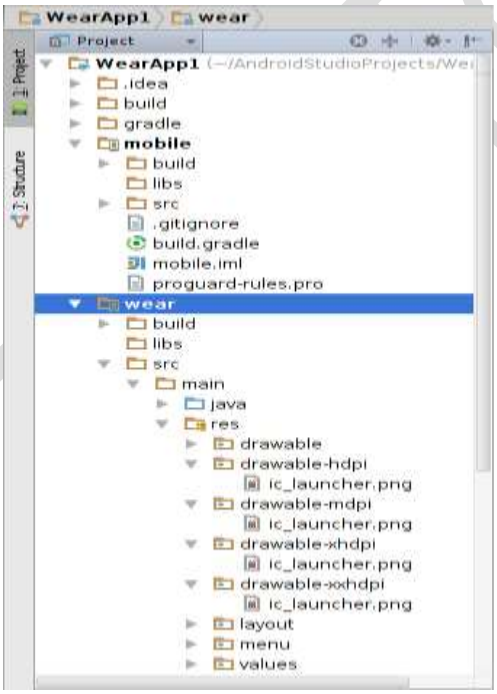


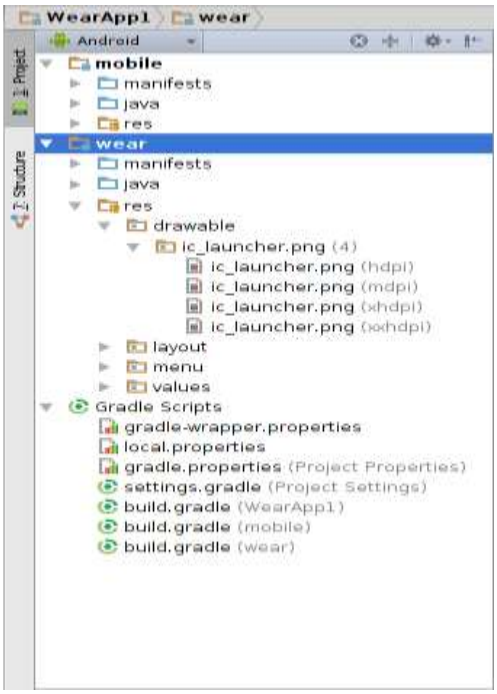
Different Views in Android Studio

- The Android project view shows all the build files at the top level of the project hierarchy under Gradle Scripts. Each project module appears as a folder at the top level of the project hierarchy and contains these three elements at the top level:
  - java/ - Source files for the module.
  - manifests/ - Manifest files for the module.
  - res/ - Resource files for the module.
  - applications.



- Note:The Android project view shows a hierarchy that helps you work with Android projects by providing a flattened structure that highlights the most commonly used files while developing Android applications. However, the project structure on disk differs from this representation.





- **src :** It contains the source packages and java source files. In our src folder it currently contains the package com.helloAndroid. The package further contains the java file “HelloAndroid.java”.

- **.idea:** This where project specific metadata is stored by Android Studio (AS). (**Eclipse Land:** project.properties file)

- **. gradle:** This is where the gradle build system's jar wrapper i.e. this jar is how AS communicates with gradle installed in Windows (the OS in my case).

- **. build:** This has all the complete output of the make process i.e. classes.dex, compiled classes and resources, etc.

- In the Android Studio GUI, only a few folders are shown. The important part is that **your R.java is found here** under **build/source/<flavor>/r/<build type(optional)>/<package>/R.java**

- **res :** It is one of the other important content folders.
  It contains three subfolders for images
  drawable –hdpi
  drawable –ldpi
  drawable –mdpi
  drawable-xhdpi
    - **layout :** The layout contains the main.xml which is called when the application is started.
    - **values :** Contained in the values folder is used to define strings to be used within the applications.
    - **mipmap**: The mipmap folders are for placing your app icons in only. Any other drawable assets you use should be placed in the relevant drawable folders .
- **manifests :**
    - **Android Manifest.xml:** (see about it later.)

Open the **HelloAndroid.java** file,

- located inside *HelloAndroid > src > com.example.helloandroid*). It should look like this:

```
package tops.example.helloandroid;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class HelloAndroid extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_hello_android);
        }
}
```

The Android Virtual Device

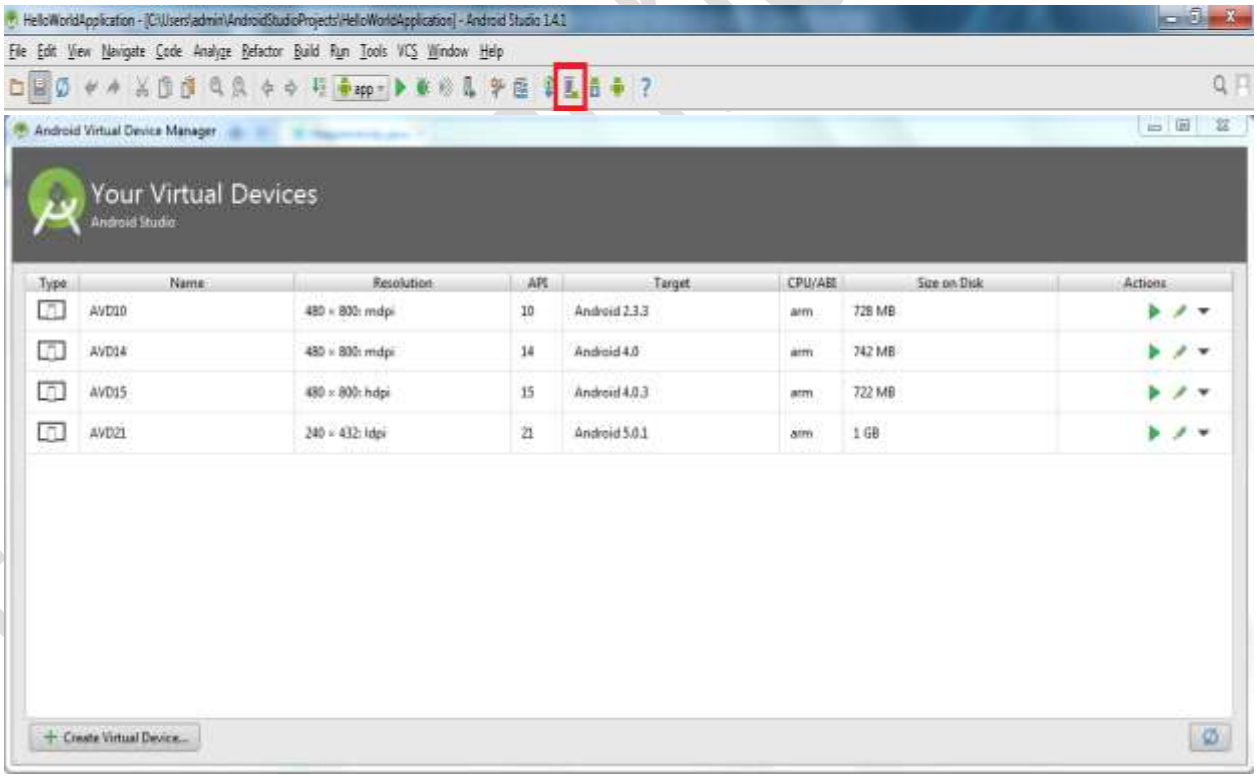
- The Virtual Device and SDK Manager is a tool used to create and manage the virtual devices that will host instances of your emulator. You can use the same tool both to see which version of the SDK you have installed and to install new SDKs when they are released.

- **Android Virtual Devices**
- Android Virtual Devices are used to simulate the software builds and hardware specifications available on different devices. This lets you test your application on a variety of hardware platforms without needing to buy a variety of phones.

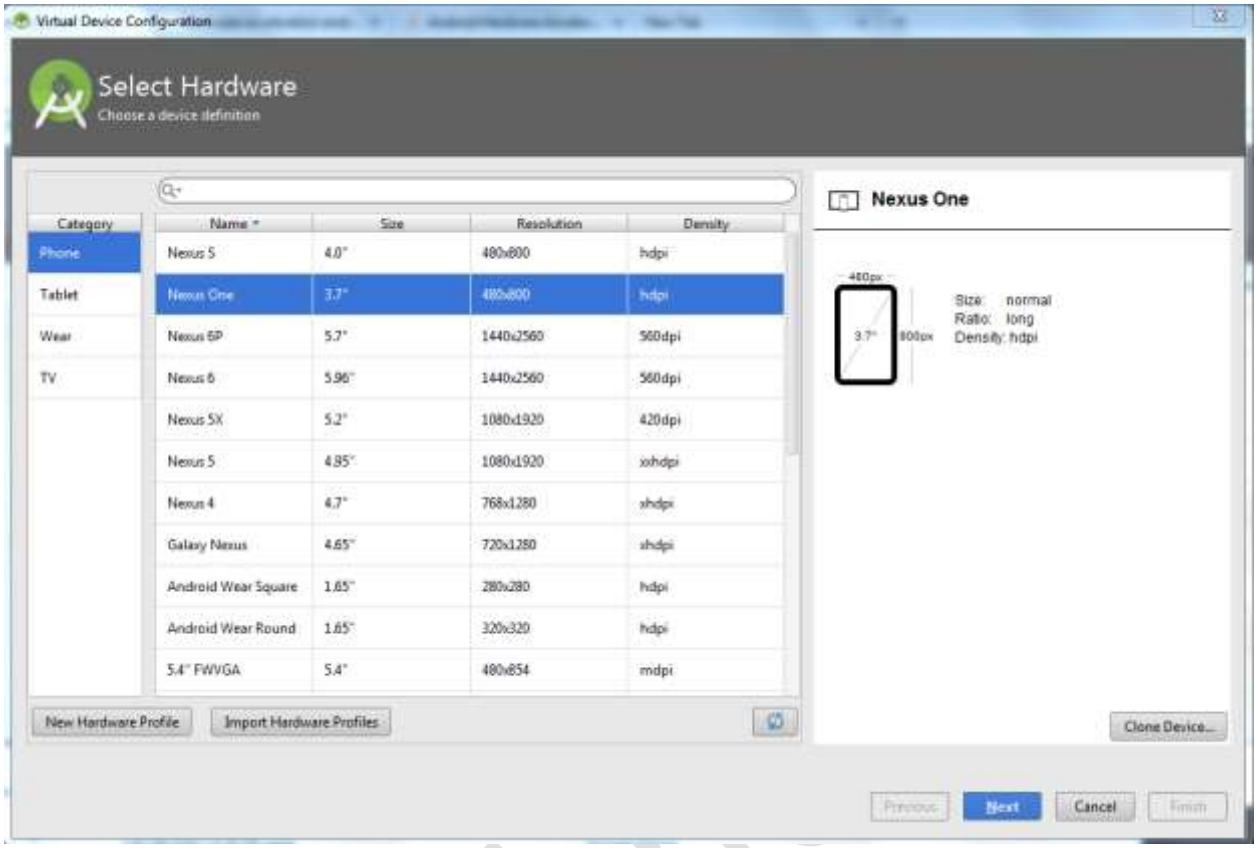
- Each virtual device is configured with a name, a target build of Android (based on the SDK version it supports), an SD Card capacity, and screen resolution, as shown in the ‘‘Create new AVD’’ dialog in Figure.
- Each virtual device also supports a number of specific hardware settings and restrictions that can be added in the form of NVPs in the hardware table. These additional settings include:
    - Maximum virtual machine heap size
    - Screen pixel density
    - SD Card support
    - The existence of DPad, touchscreen, keyboard, and trackball hardware
    - GPS support
    - Available device memory
    - Camera hardware (and resolution)
    - Support for audio recording
- Different hardware settings and screen resolutions will present alternative user-interface skins to represent the different hardware configurations. This simulates a variety of mobile device types. To complete the illusion, you can create a custom skin for each virtual device to make it look like the device it is emulating.

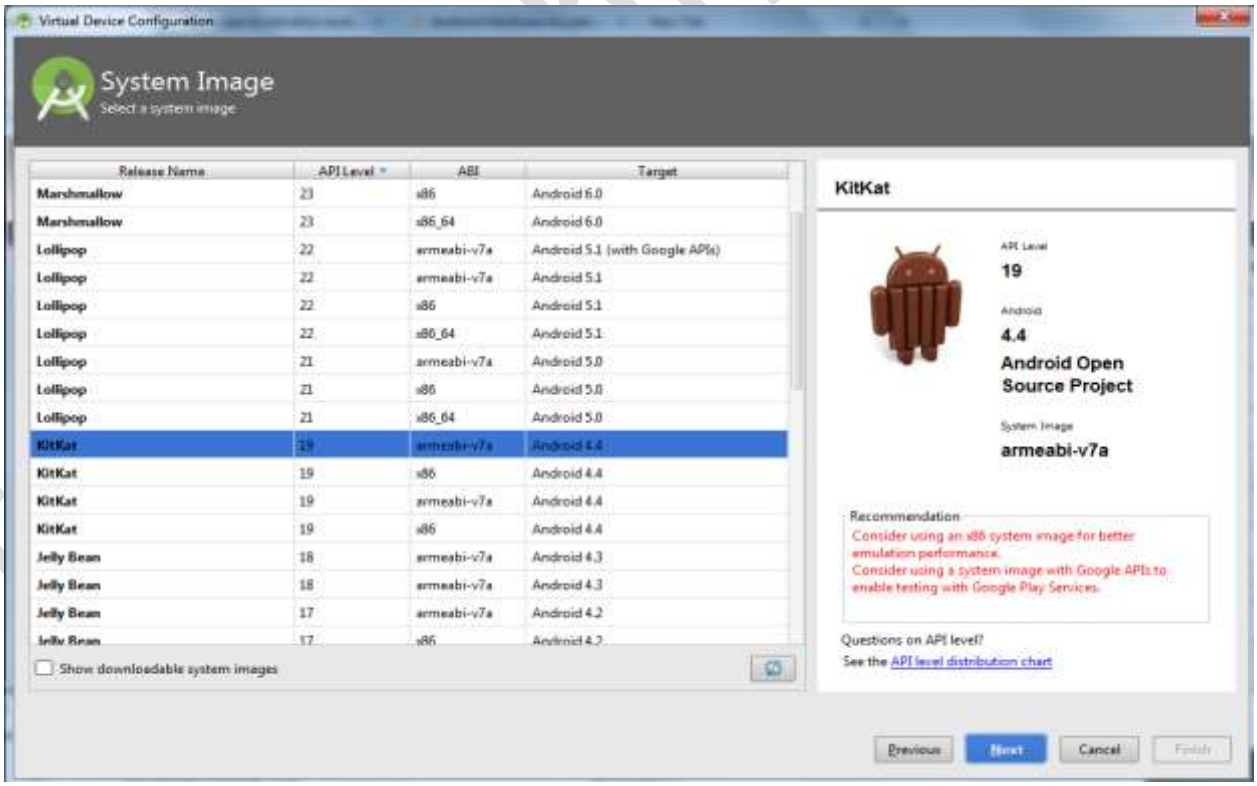
How Create Android virtual Device (AVD)

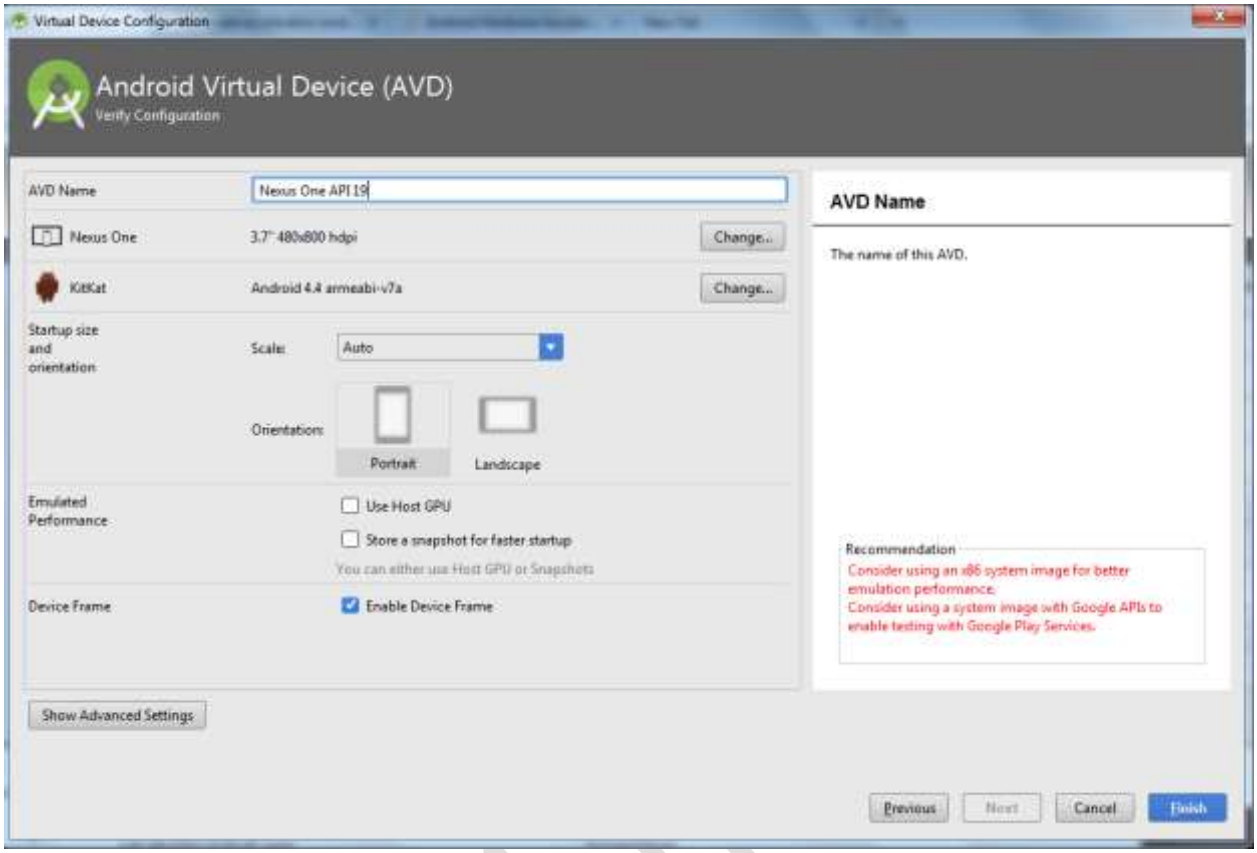
- You need to define a device which can be used for emulation. Press the device manager button, press “Create New Virtual Device“ to create new AVD.



Add This Information


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
Phone
Tablet
Wear
TV
Name
Size
Resolution
Density
Nexus S 4.0" 480x800 hdpi
Nexus One 3.7" 480x800 hdpi
Nexus 6P 5.7" 1440x2560 560dpi
Nexus 6 5.96" 1440x2560 560dpi
Nexus 5X 5.2" 1080x1920 420dpi
Nexus 5 4.95" 1080x1920 xxhdpi
Nexus 4 4.7" 768x1280 xhdpi
Galaxy Nexus 4.65" 720x1280 xhdpi
Android Wear Square 1.65" 280x280 hdpi
Android Wear Round 1.65" 320x320 hdpi
5.4" FWVGA 5.4" 480x854 mdpi
Nexus One
480px
800px
3.7"
Size: normal
Ratio: long
Density: hdpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish


Virtual Device Configuration
System Image
Select a system image
Release Name
API Level
ABI
Target
Marshmallow 23 x86 Android 6.0
Marshmallow 23 x86_64 Android 6.0
Lollipop 22 armeabi-v7a Android 5.1 (with Google APIs)
Lollipop 22 armeabi-v7a Android 5.1
Lollipop 22 x86 Android 5.1
Lollipop 22 x86_64 Android 5.1
Lollipop 21 armeabi-v7a Android 5.0
Lollipop 21 x86 Android 5.0
Lollipop 21 x86_64 Android 5.0
KitKat 19 armeabi-v7a Android 4.4
KitKat 19 x86 Android 4.4
KitKat 19 armeabi-v7a Android 4.4
KitKat 19 x86 Android 4.4
Jelly Bean 18 armeabi-v7a Android 4.3
Jelly Bean 18 armeabi-v7a Android 4.3
Jelly Bean 17 armeabi-v7a Android 4.2
Jelly Bean 17 x86 Android 4.2
Show downloadable system images
KitKat
API Level
19
Android
4.4
Android Open Source Project
System Image
armeabi-v7a
Recommendation
Consider using an x86 system image for better emulation performance.
Consider using a system image with Google APIs to enable testing with Google Play Services.
Questions on API level?
See the API level distribution chart
Previous
Next
Cancel
Finish



- Your AVD started..

Run the Application

- In android studio to run the application
- First select your application
- Then select on the green symbol to run your application.

Introducing the Application Manifest

- AndroidManifest.xml, which is similar to the web.xml file in the J2EE world, defines the contents and behavior of your application. For example, it lists your app's activities and services, along with the permissions the application needs to run.
- Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.
- The important elements in the manifest file are:
  - Manifest
  - Application
  - Permissions
  - Instrumentation
  - AndroidManifest.xml

Using the Manifest Tab

- The Manifest tab (Below Figure) contains package-wide settings, including the package name, version information, and minimum Android SDK version information.

- You can also set any hardware configuration requirements here.

Using the Application Tab

- The Application contains application-wide settings, including the application label and icon, as well as information about application components such as activities, intent filters, and other application functionality, including configuration for service and content provider implementations.

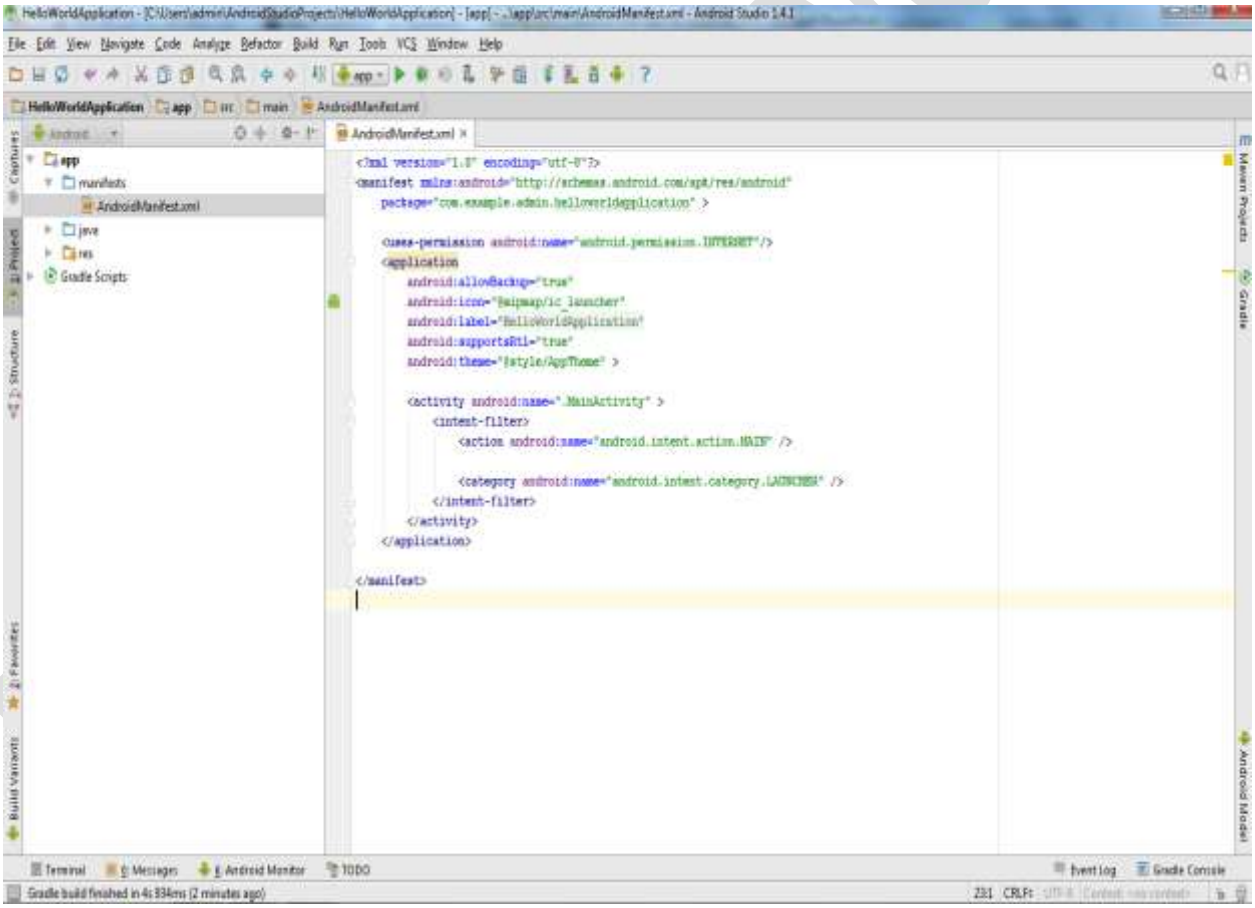
Using the PermissionTab

- The Permissions tab (Below Figure) contains any permission rules required by the application. This tab can also be used to enforce custom permissions created for the application.

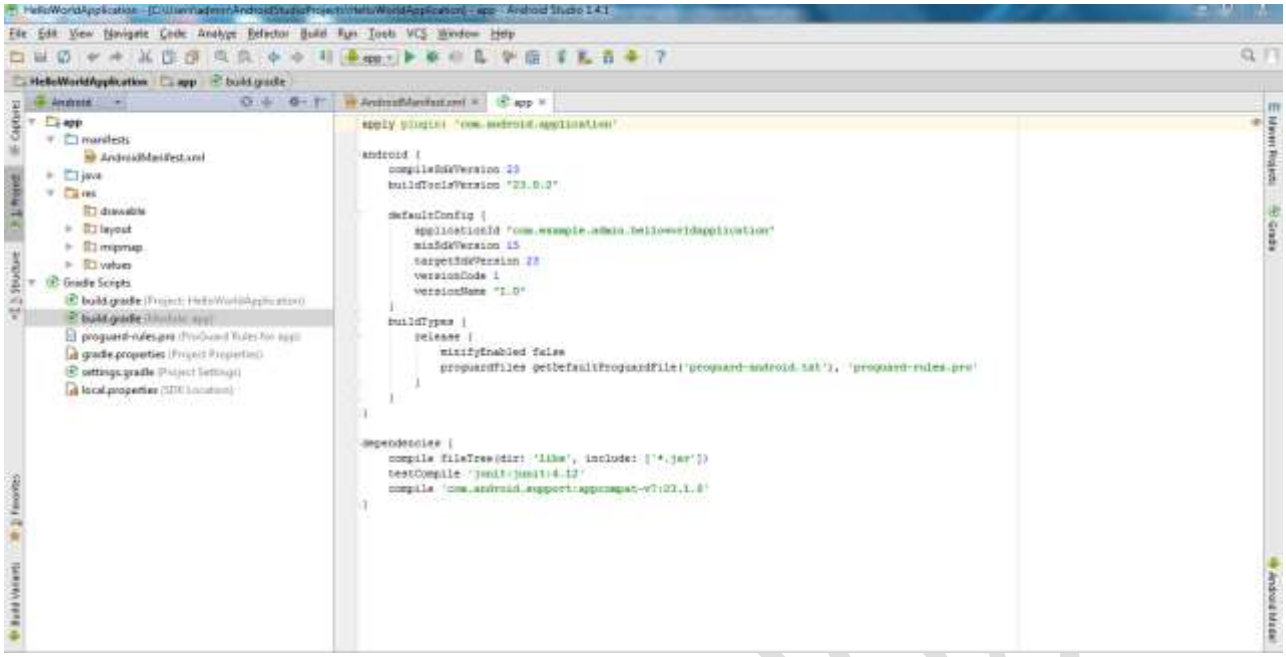
Using the Instrumentation Tab

- You can use the Instrumentation tab (Below Figure) to declare any instrumentation classes for monitoring the application. In the Name field, you fill in the fully qualified class name of the Instrumentation subclass for your application, and for Target Package, you provide the name of the package whose manifest file contains the <application> tag for the application to be monitored.

Using the AndroidManifest.xml



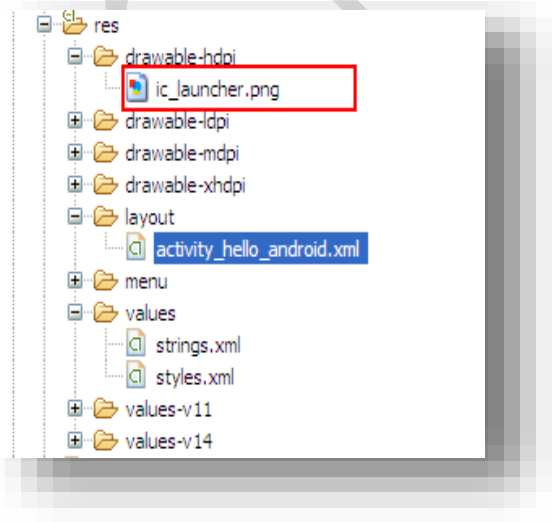
- The build.gradle file in android studio contains the min and max sdk versions.



- **Resources**
  - No matter what your development environment, it's always good practice to keep non-code resources like images and string constants external to your code. Android supports the externalization of resources ranging from simple values such as strings and colors to more complex resources like images, animations, and themes.
  - By externalizing resources, they become easier to maintain, update, and manage.
  - Android will automatically select the correct resource values withoutyourhaving to write a line of code.

Creating Resources

- Application resources are stored under the res/ folder of your project hierarchy.



- There are seven primary resource types that have different folders: simple values, drawables, layouts, animations, XML, styles, and raw resources.

- When your application is built, these resources will be compiled as efficiently as possible and included in your application package.

- R class file that contains references to each of the resources you include in your project.
- In all cases, the resource filenames should contain only lowercase letters, numbers, and the period (.) and underscore (_) symbols.

Creating Simple Values

- Supported simple values include strings, colors, dimensions, and string or integer arrays. All simple values are stored within XML files in the res/values folder.

```
<resources>
<string name="app_name">ToDoList</string>
<string name="hello_world">Hello world!</string>
<string name="menu_settings">Settings</string>
<string name="title_activity_to_do_list">ToDoList</string>
<string name="MyTitle">My Activity</string>
<dimen name="form_width">200px</dimen>
<color name="BackColor">#FFFFFF</color>
</resources>
```

Strings :

- <string name=”app_name”>Hello Android</string>
- Android supports simple text styling, so you can use the HTML tags

```
<string name="app_name">ToDoList</string>
<string name="hello_world">Hello world!</string>
<string name="menu_settings">Settings</string>
<string name="title_activity_to_do_list">ToDoList</string>
```

Colors

- Use the color tag to define a new color resource. Specify the color value using a # symbol followed by the (optional) alpha channel, then the red, green, and blue values using one or two hexadecimal numbers with any of the following notations:
- #RGB
- #RRGGBB
- #ARGB
- #ARRGGBB

```
<color name="BackColor">#FFFFFF</color>
<color name="ForeColor">#000000</color>
```

Dimensions

- Dimensions are most commonly referenced within style and layout resources. They’re useful for creating layout constants such as borders and font heights.
- To specify a dimension resource, use the dimentag, specifying the dimension value, followed by an identifier describing the scale of your dimension:
  - px Screen pixels
  - in Physical inches
  - pt Physical points
  - mm Physical millimeters
  - dp Density-independent pixels relative to a 160-dpi screen
  - sp Scale-independent pixels

```
<dimen name="form_width">200px</dimen>
<dimen name="form_Height">150dip</dimen>
```

Styles and Themes

Style resources let your applications maintain a consistent look and feel by specifying the attribute values used by Views. The most common use of themes and styles is to store the colors and fonts for an application.

To create a style, use a style tag that includes a name attribute, and contains one or more item tags. Each item tag should include a name attribute used to specify the attribute (such as font size or color) being defined.

```
<style name="CodeFont" parent="@android:style/TextAppearance.Medium">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textColor">#3399CC</item>
    <item name="android:textStyle">bold</item>
</style>
```

- **Apply a theme to an Activity or application**

- First create style into string.xml file

```
<resources>
    <string name="hello"><b><i>Hello World, helloAndroid!</i></b></string>
    <string name="app_name"><b>helloAndroid</b></string>
    <color name="custom_theme_color">#b0b0ff</color>
    <style name="CustomTheme" parent="android:Theme.Light">
        <item name="android:windowBackground">@color/custom_theme_color</item>
        <item name="android:colorBackground">@color/custom_theme_color</item>
    </style>
</resources>
```

- To set a theme for all the activities of your application, open the AndroidManifest.xml file and edit the <application> tag to include the android:theme attribute with the style name.

```
<application
              android:icon="@drawable/icon"
              android:label="@string/app_name"
              android:theme="@style/CustomTheme"
>
    <activity android:name=".helloAndroid"
              android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
 </activity>
```

- **Apply Style to our example toDoList**
    - First add style to string.xml file

```
<resources>
    <string name="hello">Hello World, helloAndroid!</string>
    <string name="app_name">toDoList</string>

    <color name="backcolor">#3399CC</color>
    <color name="listbackcolor">#FFF</color>

    <style name="ToDoTheme" parent="@android:style/Theme.Black">
        <item name="android:textSize">20sp</item>
        <item name="android:background">@color/backcolor</item>
    </style>

</resources>
```

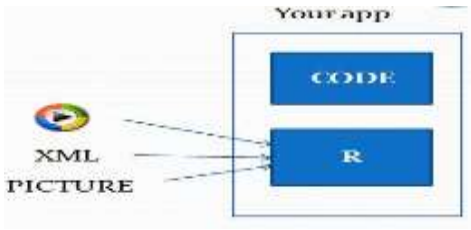
- **Apply the theme to your project in the manifest:**

```
<application    android:icon="@drawable/icon"
                android:label="@string/app_name"
                android:theme="@style/ToDoTheme">
        <activity
                android:name=".helloAndroid"
                android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        </activity>
</application>
```

- The output will be.

R.java

- The directory "gen" in an Android project contains generated values.
- "R.java" is a generated class which contains references to resources of the "res" folder in the project.



- Text, Picture, Sound etc. Everything is broken out of the code into a resource which is reference in class called R.
- These resources can for example be text or icons. If you create new resources the corresponding reference is automatically created in R.java. The references are static int values, the Android system provides methods to access the corresponding resource.

Note: Please do not try to modify "R.java" manually.

```
package com.helloAndroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int Button01=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

BUILDING BLOCK OF ANDROID

There are six components that provide the building blocks for your applications

View:

The concept of a view in J2EE and Swing carries over to Android. Views are UI elements that form the basic building blocks of a user interface. Views are hierarchical and they know how to draw themselves.

Activities:

Your application's presentation layer. An activity is a user interface concept. An activity usually represents a single screen in your application. It generally contains one or more views, but it doesn't have to.

Services:

Services in Android resemble services you see in Windows or other platforms—they're background processes that can potentially run for a long time.

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.
- **Remote services** are services that are meant to be accessed remotely by other applications running on the device.

Content Provider:

Data sharing among mobile applications on a device is common. Therefore, Android defines a standard mechanism for applications to share data (such as a list of contacts, manage and share application databases). Through content providers, you can expose your data and have your applications use data from other applications.

Intent:
Intent generically defines an "intention" to do some work. You can use intents to perform the following tasks, for instance:

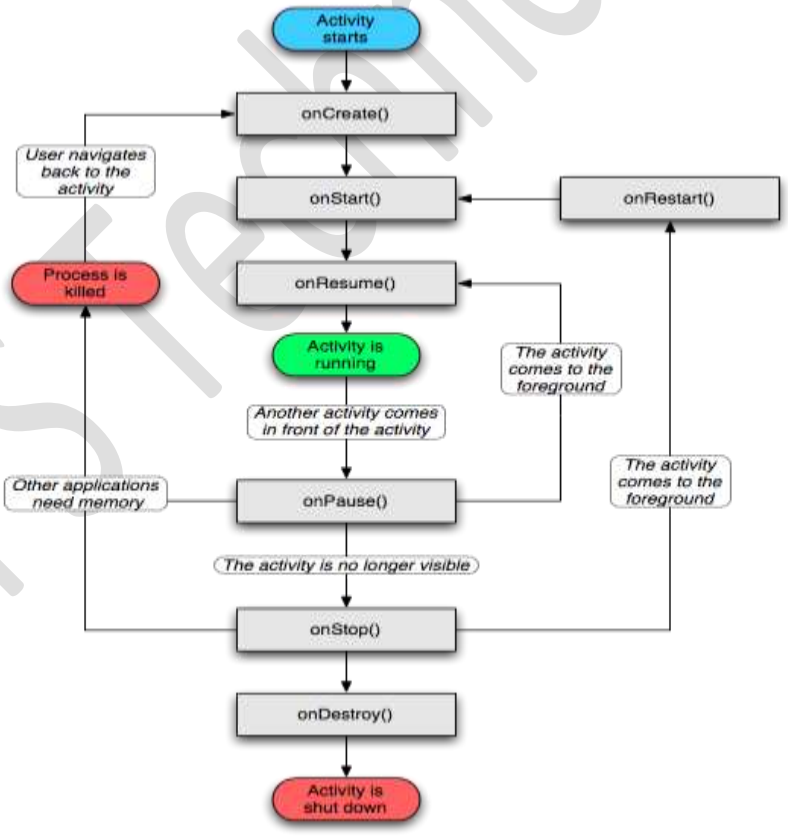
- Broadcast a message
- Start a service
- Launch an activity
- Display a web page or a list of contacts
- Dial a phone number or answer a phone call

Intents are not always initiated by your application—they're also used by the system to notify your application of specific events.

ACTIVITY AND INTENTS

- An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with **setContentView()**
- Activity is a window that contains the user interface of your application. As there are various states of activity like Running, Paused, Stopped and Killed.
- Activity base class contains events that govern the life cycle of an activity.

Activity Lifecycle



Introducing Intents

- *Intents* are used as a message-passing mechanism that lets you declare your intention that an action be performed, usually with (or on) a particular piece of data.
- One of the most common uses for Intents is to start new Activities, either *explicitly* (by specifying the class to load) or *implicitly* (by requesting an action be performed on a piece of data).

- Intents can also be used to broadcast messages across the system. Any application can register a Broadcast Receiver to listen for, and react to, these broadcast Intents. This lets you create event-driven applications based on internal, system, or third-party application events.
- Android uses broadcast Intents to announce system events, like changes in Internet connection status or battery charge levels.

Using Intents to Launch Activities

- The most common use of Intents is to bind your application components. Intents are used to start, stop, and transition between the Activities within an application.
- To open a different application screen (Activity) in your application, call startActivity, passing in an Intent, as shown in the snippet below.

```
startActivity(myIntent);
```

- The Intent can either explicitly specify the class to open, or include an action that the target should perform. In the latter case, the run time will choose the Activity to open, using a process known as "Intent resolution."

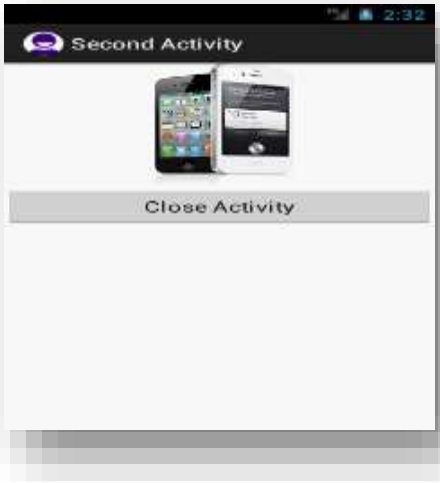
Explicitly Starting New Activities

- To explicitly select an Activity class to start, create a new Intent specifying the current application context and the class of the Activity to launch. Pass this Intent in to startActivity, as shown in the following code snippet:

```
Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);
startActivity(intent);
```

- After calling startActivity, the new Activity (in this example, MyOtherActivity) will be created and become visible and active, moving to the top of the Activity stack.
- Calling finish programmatically on the new Activity will close it and remove it from the stack. Alternatively, users can navigate to the previous Activity using the device's Back button.

Using intent example



- Create new project and make background red. Put Label red and also add one button.
- In your project res/layout/main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ImageView
        android:id="@+id/img1"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:contentDescription="Hello"
        android:src="@drawable/img_a" />
<Button
        android:id="@+id/btn1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/img1"
        android:text="Change Activity" />
</RelativeLayout>
```

In your project src/com.buttonExample/buttonExample.java file

```
public class IntentDemo extends Activity {
      private Button btn;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_intent_demo);
            btn = (Button) findViewById(R.id.btn1);
            btn.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        // Initialize intent for open activity
                        Intent i = new Intent(IntentDemo.this,
SecondIntent.class);
      // Start Activity
                        startActivity(i);
                  }
            });
      }
}
```

- Now create new layout for second activity. In res/layout/new_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<ImageView
        android:id="@+id/img2"
        android:layout_width="fill_parent"
android:layout_height="wrap_content"
        android:contentDescription="Hello"
        android:src="@drawable/img_e" />
<Button
        android:id="@+id/btn2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/img2"
        android:text="Close Activity" />
</RelativeLayout>
```

- Create new Activity class for second layout.
- In src/com.buttonExample/newActivityG.java

```
public class SecondIntent extends Activity {
	private Button btns;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_intent_second);
		btns = (Button) findViewById(R.id.btn2);
		btns.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View arg0) {
				// TODO Auto-generated method stub
				finish();
			}
		});
	}
}
```

- But here we have to add new activity in our Androidmanifest.xml file

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.intentdemo"
    android:versionCode="1"
    android:versionName="1.0" >
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
<activity
            android:name=".IntentDemo"
            android:label="@string/title_activity_intent_demo" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
            android:name=".SecondIntent"
            android:label="Second Activity" >
</activity>
</application>
</manifest>
```

Intent Filters

- Intents often describes the action which should be performed and provide data on which such an action should be done. For example your application can start via intent a browser component for a certain URL. This is demonstrated by the following example.

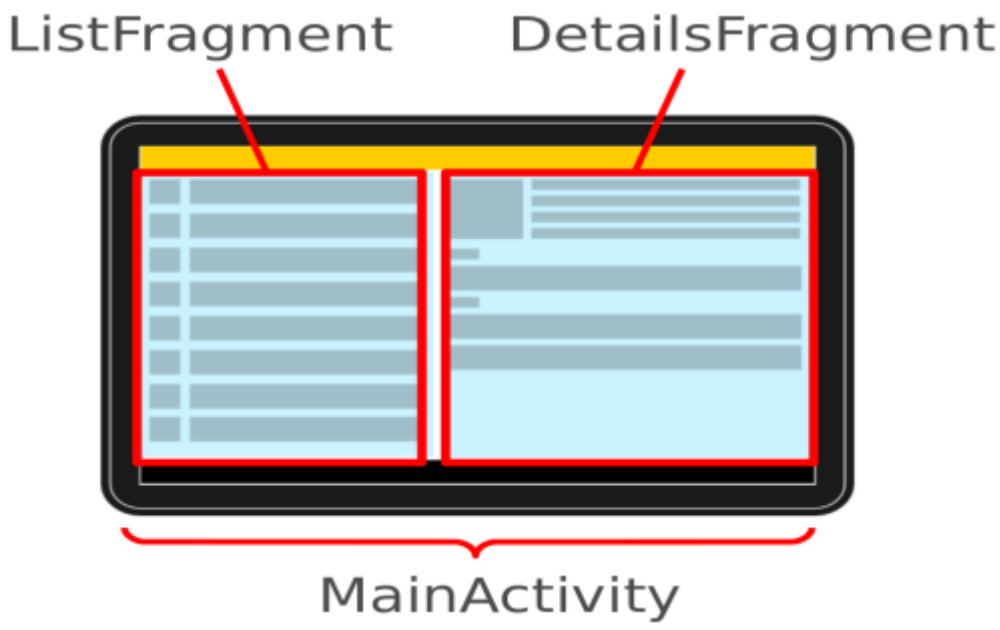
```
String url = "http://www.tops-int.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
startActivity(i);
```

But how does the Android system identify the components which can react to certain intent?
For this the concept of an *intent filter* is used. An intent filter specifies the types of intents to which that an activity, service, or broadcast receiver can respond to. It declares therefore the capabilities of a component.
Android components register intent filters either statically in the *AndroidManifest.xml* or in case of a broadcast receiver also dynamically via code.

Fragment:

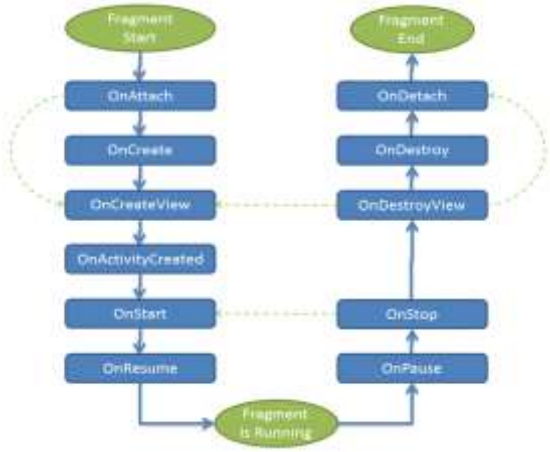
- Fragment is a modular component in an activity which has its own life cycle and event handling, and it is very similar to an activity. Although fragments have their own life cycle, they are directly affected by their owner activity's life cycle. For instance, if anactivity is destroyed, its fragments are also destroyed. Everyfragment should have anowneractivity. A fragment could be added to or removed from an activity dynamically.
- Fragments increase software reusability and provide flexibility in user interface design. A fragment could be used by more than one activity. This way you implement once and use multiple times. Furthermore, it is possible to use a fragment for different layout configurations and different screen modes. This way it provides flexibility in user interface design.

⚠ **Version Alert : This feature was added in Android-3.0 API Level 11**



Fragment Lifecycle

Fragments have their own lifecycle; however, they are still directly affected by their owner activity's lifecycle. The following diagram shows the creation flow of lifecycle of a fragment:



activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity"
   android:orientation="horizontal">
<fragment android:id="@+id/listFrag"
      android:layout_height="fill_parent"
      android:layout_width="200dip"
      class="com.example.fragmentactivity.ListFargActivity">
</fragment>
<fragment android:id="@+id/detailFr"
            android:layout_height="fill_parent"
            android:layout_width="200dip"
            class="com.example.fragmentactivity.DetailFragActivity">
</fragment>
</LinearLayout>
```

MainActivity.java

```
packagecom.example.fragmentactivity;
importandroid.os.Bundle;
importandroid.app.Activity;
importandroid.view.Menu;
publicclassMainActivityextends Activity {
            @Override
            protectedvoid onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.activity_main);
            }
            @Override
            publicboolean onCreateOptionsMenu(Menu menu) {

            getMenuInflater().inflate(R.menu.activity_main, menu);
            returntrue;
            }
```

```
}
```

list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
<ListView
      android:id="@+id/listView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
</ListView>
</LinearLayout>
```

ListFargActivity.java

```
publicView onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        View v = inflater.inflate(R.layout.list, container, false);
        ListView lv = (ListView) v.findViewById(R.id.listView1);
        lv.setAdapter(newArrayAdapter<String>(getActivity(),
                        android.R.layout.simple_list_item_1, items));
        lv.setOnItemClickListener(newOnItemClickListener() {
        @Override
        publicvoid onItemClick(AdapterView<?> arg0, View arg1, int
                                arg2,long arg3) {
                // TODO Auto-generated method stub
                String str = items[arg2];
                DetailFragActivity dfa = (DetailFragActivity) getFragmentManager()
                        .findFragmentById(R.id.detailFr);
                dfa.setText(getCapital(str));
                        }
                });     return v; }

protected String getCapital(String str) {
                // TODO Auto-generated method stub
                if (str.equalsIgnoreCase("gujarat")) {
                        return "Gandhinagar";
                }
                if (str.equalsIgnoreCase("maharashtra")) {
                        return "Mumbai";
```

```
                }
                if (str.equalsIgnoreCase("Orissa")) {
                        return "Bhubaneshwar";
                }
                if (str.equalsIgnoreCase("Rajasthan")) {
                        return "Jaipur";
                }
                return "No capital";
        }
        String[] items = { "Gujarat", "Maharashtra", "Orissa", "Rajasthan" };
```

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
<TextView
      android:id="@+id/tvDetails"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="TextView" />
</LinearLayout>
```

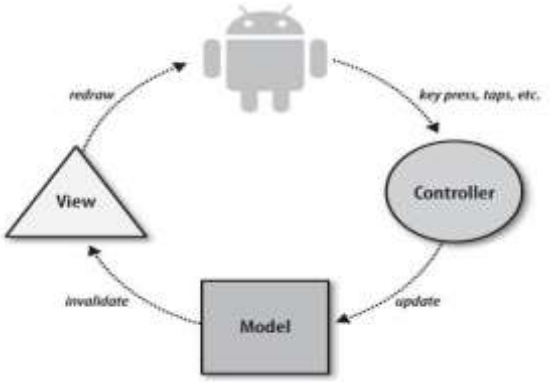
DetailFragActivity.java

```
package com.example.fragmentactivity;
@SuppressLint("NewApi")
public class DetailFragActivity extends Fragment {
        TextView tvRes;
        @TargetApi(Build.VERSION_CODES.HONEYCOMB)
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                View v = inflater.inflate(R.layout.details, container, false);
                return v;
        }
        public void setText(String item) {
                tvRes = (TextView) getView().findViewById(R.id.tvDetails);
                tvRes.setText(item);
        } }
```

ANDROID UI AND COMPONENTS

Android GUI Architecture

- The Android environment adds yet another Graphical User Interface (GUI) toolkit tothe Java ecosphere, joining AWT, Swing, SWT, and J2ME (leaving aside the web UItoolkits). If you've worked with any of these, the Android framework will look familiar.
- The Android UI framework is, like the other UI frameworks, organized around the common Model-View-Controller. It provides structure and tools for building a Controller that handles user input and a View that renders graphical information to the screen.



The Model

- The Model is the guts of your application: what it actually does. It might be, for instance, the database of tunes on your device and the code for playing them. It might be your list of contacts and the code that places phone calls or sends IMs to them.
- While a particular application's View and Controller will necessarily reflect the Model they manipulate, a single Model might be used by several different applications.

The View

- The View is the application's feedback to the user. It is the portion of the application responsible for rendering the display, user feedback, etc. The graphical portion of the Android UI framework's View.
- As an example, the display in a hypothetical MP3 player might contain a component that shows the album cover for the currently playing tune. Another component might display the name of the currently playing song, while a third contains subcomponents such as the Play, Pause, and Stop buttons.

The Controller

- The Controller is the portion of an application that responds to external actions: a keystroke, a screen tap, an incoming call, etc. It is implemented as an event queue. Each external action is represented as a unique event in the queue. The framework removes each event from the queue in order and dispatches it.
- For example, when a user presses a key on his phone, the Android system generates a KeyEvent and adds it to an event queue.

INTRODUCING VIEWS

- User Interface design, human–computer interaction, and usability are huge topics that aren't covered in great depth in this book. Nonetheless, it's important that you get them right when creating your User Interfaces. Android introduces some new terminology for familiar programming metaphors that will be explored in detail in the following sections:

- **Views:***Views* are the basic User Interface class for visual interface elements (commonly known as controls or widgets). All User Interface controls, and the layout classes, are derived from Views.
- **View Groups:***View* Groups are extensions of the View class that can contain multiple child Views. By extending the ViewGroup class, you can create compound controls that are made up of interconnected child Views. The ViewGroup class is also extended to provide the layout managers, such as LinearLayout, that help you compose User Interfaces.
- **Activities:**Activities, described in detail in the previous chapter, represent the window or screen being displayed to the user. Activities are the Android equivalent of a Form. To display a User Interface, you assign a View or layout to an Activity. Android provides several common UI controls, widgets, and layout managers.

- **Creating Activity User Interfaces with Views**
  - A new Activity starts with a temptingly empty screen onto which you place your User Interface. To set the User Interface, call setContentView, passing in the View instance (typically a layout) to display.
  - The setContentView method accepts either a layout resource ID or a single View instance.
  - The following code snippet shows how to set the User Interface for an Activity using an external layout resource. This example assumes that main.xml exists in the project's **res/layout** folder.
  - You can get references to the Views used within a layout with the findViewById method.

```
public class ToDoList extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_to_do_list);
	}
}
```

Layout

- Layouts are Android's solution to the variety of screens that come on Android devices: they can have different pixel densities, different dimensions, and different aspect ratios. Typical Android devices allow changing the screen orientation while applications are running, so the layout infrastructure needs to be able to respond on the fly.
- Layouts are intended to give developers a way to express the physical relationship of Views as they are drawn on the screen.
- Layouts in Android are in the form of a tree, with a single root and a hierarchy of Views. Each View in the tree is termed the *parent* of the Views it contains and the child of the View that contains it. Layout is a two-pass process:

Measure pass

Traversing the tree from the root, each View in the layout records its dimensionalrequest—in other words, how much vertical height and horizontal width it needsto display itself in the final display.

Layout pass

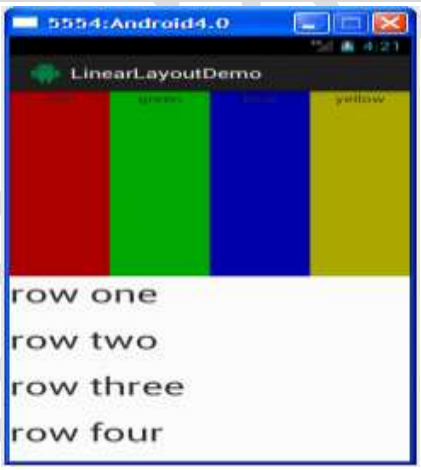
Again traversing the tree from the root, each parent View uses the available layoutinformation to position its children as requested. If the requests can't be followedexplicitly, Android does its best to make everything fit on the screen. If there areno requests given, it uses a default set of layout parameters. Each parent can passlayout information on to its children, telling them where they are positioned andwhat screen dimensions they have been granted (they might get less than theyrequested).A Layout is a View itself, so there's nothing wrong with having multiple Layouts in asingle layout XML file—they just have to be arranged in a hierarchy. So it's perfectlyvalid to have a vertical LinearLayout that includes a TableLayout as one of its rows.

- **LinearLayout** Organizes its children either horizontally or vertically.
- **TableLayout** Organizes its children in tabular form.
- **RelativeLayout** Organizes its children relative to one another or to the parent.
- **Grid Layout** A layout that places its children in a rectangular *grid.*
- **FrameLayout** Allows you to dynamically change the control(s) in the layout.

Linear Layout

- LinearLayout is used when we need to arrange the widgets/views in a horizontal or vertical manner. The direction of arrangement can be set to horizontal or vertical; by default it is being horizontal.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:orientation="horizontal" >
<TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="#aa0000"
            android:gravity="center_horizontal"
            android:text="red" />
<TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="#00aa00"
            android:gravity="center_horizontal"
            android:text="green" />
<TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="#0000aa"
            android:gravity="center_horizontal"
            android:text="blue" />
<TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="#aaaa00"
            android:gravity="center_horizontal"
            android:text="yellow" />
</LinearLayout>
<LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:orientation="vertical" >
<TextView
```



```
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="row one"
            android:textSize="15pt" />
<TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="row two"
            android:textSize="15pt" />
      <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="row three"
      <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="row four"
            android:textSize="15pt" />
</LinearLayout>
</LinearLayout>
```

Relative Layout

- Using the Relative Layout, you can define the positions of each of the child Views relative to each other and the screen boundaries.
- In res/layout/main.xml file.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<TextView
        android:id="@+id/txt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
<EditText
        android:id="@+id/Etxt1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txt1"
        android:ems="10"
        android:inputType="textNoSuggestions" >
</EditText>
<Button
        android:id="@+id/btnOK"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/Etxt1"
        android:text="OK" />
<Button
```

```
        android:id="@+id/btnC"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/Etxt1"
        android:layout_toLeftOf="@+id/btnOK"
        android:text="Cancel" />
</RelativeLayout>
```

Table Layout

- If the Layout's widgets/views need to be arranged in the form of rows and columns, we use this layout object. This is similar to html tables. The cells can span columns. The TableLayout do not display its border.

In res/layout/main.xml file.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1" >
<TableRow>
<TextView
            android:layout_height="wrap_content"
            android:padding="3dip"
            android:text="New" />
<TextView
            android:layout_height="wrap_content"
            android:gravity="right"
            android:padding="3dip"
            android:text="Ctrl+N" />
</TableRow>
<TableRow>
<TextView
            android:layout_height="wrap_content"
            android:padding="3dip"
            android:text="Open" />
<TextView
            android:layout_height="wrap_content"
            android:gravity="right"
            android:padding="3dip"
            android:text="Ctrl+O" />
</TableRow>
<TableRow>
<TextView
            android:layout_height="wrap_content"
            android:padding="3dip"
            android:text="Save" />
<TextView
            android:layout_height="wrap_content"
            android:gravity="right"
            android:padding="3dip"
            android:text="Ctrl+S" />
</TableRow>
<View
        android:layout_height="2dip"
        android:background="#FF909090" />
<TableRow>
```

```
<TextView
            android:layout_height="wrap_content"
            android:padding="3dip"
            android:text="Exit" />
<TextView
            android:layout_height="wrap_content"
            android:gravity="right"
            android:padding="3dip"
            android:text="Ctrl+X" />
</TableRow>
</TableLayout>
```

Frame Layout

- The simplest of the Layout Managers, the *Frame Layout* simply pins each child view to the top left corner. Adding multiple children stacks each new child on top of the previous, with each new View obscuring the last.

In *activity_main.xml*

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:contentDescription="Hello"
        android:scaleType="fitCenter"
        android:src="@drawable/android_logo" />
<ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:contentDescription="Hello"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_launcher" />
<TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:text="@string/LayoutText" />
</FrameLayout>
```

Grid Layout

- Grid Layout is a layout that divides its view space into rows, columns, and cells. Grid Layout places views in it automatically, but it is also possible to define the column and row index to place a view into Grid Layout. With the span property of cells, it is possible to make a view span multiple rows or columns.

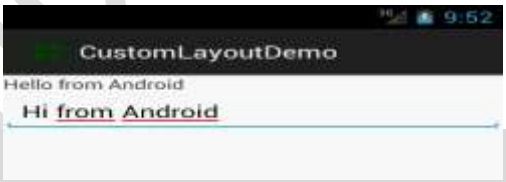
In *activity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/GridLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2"  //No of Columns
    android:orientation="horizontal"
```

```
    android:rowCount="2" >//No of Rows
<TextView
        android:id="@+id/textView1"
        android:text="Cell 1,1"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/textView2"
        android:text="Cell 1,2"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/textView3"
        android:text="Cell 2,1"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/textView4"
        android:text="Cell 2,2"
        android:textAppearance="?android:attr/textAppearanceLarge" />
</GridLayout>
```

Create custom layout in android

- Implementing layouts in XML decouples the presentation layer from View and Activity code. It also lets you create hardware-specific variations that are dynamically loaded without requiring code changes.
- Example creates a new customLayout application using native Android code.
- For this you have to add code in only in src folder



In CustomLayoutDemo.java

```
publicclass CustomLayoutDemo extends Activity {
      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            LinearLayout ll = new LinearLayout(this); \\ Create Linear Layout
            ll.setOrientation(LinearLayout.VERTICAL);
            TextView txt = new TextView(this);
            EditText edt = new EditText(this);
            txt.setText("Hello from Android");
            edt.setText("Hi from Android");
            int Height = LinearLayout.LayoutParams.WRAP_CONTENT;
            int Width = LinearLayout.LayoutParams.MATCH_PARENT;
            ll.addView(txt, new LinearLayout.LayoutParams(Width, Height));
            ll.addView(edt, new LinearLayout.LayoutParams(Width, Height));
            setContentView(ll);
      }
}
```

Android UI Widgets and Events

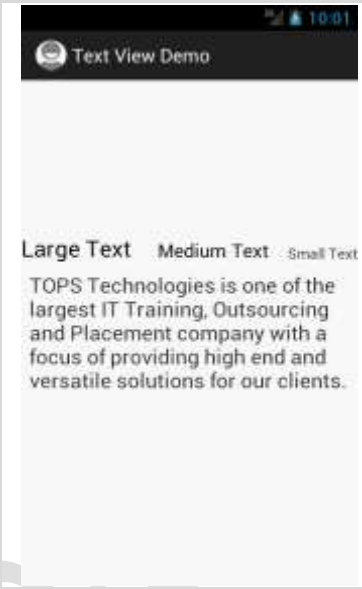
Android supplies a toolbox of standard Views to help you create simple interfaces. By using these controls (and modifying or extending them as necessary), you can simplify your development and provide consistency between applications.
The following list highlights some of the more familiar toolbox controls:

Text View example

In *activity_main.xml*

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
 android:layout_height="match_parent" >
<TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/TOPS"
        android:textSize="20dip"
        tools:context=".TextView" />
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView2"
        android:layout_alignParentRight="true"
        android:text="Small Text"
        android:textAppearance="?android:attr/textAppearanceSmall" />
<TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView2"
        android:layout_marginRight="18dp"
        android:layout_toLeftOf="@+id/textView4"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```



EditText Example

In *activity_main.xml*

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
android:text="Simple Text" />
<EditText
```

```
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/textView1"
    android:inputType="textNoSuggestions">
</EditText>
<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText1"
    android:text="Person Name" />
<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView2"
    android:inputType="textPersonName">
</EditText>
<TextView
    android:id="@+id/textView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText2"
    android:text="Password" />
<EditText
    android:id="@+id/editText3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView3"
    android:inputType="textPassword" />
<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText3"
    android:text="Number Password" />
<EditText
    android:id="@+id/editText4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView4"
```

```
    android:inputType="numberPassword" />
<TextView
    android:id="@+id/textView5"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText4"
    android:text="Email" />
<EditText
    android:id="@+id/editText5"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView5"
    android:inputType="textEmailAddress" />
<TextView
    android:id="@+id/textView6"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText5"
    android:text="Phone" />
<EditText
    android:id="@+id/editText6"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView6"
    android:inputType="phone" />
<TextView
    android:id="@+id/textView7"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText6"
    android:text="Number" />
<EditText
    android:id="@+id/editText7"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView7"
    android:inputType="number">
</EditText>
</RelativeLayout>
```

Button and Image Button

- The Button View is just a button, printed with some text to identify it, that the user can click to invoke some action. The previous section created a Button and connected it to an OnClickListener method that executes when the Button is clicked.

- Android has a very visual, mobile-oriented user interface, so you might want to use abutton with an image on it rather than one with text. Android provides the ImageButtonView for just that purpose.

Start a new project named *buttonExample*. Open the res/layout/main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<Button
        android:id="@+id/Button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Click Me" />
<ImageButton
        android:id="@+id/ImageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/Button1"
        android:contentDescription="ClickMe"
        android:background="@null"
        android:src="@drawable/btn"/>
<TextView
        android:id="@+id/TextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:layout_below="@id/ImageButton1" />
</RelativeLayout>
```

Open buttonExample.java and insert the following code

```
public class ButtonDemoActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_buttondemo);
    ButtonDemoFragment fragment = new ButtonDemoFragment ();
    FragmentTransaction ft =getFragmentManager().beginTransaction();
    ft.add(android.R.id.content, fragment);
    ft.show(fragment);
  }
}
```

Open buttonExample.java and insert the following code

```
public class ButtonDemoFragment extends Fragment {
  private Button btn1;
  private ImageButton imgBtn1;
  private TextView txt1;
  public ButtonDemoFragment() {
    // Required empty public constructor
  }
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
```

```
      // Inflate the layout for this fragment
      View view =inflater.inflate(R.layout.fragment_button_demo, container, false);
      btn1 = (Button)view.findViewById(R.id.Button1);
      imgBtn1 = (ImageButton)view.findViewById(R.id.ImageButton1);
      txt1 = (TextView) view.findViewById(R.id.TextView1);
btn1.setOnClickListener(new View.OnClickListener() {
         @Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
txt1.setText("Clicked Button");
         }
      });
imgBtn1.setOnClickListener(new View.OnClickListener() {
         @Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
txt1.setText("Clicked Image Button");
         }
      });
return view;
   }
}
```

Check Box and Radio Button

The Views we present in this section are probably familiar to you from other user interfaces. Their purpose is to allow the user to choose from multiple options. Check-Boxes are typically used when you want to offer multiple selections with a yes/no ortrue/false choice for each. Radio Buttons are used when only one choice is allowed at atime.

Create new project edit in res/layout/main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >
<CheckBox
      android:id="@+id/CheckBox1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Check Me" />
<TextView
      android:id="@+id/TextView1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/CheckBox1"/>
</RelativeLayout>
```

In checkBox.java file

```
public class CheckBoxDemo extends AppCompatActivity {
          @Override
protected void onCreate(Bundle savedInstanceState) {
```

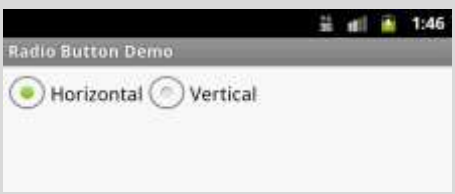
```
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_checkboxdemo);
        CheckboxDemoFragment fragment = new CheckboxDemoFragment ();
        FragmentTransaction ft =getFragmentManager().beginTransaction();
        ft.add(android.R.id.content, fragment);
        ft.show(fragment);

    }
}
```

- **In checkBox.java file**

```
public class CheckboxDemoFragment extends Fragment {
    private CheckBox chk1;
    private TextView txt1;
    public ButtonDemoFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,Bundle savedInstanceState) {

        // Inflate the layout for this fragment
        View view =inflater.inflate(R.layout.fragment_button_demo, container,
false);
        chk1 = (CheckBox) view.findViewById(R.id.CheckBox1);
        txt1 = (TextView) view.findViewById(R.id.txt1);
chk1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean
IsChecked) {
                // TODO Auto-generated method stub
                if (IsChecked) {
                    txt1.setText("Checkbox: Checked");
                } else {
                    txt1.setText("Checkbox: UnChecked");
    }
            }
        });

        return view;
    }
}
```

Create new project and set layout (in main.xml)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<RadioGroup
        android:id="@+id/rg1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
<RadioButton
            android:id="@+id/rb1"
            android:text="Horizontal" />
```



```
<RadioButton
            android:id="@+id/rb2"
            android:text="Vertical" />
</RadioGroup>
</RelativeLayout>
```

In radioButton.Java file

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Fragment fragment = new Fragment();
        FragmentTransaction ft =getFragmentManager().beginTransaction();
        ft.add(android.R.id.content, fragment);
        ft.show(fragment);

    }
}
```

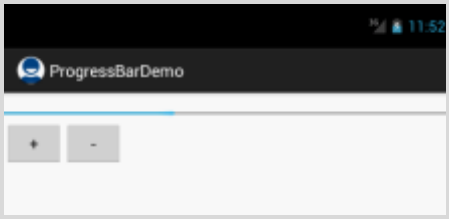
In radioButton.Java file

```
public class RadioBtnFragment extends Fragment {
private CheckBox chk1;
private TextView txt1;
private RadioGroup rg;
public RadioBtnFragment () {
// Required empty public constructor
}
    @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
// Inflate the layout for this fragment
View view =inflater.inflate(R.layout.fragment_button_demo, container, false);

rg = (RadioGroup) view.findViewById(R.id.rg1);
rg.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup arg0, int arg1)
{
if(arg0 == rg)
{
if(arg1 == R.id.rb1)
 {
rg.setOrientation(LinearLayout.HORIZONTAL);
}
               else{
                        rg.setOrientation(LinearLayout.VERTICAL);
}
                }
        }
        });
      return view;
    }
}
```

Progress Bar

- In your project structure in res/layout. In main .xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ProgressBar
        android:id="@+id/pb1"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="15dip" />
<Button
        android:id="@+id/btnPlus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/pb1"
        android:text="+" />
<Button
        android:id="@+id/btnMinus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/btnPlus"
        android:layout_toRightOf="@id/btnPlus"
        android:text="-" />
</RelativeLayout>
```



In . java file

```
public class ProgressBarDemo extends Activity implements OnClickListener {
      private Button plus,minus;
      private ProgressBar pb;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_progress_bar_demo);
        plus=(Button)findViewById(R.id.btnPlus);
        minus=(Button)findViewById(R.id.btnMinus);
        pb=(ProgressBar)findViewById(R.id.pb1);
        plus.setOnClickListener(this);
        minus.setOnClickListener(this);
    }
    public void onClick(View v) {
            // TODO Auto-generated method stub
            if(v==plus)
            {
                  pb.incrementProgressBy(1);
                  setProgress(100*pb.getProgress()); //Set Progress Increse
            }
            else
            {
                  pb.incrementProgressBy(-1);
                  setProgress(100*pb.getProgress()); //Set Progress Decrese
```

```
            }
      }
}
```

Rating Bar

Main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<RatingBar
        android:id="@+id/rt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="146dp" />
<TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/rt1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="51dp"
        android:text="Rating" />
</RelativeLayout>
```

. java file

```
public class RatingBarActivity extends AppCompatActivity {

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

RatingDemoFragment fragment = new RatingDemoFragment ();
        FragmentTransaction ft =getFragmentManager().beginTransaction();
        ft.add(android.R.id.content, fragment);
        ft.show(fragment);

    }
}
```

. java file

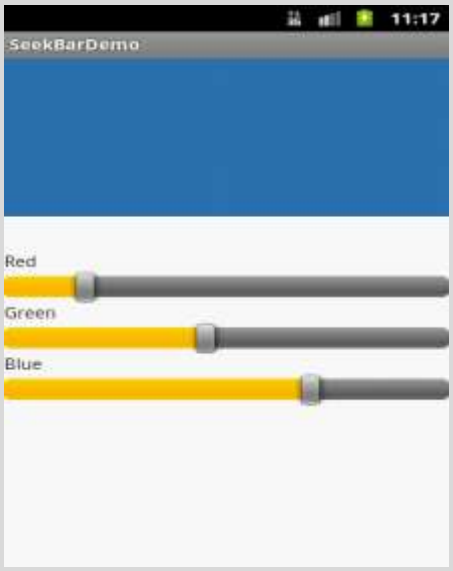
```
public class RatingDemoFragment extends Fragment {
private RatingBar rb;
private TextView tt;
public RatingDemoFragment() {
// Required empty public constructor
}
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view =inflater.inflate(R.layout.fragment_button_demo, container,
```

```
false);
rb = (RatingBar) view.findViewById(R.id.rt1);
tt = (TextView) view.findViewById(R.id.tv1);

rb.setOnRatingBarChangeListener(newRatingBar.OnRatingBarChangeListener()
{
            @Override
public void onRatingChanged(RatingBar ratingBar, float rating, boolean
fromUser) {
tt.setText("Rating: " + rating + "/" + rb.getNumStars());
            }
        });
return view;
    }}
```

SeekBar

Main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".SeekBarDemo" >

<TextView
android:id="@+id/textViewGreen"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_centerVertical="true"
android:text="Green" />
<SeekBar
android:id="@+id/seekBarGreen"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/textViewGreen"
android:max="255" />

<TextView
android:id="@+id/textViewBlue"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/seekBarGreen"
android:text="Blue" />

<SeekBar
android:id="@+id/seekBarBlue"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/textViewBlue"
android:max="255" />

<SeekBar
```



```
android:id="@+id/seekBarRed"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_above="@+id/textViewGreen"
android:layout_alignParentLeft="true"
android:max="255" />

<TextView
android:id="@+id/textViewRed"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/seekBarRed"
android:layout_alignParentLeft="true"
android:text="Red" />

<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="150dip"
android:layout_alignParentLeft="true"
android:layout_alignParentRight="true"
android:layout_alignParentTop="true"
android:text=""
android:textAppearance="?android:attr/textAppearanceLarge" />
</RelativeLayout>
```

. java file

```
public class SeekBarDemo extends Activity implements OnSeekBarChangeListener {
	TextView txt;
	SeekBar red, green, blue;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_seek_bar_demo);
		txt = (TextView) findViewById(R.id.textView1);
		red = (SeekBar) findViewById(R.id.seekBarRed);
		green = (SeekBar) findViewById(R.id.seekBarGreen);
		blue = (SeekBar) findViewById(R.id.seekBarBlue);
		red.setOnSeekBarChangeListener(this);
		green.setOnSeekBarChangeListener(this);
		blue.setOnSeekBarChangeListener(this);
	}
	@Override
	public void onProgressChanged(SeekBar seekBar, int progress,
			boolean fromUser) {
		// Call Set color on change of value
		if (seekBar == red) {
			SetColor(); // Call color setting method
		} else if (seekBar == green) {
			SetColor();
		} else {
			SetColor();
		}	}
	@Override
	public void onStartTrackingTouch(SeekBar seekBar) {
```

```
            // TODO Auto-generated method stub
      }
      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {
            // TODO Auto-generated method stub
      }
      private void SetColor() {
            int r = 0, g = 0, b = 0; // Initialize color code
            r = red.getProgress(); //Set Red Color
            g = green.getProgress(); //Set Green Color
            b = blue.getProgress(); //Ser Red Color

            txt.setBackgroundColor(Color.rgb(r, g, b));
      }}
```

WEBVIEW

Other GUI toolkits let you use HTML for presenting information, from limited HTML renderers (e.g., Java/Swing) to embedding Internet Explorer into .NET applications. Android is much the same, in that you can embed the built-in Web browser as a widget in your own activities, for displaying HTML or full-fledged browsing. The Android browser is based on WebKit, the same engine that powers Apple's Safari Web browser.

The Android browser is sufficiently complex that it gets its own Java package (android.webkit), though using the WebView widget itself can be simple or powerful, based upon your requirements.

For simple stuff, WebView is not significantly different than any other widget in Android—pop it into a layout, tell it what URL to navigate to via Java code and you're done.

Create new project in that main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<WebView
        android:id="@+id/web1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</RelativeLayout>
```

In .java file

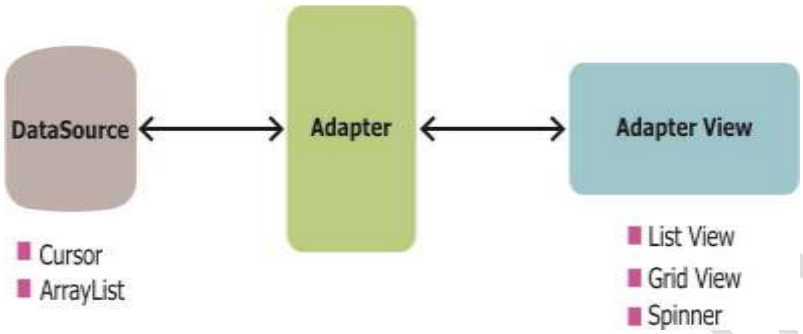
```
public class WebView extends Activity {
      private android.webkit.WebView web;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_web_view);
            web = (android.webkit.WebView) findViewById(R.id.web1);
            web.loadUrl("http://www.google.com"); //Load url in webview
      }
}
```

Because this application needs access to the Internet, you need to add the appropriate permissions to the Android manifest file. Open the AndroidManifest.xml file and add the following as a child of the <manifest> element:

```
//Add permission in manifest.
<uses-permission android:name="android.permission.INTERNET"/>
```

Introduction to Adapters

- An *adapters* manages the data model and adapts it to the individual rows in the list view. An adapter extend the BaseAdapter class.
- Adapters call the getView() method which returns a view for each item within the adapter view. The layout format and the corresponding data for an item within the adapter view is set in the getView() method.



ListView

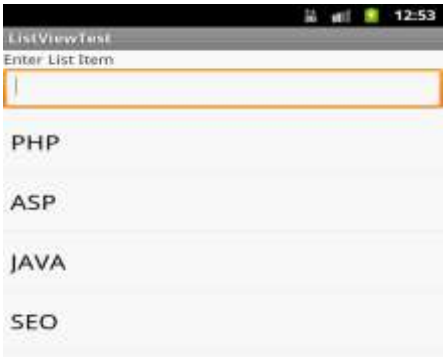
- List View is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.
- For example when we need to show items in a vertical scrolling list. One interesting aspect is this component can be deeply customized and can be adapted to our needs.

Add Listview control to your layout .xml file

In .java file

```
public class ListViewTest extends Activity {
	private EditText txt1;
	private ListView lst1;
	final ArrayList<String> ListItems = new ArrayList<String>();
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_list_view_test);
		txt1 = (EditText) findViewById(R.id.editText1);
		lst1 = (ListView) findViewById(R.id.listView1);//Find Listview
		lst1.setAdapter(new ArrayAdapter<String>(this,
				android.R.layout.simple_list_item_1, ListItems));
		txt1.setOnKeyListener(new OnKeyListener() { //On Enter press
			@Override
			public boolean onKey(View v, int keyCode, KeyEvent event) {
				// TODO Auto-generated method stub
				if (event.getAction() == KeyEvent.ACTION_DOWN) {
					if (txt1.getText().toString() != "") {
					String str = txt1.getText().toString();
						ListItems.add(str);
						txt1.setText("");
						return true;
					}
				}
				return false;
			}
		});
	}
}
```

The output will be…



Add string into EditText and press down button otherwise "D-pad center button"

Spinners

Spinners are similar to combo boxes in some frameworks. A combo box typically displaysthe currently selected option, along with a pull-down list from which the user canclick on another option to select it.

In your project res/layout/main.xml file

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<Spinner
        android:id="@+id/spn1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
<TextView
        android:id="@+id/TextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/spn1" />
</RelativeLayout>
```

In spinDemo.java file

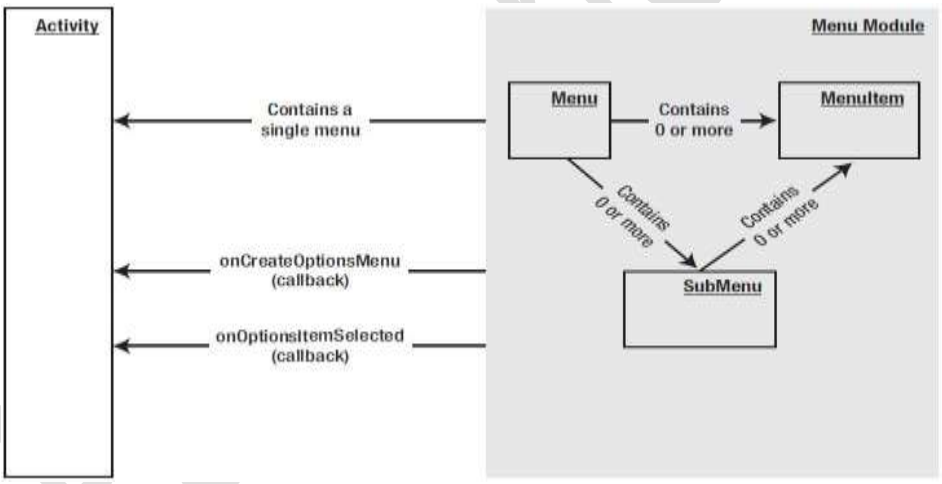
```
public class SpinnerDemo extends Activity {
      String[] Items = { "Option1", "Option2", "Option3", "Option4", "Option5"
};
      private Spinner spn;
      private TextView txt1;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_spinner_demo);
            spn = (Spinner) findViewById(R.id.spn1);
            txt1 = (TextView) findViewById(R.id.TextView1);
      // Create Adapter
            ArrayAdapter<String> a = new ArrayAdapter<String>(this,
                        android.R.layout.simple_spinner_item, Items);
            spn.setAdapter(a); //Set Adapter
            spn.setOnItemSelectedListener(new OnItemSelectedListener() {
                  @Override
                  public void onItemSelected(AdapterView<?> arg0, View arg1,
```

```
                              int arg2, long arg3) {
                        // TODO Auto-generated method stub
                        txt1.setText(Items[arg2]);
                  }

                  @Override
                  public void onNothingSelected(AdapterView<?> arg0) {
                        // TODO Auto-generated method stub
                  }
            });
      }
}
```

Android Menus

Whether you've worked with Swing in Java, with WPF (Windows Presentation Foundation) in Windows, or with any other UI framework, you've no doubt worked with menus. In addition to providing comprehensive support for menus, Android presents some new patterns such as XML menus and alternative menus.

you will learn how to create menus and menu items, and how to respond to menu items. The key class in Android menu support is android.view.Menu. Every activity in Android is associated with a menu object of this type, which can contain a number of menu items and submenus. Menu items are represented by android.view.MenuItem and submenus are represented by android.view.SubMenu.



You can group menu items together by assigning each one a group ID, which is merely an attribute. Multiple menu items that carry the same group ID are considered part of the same group.

Activity Menu

To define a menu for an Activity, override its onCreateOptionsMenu method. This method is triggered the first time an Activity's menu is displayed.

The onCreateOptionsMenu receives a Menu object as a parameter. You can store a reference to, and continue to use, the Menu reference elsewhere in your code until the next time that onCreateOptionsMenu is called.

Use the add method on the Menu object to populate your menu. For each Menu Item, you must specify:

- A group value to separate Menu Items for batch processing and ordering
- A unique identifier for each Menu Item. For efficiency reasons, Menu Item selections are generally handled by the onOptionsItemSelected event handler, so this unique identifier is important to

determine which Menu Item was pressed. It is convention to declare each menu ID as a private static variable within the Activity class. You can use the Menu. FIRST static constant and simply increment that value for each subsequent item.

- An order value that defines the order in which the Menu Items are displayed
- The menu text, either as a character string or as a string resource.

Simple Icon Menu example
In your project in .java file

```
public class MenuDemo extends Activity {

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

	setContentView(R.layout.activity_menu_demo);
	}
	@Override
	public boolean onCreateOptionsMenu(Menu menu)
{
		super.onCreateOptionsMenu(menu);
		int GroupId = 0; // Group Id
		int ItemId = Menu.FIRST; // Unique Menu
Item Identifier
		int OrderId = Menu.NONE; // Order Position
		MenuItem m1 = menu.add(GroupId, ItemId, OrderId, "First Menu");
		MenuItem m2 = menu.add(GroupId, ItemId + 1, OrderId + 1, "Second
Menu");
		MenuItem m3 = menu.add(GroupId, ItemId + 1, OrderId + 1, "Third
Menu");
		MenuItem m4 = menu.add(GroupId, ItemId + 1, OrderId + 1, "Fourth
Menu");
		return true;
	}
}
```

Creating Submenus

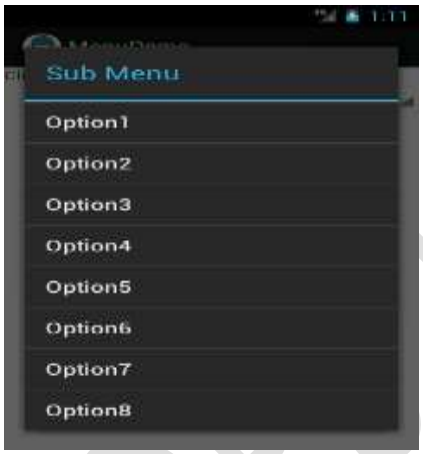
Submenus are displayed as regular Menu Items that, when selected, reveal more items. Traditionally, submenus are displayed using a hierarchical tree layout. Android uses a different approach to simplify menu navigation for small-screen devices. Rather than a tree structure, selecting a submenu presents a single floating window that displays all of its Menu Items.

You can add submenus using the addSubMenu method. It supports the same parameters as the add method used to add normal Menu Items, allowing you to specify a group, unique identifier, and textstring for each submenu. You can also use the setHeaderIcon and setIcon methods to specify anicon to display in the submenu's header bar or the regular icon menu, respectively.

Example :

```
public class MenuDemo extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_menu_demo);
	}
	@Override
```

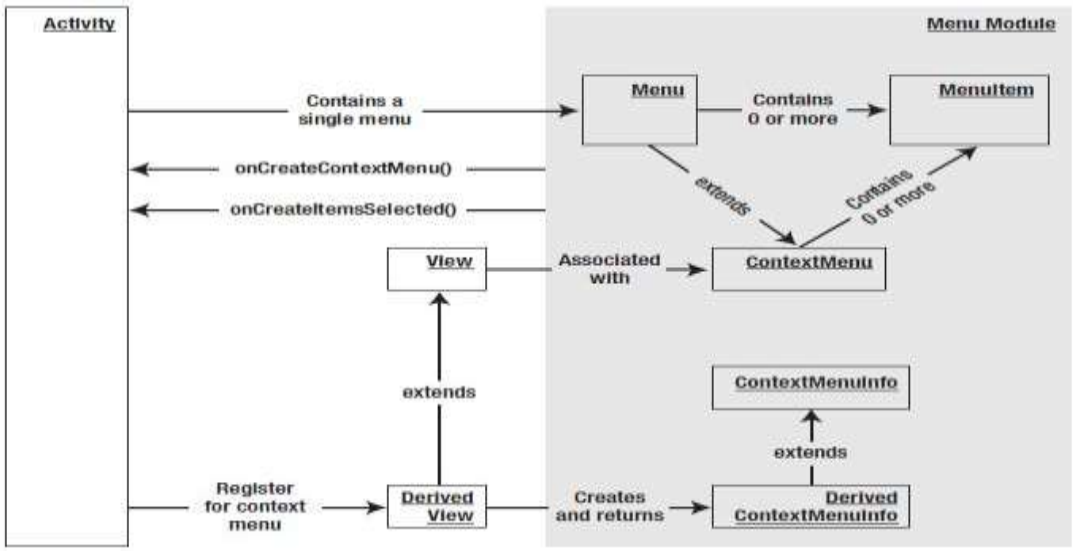
```
	public boolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		int GroupId = 0; //Group Id
		int ItemId = Menu.FIRST; //Unique Menu Item Identifier
		int OrderId = Menu.NONE; //Order Position
	// Add Menu Item in sub menu
	MenuItem m1 = menu.add(GroupId, ItemId, OrderId, "First Menu");
	SubMenu sub = menu.addSubMenu(GroupId, ItemId, OrderId, "SubMenu");
		sub.setHeaderTitle("Sub Menu");
		sub.setIcon(R.drawable.more);
	// Add Menu Item in sub menu
		MenuItem sm1 = sub.add(0, 0, Menu.NONE, "Option1");
		MenuItem sm2 = sub.add(0, 0, Menu.NONE, "Option2");
		MenuItem sm3 = sub.add(0, 0, Menu.NONE, "Option3");
		MenuItem sm4 = sub.add(0, 0, Menu.NONE, "Option4");
		MenuItem sm5 = sub.add(0, 0, Menu.NONE, "Option5");
		MenuItem sm6 = sub.add(0, 0, Menu.NONE, "Option6");
		MenuItem sm7 = sub.add(0, 0, Menu.NONE, "Option7");
		MenuItem sm8 = sub.add(0, 0, Menu.NONE, "Option8");
	MenuItem m3 = menu.add(GroupId, ItemId + 1, OrderId + 1,"Third Menu");
	MenuItem m4 = menu.add(GroupId, ItemId + 1, OrderId + 1,"Fourth Menu");
	return true;
	}
}
```



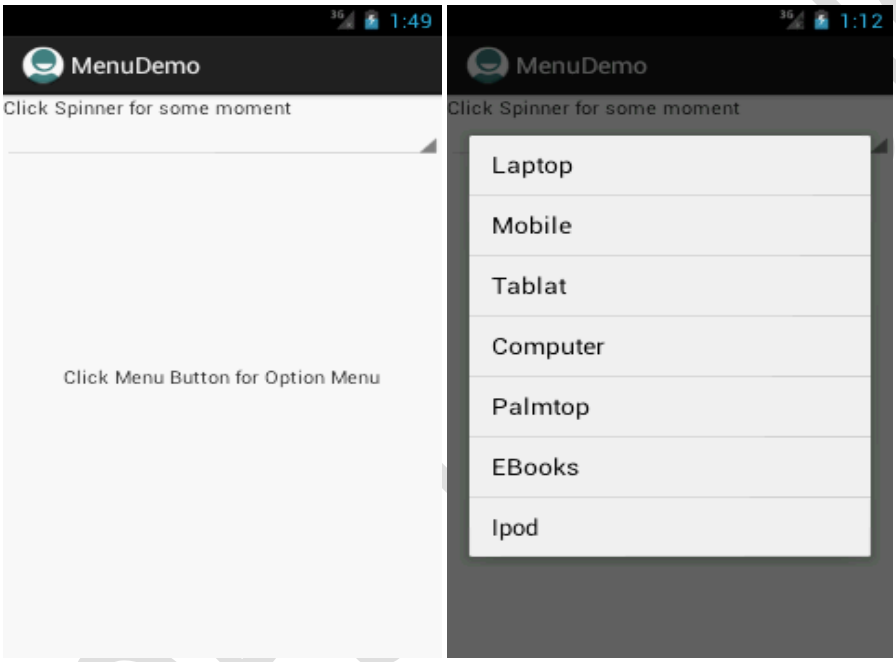
Using Context Menus

Context Menus are contextualized by the currently focused View and are triggered by pressing the trackball, middle D-pad button, or the View for around 3 seconds.

You define and populate Context Menus similarly to the Activity menu. There are two options available for creating Context Menus for a particular View.



Context menu example



In .java file

```
public class MenuDemo extends Activity {
	public ImageView img;
	public Spinner spn;
	public String[] items = { "Laptop", "Mobile", "Tablat", "Computer",
			"Palmtop", "EBooks", "Ipod" };
	public ArrayList<String> list = new ArrayList<String>();
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_menu_demo);
		img = (ImageView) findViewById(R.id.img1);
		spn = (Spinner) findViewById(R.id.spn1);
		spn.setAdapter(new ArrayAdapter<String>(this,
				android.R.layout.simple_spinner_item, list));
		registerForContextMenu(spn);
	}
```

```
	@Override
	public void onCreateContextMenu(ContextMenu menu, View v,
			ContextMenuInfo menuInfo) {
		// Create Context Menu
		super.onCreateContextMenu(menu, v, menuInfo);
		for (int i = 0; i < items.length; i++) {
			menu.add(0, i, i, items[i]);
		}
	}
	@Override
	public boolean onContextItemSelected(MenuItem item) {
		// TODO Auto-generated method stub
		for (int i = 0; i < items.length; i++) {
			if (item.toString() == items[i]) {
				list.add(item.getTitle().toString());
			}
		}
		spn.setAdapter(new ArrayAdapter<String>(this,
				android.R.layout.simple_spinner_item, list));
		return true;
	}
@Override
	publicboolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		int GroupId = 0;
		int ItemId = Menu.FIRST;
		int OrderId = Menu.NONE;
		MenuItem m1 = menu.add(GroupId, ItemId, OrderId, "First Menu");
		m1.setIcon(R.drawable.ic_action_search);
	SubMenu sub = menu.addSubMenu(GroupId, ItemId, OrderId, "SubMenu");
		sub.setHeaderTitle("Sub Menu");
		sub.setIcon(R.drawable.more);
		MenuItem sm1 = sub.add(0, 0, Menu.NONE, "Option1");
		MenuItem sm2 = sub.add(0, 0, Menu.NONE, "Option2");
		MenuItem sm3 = sub.add(0, 0, Menu.NONE, "Option3");
		MenuItem sm4 = sub.add(0, 0, Menu.NONE, "Option4");
		MenuItem sm5 = sub.add(0, 0, Menu.NONE, "Option5");
		MenuItem sm6 = sub.add(0, 0, Menu.NONE, "Option6");
		MenuItem sm7 = sub.add(0, 0, Menu.NONE, "Option7");
		MenuItem sm8 = sub.add(0, 0, Menu.NONE, "Option8");
	MenuItem m3 = menu.add(GroupId, ItemId + 1, OrderId + 1, "Third Menu");
	MenuItem m4 = menu.add(GroupId, ItemId + 1, OrderId + 1, "Fourth Menu");

m1.setOnMenuItemClickListener(new OnMenuItemClickListener() {
				publicboolean onMenuItemClick(MenuItem arg0) {
					img.setImageResource(R.drawable.blackberry);
					returntrue;
				}
	});
	returntrue;
	}
}
```

Dialogs

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

- **AlertDialog**

A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.

- **Toast**

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive.

- **DatePickerDialog or TimePickerDialog**

A dialog with a pre-defined UI that allows the user to select a date or time.

Showing Pop-Up Messages

Sometimes your activity (or other piece of Android code) will need to speak up. Something asynchronous may occur, such as an incoming message. In these and other cases, you may need to communicate with the user outside the bounds of the traditional user interface.

Of course, this is nothing new. Error messages in the form of dialog boxes have beenaround for a very long time. More-subtle indicators also exist, from task-tray icons to bouncing dock icons to a vibrating cell phone.

Android has quite a few systems for letting you alert your users outside the bounds of anActivity-based UI. You will see two means of raising pop-up messages:

toasts and **alerts**.

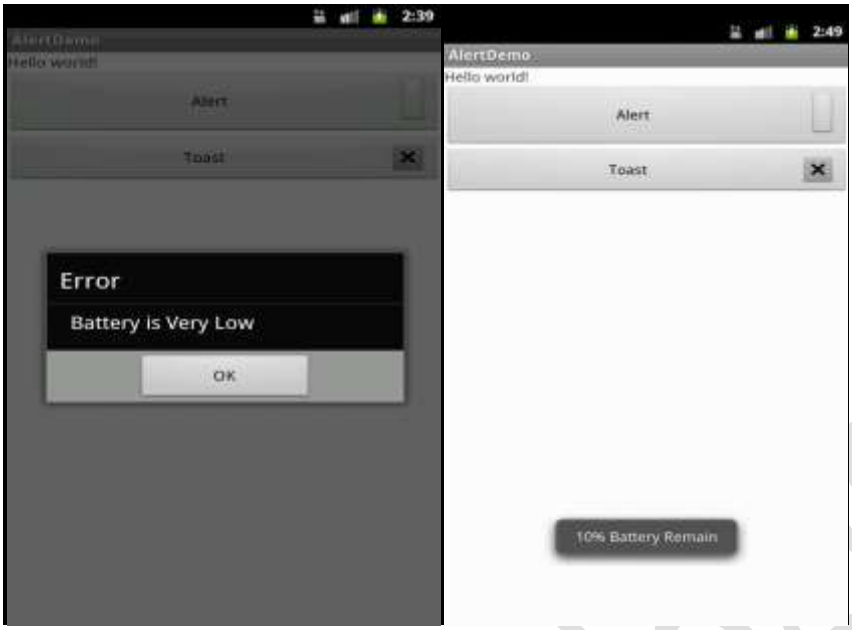
Alert! Alert!

- If you would prefer something in the more classic dialog-box style, what you want is an AlertDialog. As with any other modal dialog box, an AlertDialog pops up, grabs the focus, and stays there until closed by the user.
- You might use this for a critical error, a validation message that cannot be effectively displayed in the base activity UI or something else where you are sure that the user needs to see the message and needs to see it now.
- The simplest way to construct an AlertDialog is to use the Builder class. Following in true builder style, Builder offers a series of methods to configure an AlertDialog, each method returning the Builder for easy chaining. At the end, you call show() on the builder to display the dialog box.

Raising Toasts

- A Toast is a transient message, meaning that it displays and disappears on its own without user interaction.
- Toast is transient, you have no way of knowing if the user even notices it. You get no acknowledgment from them, nor does the message stick around for a long time to pester the user. Hence, the Toast is mostly for advisory messages, such as indicating a long-running background task is completed, the battery has dropped to a low-but-not-too-low level, etc.

Simple alert and toasts example



Create new project… in main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<TextView
        android:id="@+id/txt1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        tools:context=".AlertDemo" />
<Button
        android:id="@+id/alert1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/txt1"
        android:drawableRight="@android:drawable/btn_default"
        android:text="Alert" />
<Button
        android:id="@+id/toast1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/alert1"
        android:drawableRight="@android:drawable/btn_dialog"
        android:text="Toast" />
</RelativeLayout>
```

In PopUpDemo.java file

```
public class AlertDemo extends Activity implements OnClickListener {
      private Button btnAlert, btnToast;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_alert_demo);
```

```
            btnAlert = (Button) findViewById(R.id.alert1);
            btnToast = (Button) findViewById(R.id.toast1);
            btnAlert.setOnClickListener(this);
            btnToast.setOnClickListener(this);
      }
      @Override //Create Alert and Show Toast
      public void onClick(View v) {
            if (v == btnAlert) {
                  new AlertDialog.Builder(this)
                        .setTitle("Error")
                        .setMessage("Battery is Very Low")
                        .setNeutralButton("OK",
      new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,int which) {
            // TODO Auto-generated method stub}
                  }).show();
            } else {
            Toast.makeText(this, "10% Battery Remain",
Toast.LENGTH_LONG).show();
            }
      }
}
```

Progress Dialog

In .java file....


```
public class ProgressDialogDemo extends Activity {
    private ProgressDialog PBD;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        PBD = new ProgressDialog(this);
        ProgressDialog.show(this, "Working...", "Please wait until complate...");
    }
}
```

Date and Time picker Dialog

The DatePicker and DatePickerDialog allow you to set the starting date for the selection,in the form of a year, month, and day of month value. Note that the month runs from 0 for January through 11 for December. It is up to you to store that date someplace, particularly if you are using the dialog, since there is no other way for you to get at the chosen date later on.

Simple layout in main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<Button
        android:id="@+id/btnDate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Date" />
<Button
        android:id="@+id/btnTime"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/btnDate"
```

```
        android:text="Time" />
<TextView
        android:id="@+id/txt1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/btnTime"
        android:text="" />
</RelativeLayout>
```

in the **DateTimePicker.java** file

```
publicclass DateAndTime extends Activity {
      private Button btDate, btTime;
      private TextView txt;
      privateint years, months, days, hours, minutes;
      Calendar c = Calendar.getInstance();

      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_date_and_time);
            btDate = (Button) findViewById(R.id.btnDate);
            btTime = (Button) findViewById(R.id.btnTime);
            txt = (TextView) findViewById(R.id.txt1);
            btDate.setOnClickListener(new OnClickListener() {
                  publicvoid onClick(View arg0) {
                        // TODO Auto-generated method stub
                        new DatePickerDialog(DateAndTime.this, setDT, c
                                    .get(Calendar.YEAR), c.get(Calendar.MONTH),
                                    .get(Calendar.DAY_OF_MONTH)).show();
                  }
            });
            btTime.setOnClickListener(new OnClickListener() {
                  publicvoid onClick(View arg0) {
                        // TODO Auto-generated method stub
                  new TimePickerDialog(DateAndTime.this, setTM, c
                              .get(Calendar.HOUR_OF_DAY),
c.get(Calendar.MINUTE),
                                    true).show();
                  }
            });
            final Calendar c = Calendar.getInstance();
            years = c.get(Calendar.YEAR);
            months = c.get(Calendar.MONTH);
            days = c.get(Calendar.DAY_OF_MONTH);
            hours = c.get(Calendar.HOUR_OF_DAY);
            minutes = c.get(Calendar.MINUTE);
            setDateTimeLbl();
      }
      // Set TextView
      publicvoid setDateTimeLbl() {
            txt.setText("Date:" + days + "/" + months + "/" + years + " "
                        + padTime(hours) + ":" + padTime(minutes));
      }
```

```
	private String padTime(int t) {
		if (t <= 10) {
			return "0" + String.valueOf(t);
		} else {
			return String.valueOf(t);
		}
	}
	// Set Time
	private TimePickerDialog.OnTimeSetListener setTM = new
OnTimeSetListener() {
		publicvoid onTimeSet(TimePicker view, int hourOfDay, int minute) {
			// TODO Auto-generated method stub
			hours = hourOfDay;
			minutes = minute;
			setDateTimeLbl();
		}
	};
	// Set Date
	private DatePickerDialog.OnDateSetListener setDT = new
OnDateSetListener() {
		publicvoid onDateSet(DatePicker view, int year, int monthOfYear,
				int dayOfMonth) {
			// TODO Auto-generated method stub
			years = year;
			months = monthOfYear + 1;
			days = dayOfMonth;
			setDateTimeLbl();
		}
	};
}
```

Material Design Toolbar, Tab Layout :

Material Designing:

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. Android now includes support for material design apps. To use material design in your Android apps, follow the guidelines defined in the material design specification and use the new components and functionality available in Android 5.0 (API level 21) and above.

Android provides the following elements for you to build material design apps:

- A new theme
- New widgets for complex views
- New APIs for custom shadows and animations

android Material Design which was introduced in Android Lollipop version. In Material Design lot of new things were introduced like **Material Theme**, new **widgets**, **custom shadows**, **vector drawables** and custom **animations**. If you haven't working on Material Design yet, this article will give you a good start.

ANDROID TOOL BAR :

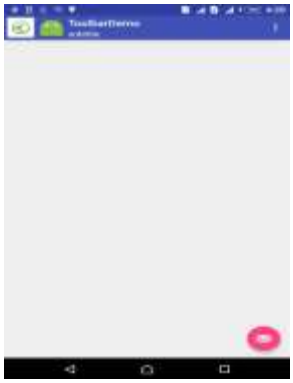
Introduction Of Toolbar:

- A Toolbar is similar to an ActionBar.
- It's a ViewGroup so you can place it anywhere in your layout. You can even replace the ActionBar with a Toolbar.
- Toolbar's are also more flexible. You can modify its size, color, position, etc. You can also add logos, labels, navigation icons and other views to it.
- With the release of Android 5.0 and material design, Android has updated the AppCompat support libraries so that we can use Toolbars on devices running API Level 7 and up.

Toolbar Implementation in Layout File:

```
<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay">

    </android.support.v7.widget.Toolbar>
</android.support.design.widget.AppBarLayout>
```

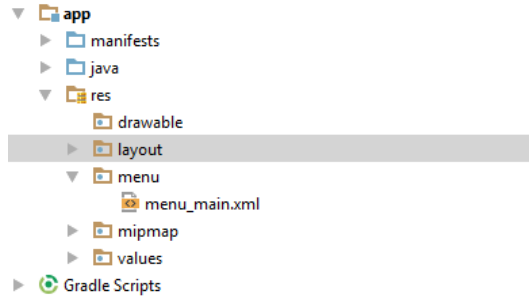
Add Icons in Toolbar:



Adding Menu in Toolbar:

Create one menu file in menu Directory

Go->res folder ->menu->menu_main.xml



Here the important xml attributes should be known are

android:title – Title for show items name

menu_main.xml

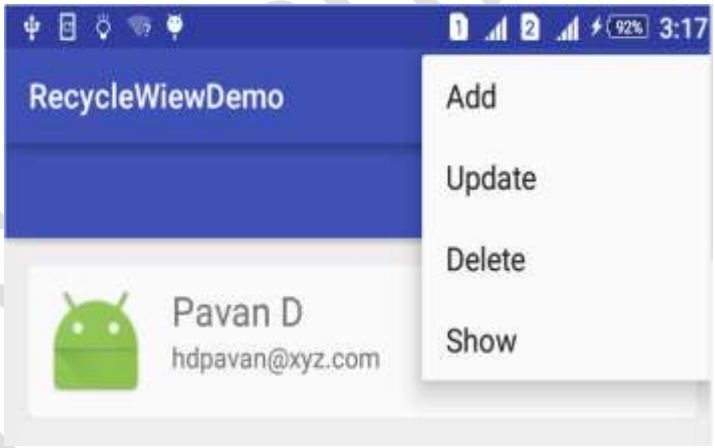
```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
<item android:id="@+id/add"
      android:title="Add"/>
<item android:id="@+id/update"
         android:title="Update"/>
<item android:id="@+id/delete"
      android:title="Delete"/>
<item android:id="@+id/show"
          android:title="Show"/>
</menu>
```

Implementation action on Toolbar icons and menu:

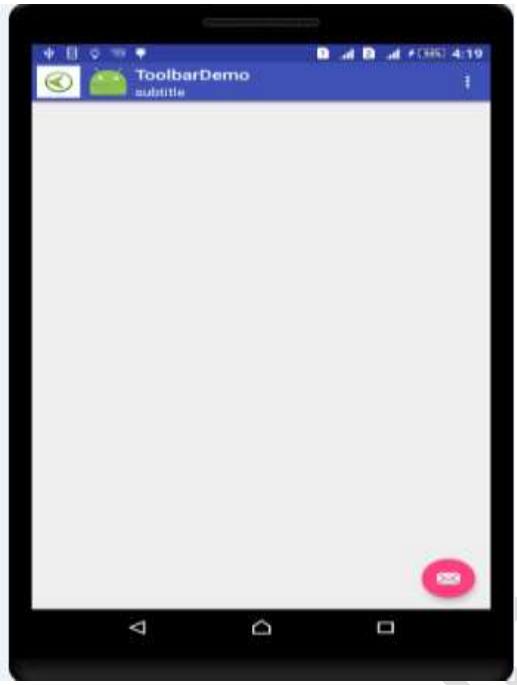
```
public class MainActivity extends AppCompatActivity {
     @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
//Setting appIcon
toolbar.setLogo(R.mipmap.ic_launcher);
//Setting navigationIcon
toolbar.setNavigationIcon(R.mipmap.dd);
//Setting Subtitle
toolbar.setSubtitle("subtitle");
//Setting menu
toolbar.inflateMenu(R.menu.menu_main);
//now set click listener on navigation  or  Back button
```

```
toolbar.setNavigationOnClickListener(new View.OnClickListener() {
   @Override
public void onClick(View v) {
      Toast.makeText(MainActivity.this, "Back Button Pressed", Toast.LENGTH_LONG).show();
   }
});
//now set click listener on menu item or option menu using toolbar
toolbar.setOnMenuItemClickListener(new Toolbar.OnMenuItemClickListener() {
   @Override
public boolean onMenuItemClick(MenuItem item) {
      Toast.makeText(MainActivity.this, "Button Pressed", Toast.LENGTH_LONG).show();
return true;
   }
});
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.menu_main,menu);
return super.onCreateOptionsMenu(menu);
}}
```

Now if you run the project, you can see the tool bar with action icons. You can also notice that an overflow icon shown which opens the unimportant action items as a drop down menu.



Finally Output Display



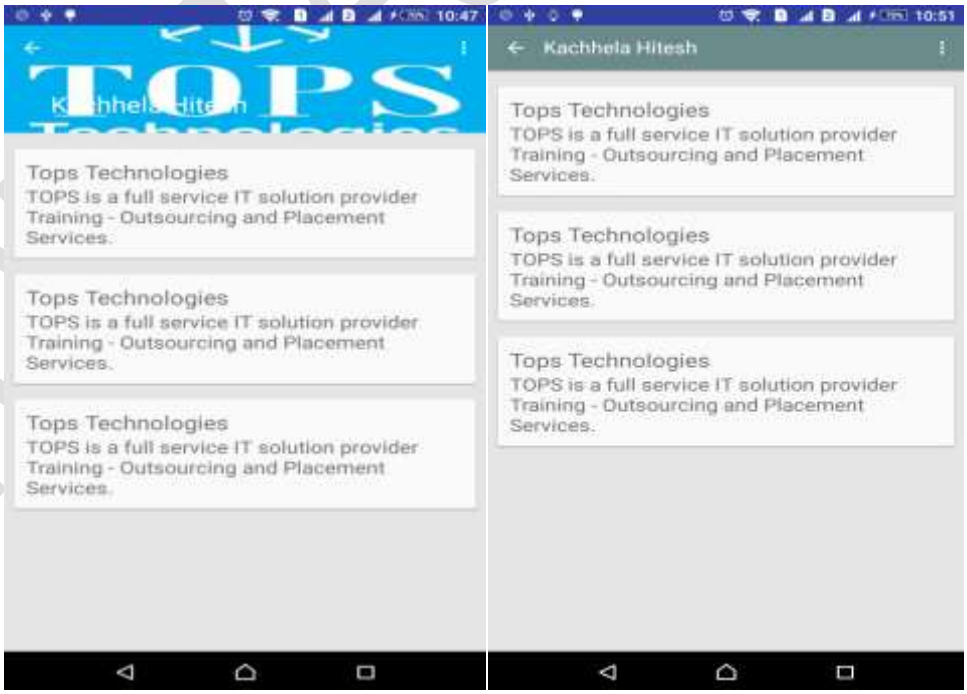
Android Collapsing Toolbar Layout

What is Collapsing Toolbar Layout?

- CollapsingToolbarLayout is the newly introduced in Lollipop , using which you can create awesome scrolling effects .
- According to the android documentation Collapsing

Toolbar layout is a wrapper for Toolbar which implements a collapsing app bar.

- It is designed to be used as a direct child of a AppBarLayout



Required Dependencies For Implementing Collapsing Toolbar Layout

```
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   compile 'com.android.support:appcompat-v7:23.0.1'
   compile 'com.android.support:design:23.0.1'
   compile 'com.android.support:cardview-v7:23.0.1'
   compile 'com.android.support:palette-v7:22.2.0'
}
```

XML Layout

Create XML layout file in res/layout and name it activity_main.xml

file : activity_main.xml

```
<android.support.design.widget.CoordinatorLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
           xmlns:app="http://schemas.android.com/apk/res-auto"
          android:layout_width="match_parent"
          android:layout_height="match_parent">

   <android.support.design.widget.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="250dp"
      android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

      <android.support.design.widget.CollapsingToolbarLayout
         android:id="@+id/collapsing_toolbar"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         app:contentScrim="?attr/colorPrimary"
         app:layout_scrollFlags="scroll|exitUntilCollapsed">

          <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
                android:background="@drawable/profile_pic"
                app:layout_collapseMode="parallax" />

         <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:layout_collapseMode="pin" />

      </android.support.design.widget.CollapsingToolbarLayout>
```

```
  </android.support.design.widget.AppBarLayout>
<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffe5e5e5"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">
    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      android:paddingTop="10dp">
      <include layout="@layout/card_layout" />
      <include layout="@layout/card_layout" />
      <include layout="@layout/card_layout" />
    </LinearLayout>
  </android.support.v4.widget.NestedScrollView>
</android.support.design.widget.CoordinatorLayout>
```

XML Layout

Create a XML Layout inside res/layout by filename card_layout.xml and add cardview to it .

file : card_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_margin="10dp">
   <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="vertical"
     android:padding="12dp">

       <TextView
              android:layout_width="match_parent"
             android:layout_height="wrap_content"
            android:text="Lorem ipsum"
            android:textSize="20sp" />
       <TextView
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
            android:text="@string/text_content"
            android:textSize="18sp" />
  </LinearLayout>
```

</android.support.v7.widget.CardView>

Lets Understand the XML Tags in the above xml layout

1. CoordinatorLayout

A powerful FrameLayout that specifies behavior for child views for various interactions. Allows floating views to be anchored in layout.

2. AppBarLayout

Is essentially a LinearLayout (vertical). It helps respond to its children's scroll events (scroll gestures). Responsible for implementing many features of material design's app bar. Depends heavily on being used as a direct child within CoordinatorLayout.

3. CollapsingToolbarLayout

Wrapper for Toolbar that makes the header image collapse into the Toolbar adjusting its title size.What's left is the ImageView which holds our actual header's image and Toolbar which we're familiar with.

4. NestedScrollView

1. It's an special scroll view for the smooth scrolling effect, inside this place the desired content . Here in this example will add several Cards as its children.
   Straight from the **developer's blog**:
   Flags include :
   scroll: this flag should be set for all views that want to scroll off the screen - for views that do not use this flag, they'll remain pinned to the top of the screen.
2. **enterAlways**: this flag ensures that any downward scroll will cause this view to become visible, enabling the 'quick return' pattern .
3. **enterAlwaysCollapsed**: When your view has declared a minHeight and you use this flag, your View will only enter at its minimum height (i.e., 'collapsed'), only re-expanding to its full height when the scrolling view has reached it's top.
4. **exitUntilCollapsed**: this flag causes the view to scroll off until it is 'collapsed' (its minHeight) before exiting .

Note : all views using the scroll flag must be declared before views that do not use the flag.This ensures that all views exit from the top, leaving the fixed elements behind.

For CollapsingToolbarLayout XML Tag set the layout_scrollFlags property with
scroll|exitUntilCollapsed

Collapse Mode

Parallax scrolling with the ImageView is achieved by simply setting its layout_collapseMode to parallax.

- The Toolbar must use pin as its collapseMode because we want the Toolbar to persist and remain on top as the user scrolls down.

Dynamic TextAppearance :

Setting The Title Large when the layout is expanded(layout is fully visible) and Smaller when the layout is collapsed(layout is scrolled off screen)
Add the following style code inside the style.xml

```
<style name="expandedappbar" parent="@android:style/TextAppearance.Medium">
    <item name="android:textSize">48sp</item>
    <item name="android:textColor">@color/white</item>
    <item name="android:textStyle">bold</item>
  </style>
```

```
  <style name="collapsedappbar" parent="@android:style/TextAppearance.Medium">
    <item name="android:textSize">18sp</item>
    <item name="android:textColor">@color/white</item>
  </style>
```

The Complete MainActivity Source code

file : MainActivity.java

```
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
public class MainActivity extends AppCompatActivity {
 private CollapsingToolbarLayout collapsingToolbarLayout = null;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
          Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
            setSupportActionBar(toolbar);
          ActionBar actionBar = getSupportActionBar();
          actionBar.setDisplayHomeAsUpEnabled(true);
           collapsingToolbarLayout = (CollapsingToolbarLayout)
          findViewById(R.id.collapsing_toolbar);
     collapsingToolbarLayout.setTitle(getResources().getString(R.string.user_name));
     dynamicToolbarColor();
     toolbarTextAppernce();
   }
private void dynamicToolbarColor() {
     Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
          R.drawable.profile_pic);
     Palette.from(bitmap).generate(new Palette.PaletteAsyncListener() {
       @Override
       public void onGenerated(Palette palette) {
 collapsingToolbarLayout.setContentScrimColor(palette.getMutedColor(R.attr.colorPrimary));
 collapsingToolbarLayout.setStatusBarScrimColor(palette.getMutedColor(R.attr.colorPrimaryDark);
     }});}
  private void toolbarTextAppernce() {
     collapsingToolbarLayout.setCollapsedTitleTextAppearance(R.style.collapsedappbar);
     collapsingToolbarLayout.setExpandedTitleTextAppearance(R.style.expandedappbar);
   }
}
```

Note:

- **You Need To ad images and style .xml in your project**
- **Also You need add entry in String.xml file…**
- <**string name="text_content"**>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin consectetur diam id aliquam scelerisque. Donec ultrices lacus vel </**string**>
  <**string name="user_name"**>Android</**string**>

Material Design Tabs

What is TabLayout?

- TabLayout provides a horizontal layout to display tabs. The Tabs inside the TabLayout is Scroll-able and is also used to switch between different views .
- The design support library simplifies the process of Creating TabLayout widget and adding tabs to it .
- **Required Dependencies For Implementing Collapsing Toolbar Layout**

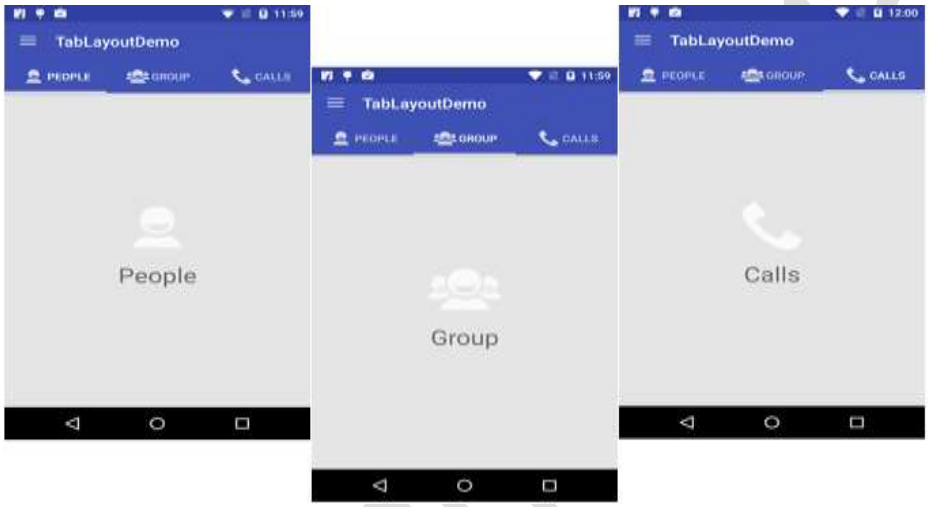
```
dependencies {
        compile fileTree(dir:'libs',include:['*.jar'])
        compile     'com.android.support:appcompat-v7:23.0.1'
        compile     'com.android.support:design:23.0.1'
}
```

```
public class TabsActivity extends AppCompatActivity {
private Toolbar toolbar;
private TabLayout tabLayout;
private ViewPager viewPager;
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_simple_tabs);
toolbar = (Toolbar) findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);
      getSupportActionBar().setDisplayHomeAsUpEnabled(true);
viewPager = (ViewPager) findViewById(R.id.viewpager);
      setupViewPager(viewPager);
tabLayout = (TabLayout) findViewById(R.id.tabs);
       ViewPagerAdapter adapter = new   ViewPagerAdapter(getSupportFragmentManager());
            adapter.addFragment(new OneFragment(), "ONE");
             adapter.addFragment(new TwoFragment(), "TWO");
             adapter.addFragment(new ThreeFragment(), "THREE");
             viewPager.setAdapter(adapter);
class ViewPagerAdapter extends FragmentPagerAdapter {
private final List<Fragment>mFragmentList = new ArrayList<>();
private final List<String>mFragmentTitleList = new ArrayList<>();
public ViewPagerAdapter(FragmentManager manager) {
super(manager);
   }
   @Override
public Fragment getItem(int position) {
return mFragmentList.get(position);
   }
   @Override
public int getCount() {
return mFragmentList.size();
   }
public void addFragment(Fragment fragment, String title) {
mFragmentList.add(fragment);
```

```
    mFragmentTitleList.add(title);
  }
  @Override
  public CharSequence getPageTitle(int position) {
    return mFragmentTitleList.get(position);
  }
}
```

Android Material Design Tablayout With icon

- In this tutorial we will see how to create tab layout with icons .



Create a resource file in the res/values folder called colors.xml. Modify it as below:
file : colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="primary">#3F51B5</color>
  <color name="primary_dark">#303F9F</color>
  <color name="accent">#ffffff</color>
  <color name="fragment_bg">#e5e5e5</color>
</resources>
```

STYLE:
MODIFY RES/VALUES/STYLES.XML AS BELOW:

```
<resources>
  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
  </style>
</resources>
```

XML LAYOUT :

create a xml layout file activity_main.xml in the res/layout . which contains coordinate layout as parent view and isnide this add appbarlayout and viewpager.

Appbarlayout : the design library provides the appbarlayout inside this view add toolbar and tablayout .

Viewpager : the viewpager will be used to enable horizontal paging between tabs.

file : activity_main.xml

```
<android.support.design.widget.CoordinatorLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">
    <android.support.v7.widget.Toolbar
      android:id="@+id/toolbar"
      android:layout_width="match_parent"
      android:layout_height="?attr/actionBarSize"
      android:background="?attr/colorPrimary"
      app:layout_scrollFlags="scroll|enterAlways" />
<android.support.design.widget.TabLayout
      android:id="@+id/tablayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      app:tabGravity="fill"
      app:tabMode="fixed" />
  </android.support.design.widget.AppBarLayout>
  <android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</android.support.design.widget.CoordinatorLayout>
```

Create a Fragment for each tab in tablayout .

file : people_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/fragment_bg">

  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="148dp"
    android:src="@drawable/ic_action_person" />

 <TextView
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_below="@+id/imageView1"
    android:gravity="center"
    android:text="People"
    android:textSize="30sp" />
</RelativeLayout>
```

Create PeopleFragment and extend this class to Fragment and inflate this fragment with above defined xml layout .

file : PeopleFragment

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.tutorialsbuzz.tablayoutdemo.R;
public class PeopleFragment extends Fragment {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }
@Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
        return inflater.inflate(R.layout.people_fragment, container, false);
  }
}
```

Create a Fragment for each tab in tablayout .

file : group_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/fragment_bg">

   <ImageView
     android:id="@+id/imageView1"
     android:layout_width="100dp"
     android:layout_height="100dp"
     android:layout_alignParentTop="true"
     android:layout_centerHorizontal="true"
     android:layout_marginTop="148dp"
     android:src="@drawable/ic_action_group" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:layout_below="@+id/imageView1"
    android:gravity="center"
    android:text="Group"
    android:textSize="30sp" />
</RelativeLayout>
```

Create GroupFragment and extend this class to Fragment and inflate this fragment with above defined xml layout .

file : GroupFragment

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.tutorialsbuzz.tablayoutdemo.R;
public class GroupFragment extends Fragment {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }
@Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.group_fragment, container, false);
  }
}
```

XML Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/fragment_bg">

  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="148dp"
    android:src="@drawable/ic_action_call" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/imageView1"
    android:gravity="center"
    android:text="Calls"
```

```
    android:textSize="30sp" />
</RelativeLayout>
```

Create CallFragment and extend this class to Fragment and inflate this fragment with above defined xml layout .

file : CallFragment

```
public class CallFragment extends Fragment {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.call_fragment, container, false);
  }
}
```

FragmentStatePagerAdapter:

- Create a ViewPagerAdapter Class and extend this class to FragmentStateViewPager .
- Override the getItem ,getCount , getPageTitle methods of FragmentStateViewPager class .
- Have a method addFragment which takes fragment object and title, inside this method we will create a list of fragment with it's title for each of the tabs .

file : ViewPagerAdapter.java

```
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import java.util.ArrayList;
import java.util.List;
public class ViewPagerAdapter extends FragmentStatePagerAdapter {
  private final List<Fragment> mFragmentList = new ArrayList<>();
  private final List<String> mFragmentTitleList = new ArrayList<>();

  public ViewPagerAdapter(FragmentManager fm) {
    super(fm);
  }
  @Override
  public Fragment getItem(int position) {
    return mFragmentList.get(position);
  }
 @Override
  public int getCount() {
    return mFragmentList.size();
  }
  @Override
  public CharSequence getPageTitle(int position) {
```

```
    return mFragmentTitleList.get(position);
  }
  public void addFragment(Fragment fragment, String title) {
    mFragmentList.add(fragment);
    mFragmentTitleList.add(title);
  }
}
```

MainActivity:

- Create a MainActivity class which extends Activity class
- Override the onCreate method inside this method set the content of this activity with above defined activity_main.xml layout .
- Inside the setupViewPager method make a call to addFragment method on adapter object and then call setAdapter on ViewPager reference by passing the adapter object.
- To Setting Icons for Tabs Tablayout , get the refrence to tabs at each position and upon which call setIcon by passing the drawable resource id .

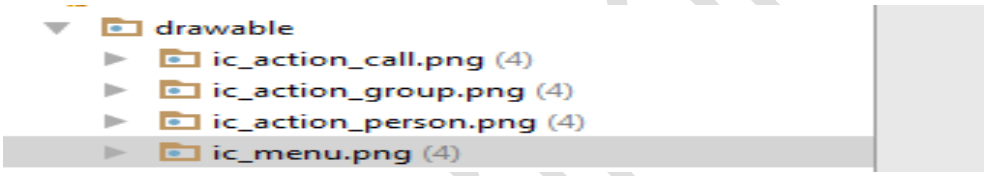
file : MainActivity.java

```
import android.support.design.widget.TabLayout;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBar;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import com.tutorialsbuzz.tablayoutdemo.TabFragments.PeopleFragment;
import com.tutorialsbuzz.tablayoutdemo.TabFragments.CallFragment;
import com.tutorialsbuzz.tablayoutdemo.TabFragments.GroupFragment;
public class MainActivity extends AppCompatActivity {
  private Toolbar toolbar;
   private TabLayout tabLayout;
   private ViewPager viewPager;
  private int[] tabIcons = {
        R.drawable.ic_action_person,
        R.drawable.ic_action_group,
       R.drawable.ic_action_call
  };
@Override
   protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     toolbar = (Toolbar) findViewById(R.id.toolbar);
     setSupportActionBar(toolbar);
     ActionBar actionBar = getSupportActionBar();
     actionBar.setHomeAsUpIndicator(R.drawable.ic_menu);
     actionBar.setDisplayHomeAsUpEnabled(true);
     viewPager = (ViewPager) findViewById(R.id.viewpager);
     setupViewPager(viewPager);
     tabLayout = (TabLayout) findViewById(R.id.tablayout);
     tabLayout.setupWithViewPager(viewPager);
```

```
        setupTabIcons();
    }
 private void setupViewPager(ViewPager viewPager) {
        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFragment(new PeopleFragment(), "People");
        adapter.addFragment(new GroupFragment(), "Group");
        adapter.addFragment(new CallFragment(), "Calls");
        viewPager.setAdapter(adapter);
    }
  private void setupTabIcons() {
        tabLayout.getTabAt(0).setIcon(tabIcons[0]);
        tabLayout.getTabAt(1).setIcon(tabIcons[1]);
        tabLayout.getTabAt(2).setIcon(tabIcons[2]);
    }
}
```

Note:

- **You Need To add images 4 images in drawable folder.**



Android Circular Reveal Effect Like in Whatsapp

What is Circular Reveal?

- In Android whatsapp while sending media files you can notice a attachment icon on right side of the toolbar which performs a neat toggle circular effect , as you can see in below gif image which is exactly what I am talking about in this tutorial we will see how to mimic this effect .

What is Reveal Effect?

- The Reveal Effect uses the **RippleDrawable** class to achieve this . So what is "reveal"? It is just an animation which animates view's clipping boundaries. Android provides really convenient helper method to create this animation.
- The reveal is intended to for LOLIPOP and Above version , this effect does not work with below LOLIPOP.

However there is a **Library** the manages to support to work with lower version , this library is very convenient to use .Though one must be careful about the imports.

Note:

- **You Need add dependencies in gradle.bulild**

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.1.0'
    compile ('com.github.ozodrukh:CircularReveal:1.1.1@aar') {
        transitive = true;
    }
}
```

Note:

You Need add dependencies in file main_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context=".MainActivity">
   <item
      android:id="@+id/action_settings"
      android:orderInCategory="100"
      android:title="@string/action_settings"
      app:showAsAction="never" />
   <item
      android:id="@+id/action_clip"
      android:icon="@drawable/ic_attach"
      android:title="Media"
      app:showAsAction="ifRoom" />
</menu>
```

file : toolbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:id="@+id/toolbar"
   android:layout_width="match_parent"
   android:layout_height="?attr/actionBarSize"
   android:background="?attr/colorPrimary"
   android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
   app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
```

file : media_attach_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/reveal_items"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
    <!--row 1 -->
     <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="#F0EFED"
      android:orientation="horizontal"
      android:padding="16dp">
<!--Gallery Icon -->
      <LinearLayout
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
```

```
        android:gravity="center"
        android:orientation="vertical">
         <ImageButton
           android:id="@+id/gallery_img_btn"
           android:layout_width="70dp"
           android:layout_height="70dp"
           android:background="@drawable/gallery" />
         <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="16dp"
           android:text="Gallery" />
      </LinearLayout>
<!--Photo Icon -->
      <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
         android:gravity="center"
         android:orientation="vertical">
          <ImageButton
            android:id="@+id/photo_img_btn"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:background="@drawable/photo" />
          <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="Photo" />
      </LinearLayout>
<!--Video Icon -->
      <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="vertical">
         <ImageButton
           android:id="@+id/video_img_btn"
           android:layout_width="70dp"
           android:layout_height="70dp"
           android:background="@drawable/video" />
         <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="16dp"
```

```
            android:text="Video" />
        </LinearLayout>
    </LinearLayout>
```

file : activity_main.xml

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context="com.suleiman.material.activities.RevealAnimation">
    <include layout="@layout/toolbar" />
     <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="?attr/actionBarSize">
            <io.codetail.widget.RevealFrameLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content">
                <include layout="@layout/media_attach_menu" />
            </io.codetail.widget.RevealFrameLayout>
    </FrameLayout>
</FrameLayout>
```

Parameters of createCircularReveal methods .

createCircularReveal(View view, int cx, int cy, float startRadius, float endRadius)

view - view to reveal

cx- start X coordinate of reveal

cy- start Y coordinate of reveal

startRadius - start radius. In most cases - 0

endRadius - end radius - depends on your view's bounds .

- **Toggle Effect**

We will use a simple boolean hidden to handle the toggling and update it's value inside if-else.

- Onclick of Image Button inside Action Option we will make a call to hideRevealView method , inside this method we will hide the RevealFrameLayout .

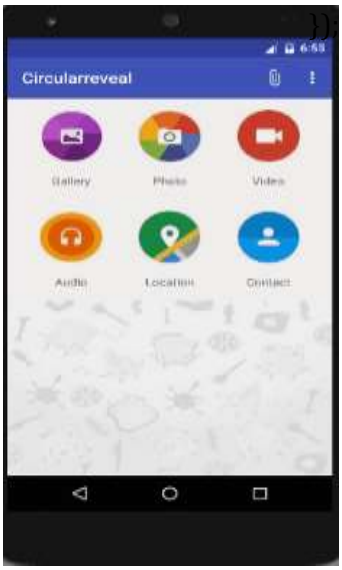
file : MainActivity.java

```
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.os.Build;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.AccelerateDecelerateInterpolator;
```

```
import android.widget.ImageButton;
import android.widget.LinearLayout;
import io.codetail.animation.SupportAnimator;
import io.codetail.animation.ViewAnimationUtils;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private LinearLayout mRevealView;
   private boolean hidden = true;
   private ImageButton gallery_btn, photo_btn, video_btn, audio_btn, location_btn, contact_btn;
    @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      initView();
   }
private void initView() {
      Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);
      mRevealView = (LinearLayout) findViewById(R.id.reveal_items);
      mRevealView.setVisibility(View.GONE);
      gallery_btn = (ImageButton) findViewById(R.id.gallery_img_btn);
      photo_btn = (ImageButton) findViewById(R.id.photo_img_btn);
      video_btn = (ImageButton) findViewById(R.id.video_img_btn);
      audio_btn = (ImageButton) findViewById(R.id.audio_img_btn);
      location_btn = (ImageButton) findViewById(R.id.location_img_btn);
      contact_btn = (ImageButton) findViewById(R.id.contact_img_btn);
       gallery_btn.setOnClickListener(this);
      photo_btn.setOnClickListener(this);
      video_btn.setOnClickListener(this);
      audio_btn.setOnClickListener(this);
      location_btn.setOnClickListener(this);
      contact_btn.setOnClickListener(this);
   }
@Override
   public void onClick(View v) {
      hideRevealView();
      switch (v.getId()) {
          case R.id.gallery_img_btn:
             break;
          case R.id.photo_img_btn:
             break;
          case R.id.video_img_btn:
             break;
          case R.id.audio_img_btn:
             break;
          case R.id.location_img_btn:
```

```
            break;
         case R.id.contact_img_btn:
            break;
      }
   }
private void hideRevealView() {
      if (mRevealView.getVisibility() == View.VISIBLE) {
         mRevealView.setVisibility(View.GONE);
         hidden = true;
      }
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
   }
@Override
   public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {

         case R.id.action_clip:

            int cx = (mRevealView.getLeft() + mRevealView.getRight());
            int cy = mRevealView.getTop();
            int radius = Math.max(mRevealView.getWidth(), mRevealView.getHeight());

            //Below Android LOLIPOP Version
            if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
               SupportAnimator animator =
                     ViewAnimationUtils.createCircularReveal(mRevealView, cx, cy, 0, radius);
               animator.setInterpolator(new AccelerateDecelerateInterpolator());
               animator.setDuration(700);

            @Override
                  public void onAnimationCancel() {
                  }
            @Override
                  public void onAnimationRepeat() {
                  }
                });
                animator_reverse.start();
               }
            }
            SupportAnimator animator_reverse = animator.reverse();
                if (hidden) {
                  mRevealView.setVisibility(View.VISIBLE);
                  animator.start();
```

```
            hidden = false;
        } else {
            animator_reverse.addListener(new SupportAnimator.AnimatorListener() {
                @Override
                    public void onAnimationStart() {
                }
    @Override
                    public void onAnimationEnd() {
                    mRevealView.setVisibility(View.INVISIBLE);
                    hidden = true;
                }
    else {
        if (hidden) {
            Animator anim = android.view.ViewAnimationUtils.
                                createCircularReveal(mRevealView, cx, cy, 0, radius);
            mRevealView.setVisibility(View.VISIBLE);
            anim.start();
            hidden = false;
        } else {
            Animator anim = android.view.ViewAnimationUtils.
                                createCircularReveal(mRevealView, cx, cy, radius, 0);
            anim.addListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    super.onAnimationEnd(animation);
                    mRevealView.setVisibility(View.INVISIBLE);
                    hidden = true;
                }

            anim.start();
        }
    }
    return true;
    case android.R.id.home:
                supportFinishAfterTransition();
                return true;
    }
    return super.onOptionsItemSelected(item);
}}
```



RecyclerView With CardView

What is RecyclerView?

- RecyclerView is more flexible that *ListView* even if it introduces some complexities.
- We all know how to use ListView in our app and we know if we want to increase the ListView performances we can use a pattern called *ViewHolder*.
- This pattern consists of a simple class that holds the references to the UI components for each row in the ListView.

- This pattern avoids looking up the UI components all the time the system shows a row in the list. Even if this pattern introduces some benefits, we can implement the ListView without using it at all.
- RecyclerView forces us to use the *ViewHolder pattern*. To show how we can use the RecyclerView, we can suppose we want to create a simple app that shows a list of contact cards.

RecyclerView is very similar to the ListView and we can use them in the same way:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:paddingLeft="@dimen/activity_horizontal_margin"
       android:paddingRight="@dimen/activity_horizontal_margin"
       android:paddingTop="@dimen/activity_vertical_margin"
       android:paddingBottom="@dimen/activity_vertical_margin"
       tools:context=".MyActivity">
       <android.support.v7.widget.RecyclerView
              android:id="@+id/cardList"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
        />
</RelativeLayout>
```

Note:

As you'll notice from the layout shown above, the RecyclerView is available in the Android support library, so we have to modify build.gradle to include this dependency:

Follow Some Steps To add Recycler view Library in our Project:

Go->File menu->Select Project Structure->Select app in Module Section->then Select Dependencies->Click on Green (+)Plus Sign->1 Library Depencencies ->then appear list and select ->recyclerview-v7 library then Press Ok.

What is CardView?

- The CardView UI component shows information inside cards.
- We can customise its corners, the elevation and so on.
- We want to use this component to show contact information.
- These cards will be the rows of RecyclerView and we will see later how to integrate these two components.

Note:

- Follow Previous same Steps To add CardView Library in our Project

Like Recyclerview Library add in our Project

Library Name: **cardview-v7:21.0.0-rc1'**

```
<android.support.v7.widget.CardView
      xmlns:card_view="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/card_view"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      card_view:cardCornerRadius="4dp"
      android:layout_margin="5dp">

  <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent">

     <TextView
         android:id="@+id/title"
         android:layout_width="match_parent"
         android:layout_height="20dp"
         android:background="@color/bkg_card"
         android:text="contact det"
         android:gravity="center_vertical"
         android:textColor="@android:color/white"
         android:textSize="14dp"/>

```

Implementation ofAndroid RecyclerView With CardView:

Note: Must add this libraries in project.

```
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   compile 'com.android.support:appcompat-v7:23.0.1'
   compile 'com.android.support:design:23.0.1'
   compile 'com.android.support:cardview-v7:23.0.1'
   compile 'com.android.support:recyclerview-v7:23.0.1'
}
```

[A].Before we start coding we have initialize all the necessary string-array constant inside res/strings.xml and we also kept all the necessary images inside the drawable folder of the project .

file : strings.xml

```
<resources>
   <string name="app_name">RecycleViewDemo</string>
   <string name="action_settings">Settings</string>
   <!-- Names -->
   <string-array name="names">
      <item>Pavan D</item>
      <item>DayaSagar</item>
```

```
    <item>Pavan B</item>
    <item>Maddy</item>
    <item>Pavan H</item>
    <item>Nikhil</item>
  </string-array>
<!-- Pictures -->
  <array name="profile_pics">
    <item>@drawable/pavand_pic</item>
    <item>@drawable/daya_pic</item>
    <item>@drawable/pavanb_pic</item>
    <item>@drawable/maddy_pic</item>
    <item>@drawable/pavanh_pic</item>
    <item>@drawable/nikhil_pic</item>
  </array>
  <!-- Email -->
  <string-array name="email">
    <item>hdpavan@xyz.com</item>
    <item>daya@xyz.com</item>
    <item>pavanb@xyz.com</item>
    <item>maddy@xyz.com</item>
    <item>pavanh@xyz.com</item>
    <item>nikhil@xyz.com</item>
  </string-array>
</resources>
```

[B].Create XML Layout:

Create XML layout file in res/layout and name it activity_main.xml with LinearLayout as a parent view add the toolbar widget inside the appbarlayout and then add recyclerView as to child to linear layout .

file : activity_main.xml

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
   <android.support.design.widget.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark">
<android.support.design.widget.CoordinatorLayout
          android:id="@+id/coordinator"
          android:layout_width="match_parent"
          android:layout_height="wrap_content">
         <android.support.v7.widget.Toolbar
           android:id="@+id/toolbar"
```

```
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways" />
        </android.support.design.widget.CoordinatorLayout>
  </android.support.design.widget.AppBarLayout>
      <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />
</LinearLayout>
```

[C].Create a XML Layout inside res/layout and name it cardview_row_item.xml and add cardview to it .
In the previous tutorial we have cover on android cardview and it's customization .

file : cardview_row_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">
   <android.support.v7.widget.CardView
      android:id="@+id/cv"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
<RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <ImageView
           android:id="@+id/profile_pic"
           android:layout_width="70dp"
           android:layout_height="70dp"
           android:layout_alignParentLeft="true"
           android:layout_alignParentTop="true"
           android:layout_marginRight="16dp" />
        <TextView
           android:id="@+id/member_name"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignParentTop="true"
           android:layout_toRightOf="@+id/profile_pic"
           android:textSize="20sp" />
<TextView
           android:id="@+id/member_email"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/member_name"
            android:layout_toRightOf="@+id/profile_pic" />
      </RelativeLayout>
   </android.support.v7.widget.CardView>
</LinearLayout>
```

[D].Java Bean

Create a Bean Class Member which defines the data for items of recyclerview .

file : Member.java

```
public class Member {
   String name;
   String email;
   int photoID;
   Member(String name, String email, int photoID) {
      this.name = name;
      this.email = email;
      this.photoID = photoID;
   }
  public String getName() {
      return name;
   }
   public String getEmail() {
      return email;
   }
   public int getPhotoID() {
      return photoID;
   }
   public void setName(String name) {
      this.name = name;
   }
  public void setEmail(String email) {
      this.email = email;
   }
  public void setPhotoID(int photoID) {
      this.photoID = photoID;
   }
}
```

[E]. RecyclerView Adapter

- Create a Adapter That RecyclerView Can Use .

- Create A Class RVAdapter and this class Must Extend RecyclerView.Adapter. This adapter follows the view holder design pattern, which means that you have to define a custom class that extends RecyclerView.ViewHolder (This pattern minimizes the number of calls to the costly findViewById method.) .

- So Create a MemberViewHolder inside the RVAdapter and this class Must Extend RecyclerView.ViewHolder . we already defined the XML layout cardview_row_item.xml , We are going to reuse that layout now. Inside the constructor of our custom ViewHolder, initialize the views that belong to the items of our RecyclerView. Next, add a constructor to the custom adapter so that it has a handle to the data that the RecyclerView displays , As our data is in the form of a List of Member objects.

RecyclerView.Adapter has three abstract methods that we must override .

- **getItemCount() :** This method return the number of items present in the data
- **2. onCreateViewHolder() :** Inside this method we specify the layout that each item of the RecyclerView should use. This is done by inflating the layout using LayoutInflater, passing the output to the constructor of the custom ViewHolder.
- **3. onBindViewHolder() :** This Method This method is very similar to the getView method of a ListView's adapter. In our example, here's where you have to set the values of the name, email and photoID data member of Member class to the respective view inside the CardView.

file : RVAdapter.java

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.MemberViewHolder> {
   private List<Member> members;
   private Context context;
   public RVAdapter(List<Member> members, Context context) {
      this.members = members;
      this.context = context;     }
   public class MemberViewHolder extends RecyclerView.ViewHolder {
      private CardView cardView;
      private TextView name_tv;
      private TextView email_tv;
      private ImageView pic_iv;
public MemberViewHolder(View itemView) {
         super(itemView);
         cardView = (CardView) itemView.findViewById(R.id.cv);
         name_tv = (TextView) itemView.findViewById(R.id.member_name);
         email_tv = (TextView) itemView.findViewById(R.id.member_email);
         pic_iv = (ImageView) itemView.findViewById(R.id.profile_pic);
      }   }
   @Override
   public void onBindViewHolder(MemberViewHolder memberViewHolder, int i) {
      memberViewHolder.name_tv.setText(members.get(i).getName());
      memberViewHolder.email_tv.setText(members.get(i).getEmail());
      memberViewHolder.pic_iv.setImageResource(members.get(i).getPhotoID());
   }
@Override
   public MemberViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
      View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.cardview_row_item,
viewGroup, false);
      MemberViewHolder memberViewHolder = new MemberViewHolder(view);
```

```
    return memberViewHolder;
  }
  @Override
  public int getItemCount() {
    return members.size();
  }}
```

File:MainActivity .java

1. Create a class MainActivity and extend it to AppCompatActivity .
   2. Inside the onCreate method set the content of this activity with above defined xml layout and then set-up toolbar and also initialize the data member of Member class
2. **Setting LayoutManager** .
   Unlike a ListView, a RecyclerView needs a LayoutManager to manage the positioning of its items
   you could simply use one of the predefined LayoutManager subclasses:
   1.LinearLayoutManager.
   2.GridLayoutManager.
   3.StaggeredGridLayoutManager.

file : MainActivity.java

```
public class MainActivity extends AppCompatActivity {
  private RecyclerView recyclerview;
  private String[] names;
  private TypedArray profile_pics;
  private String[] emails;
  private List<Member> memberList;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    names = getResources().getStringArray(R.array.names);
    profile_pics = getResources().obtainTypedArray(R.array.profile_pics);
    emails = getResources().getStringArray(R.array.email);
memberList = new ArrayList<Member>();
    for (int i = 0; i < names.length; i++) {
      Member member = new Member(names[i], emails[i], profile_pics.getResourceId(i, -1));
      memberList.add(member);
    }
    recyclerview = (RecyclerView) findViewById(R.id.recyclerview);
    LinearLayoutManager layoutManager = new LinearLayoutManager(MainActivity.this);
    recyclerview.setLayoutManager(layoutManager);
    RVAdapter adapter = new RVAdapter(memberList, MainActivity.this);
    recyclerview.setAdapter(adapter);
  }
```

```
}
```

Output:

Android RecyclerView with Scrolling Behaviour :

Note:

- **You Need add dependencies in gradle.bulild**

```
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   compile 'com.android.support:appcompat-v7:23.0.1'
   compile 'com.android.support:design:23.0.1'
   compile 'com.android.support:recyclerview-v7:23.0.1'
}
```

file : tab_one_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/fragment_bg">
   <android.support.v7.widget.RecyclerView
     android:id="@+id/recyclerview"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     app:layout_behavior="@string/appbar_scrolling_view_behavior"/>
</RelativeLayout>
```

file : list_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:padding="16dp"
   android:layout_width="match_parent"
   android:layout_height="56dp">
```

```
   <TextView
      android:id="@+id/list_item"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
</LinearLayout>
```

file :RVAdapter

```
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.tutorialsbuzz.recyclerview.R;
import java.util.List;
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ItemViewHolder> {
   private List<String> mItems;
    public RVAdapter(List<String> mItems) {
      this.mItems = mItems;
   }

public class ItemViewHolder extends RecyclerView.ViewHolder {
      private TextView mTextView;
      public ItemViewHolder(View itemView) {
         super(itemView);
         mTextView = (TextView) itemView.findViewById(R.id.list_item);
      }
   }
 @Override
   public void onBindViewHolder(ItemViewHolder itemViewHolder, int i) {
      itemViewHolder.mTextView.setText(mItems.get(i));
   }
```

```
@Override
   public ItemViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
      View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list_row, viewGroup, false);
      ItemViewHolder itemViewHolder = new ItemViewHolder(view);
      return itemViewHolder;
   }

@Override
   public int getItemCount() {
      return mItems.size();
   }
}
```

file :TabOneFragment

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import java.util.ArrayList;

public class TabOneFragment extends Fragment {
   private RecyclerView recyclerview;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
   }

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
   View view =  inflater.inflate(R.layout.tab_one_fragment, container, false);
   recyclerview = (RecyclerView)view.findViewById(R.id.recyclerview);
   LinearLayoutManager layoutManager = new LinearLayoutManager(getActivity());
   recyclerview.setLayoutManager(layoutManager);
   return view;
}
@Override
public void onActivityCreated(Bundle savedInstanceState) {
   super.onActivityCreated(savedInstanceState);

   ArrayList<String> items = new ArrayList<String>();
   for (int i = 0; i < 50; i++) {
      items.add("TextView_"+i);
   }
```

```
            RVAdapter adapter = new RVAdapter(items);
        recyclerview.setAdapter(adapter);
    }
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    ArrayList<String> items = new ArrayList<String>();
    for (int i = 0; i < 50; i++) {
        items.add("TextView_"+i);
    }
    RVAdapter adapter = new RVAdapter(items);
    recyclerview.setAdapter(adapter);
}

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout

            xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:app="http://schemas.android.com/apk/res-auto"
             android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/fragment_bg">
    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"/>
</RelativeLayout>
```

file :

Create Same As Fragment Two And Three And XML Layout Fro Fragment Two And Three.

Android Material Design SnackBar :

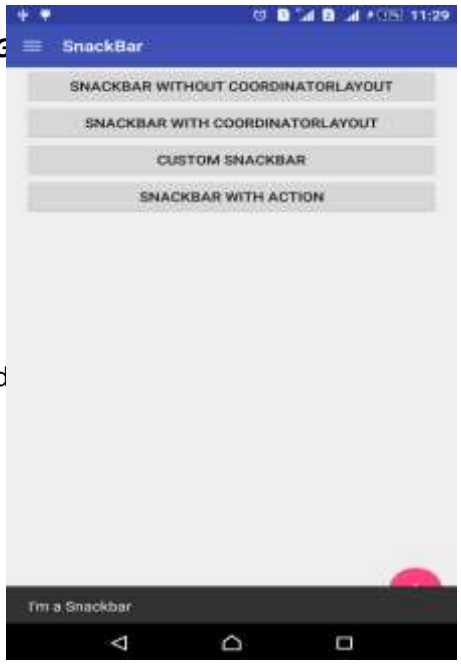
- Another Interesting Component being introduced in material design is SnackBar , SnackBar is just like that of Toast to present quick brief feedback to the user , expect they provide additional ability to interact with through actions and swiping to dismiss they are considerably more powerful than toasts, and the API is familiar.

Note:

- **You Need add dependencies in gradle.bulild**

```
 dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.android.support:design:23.0.1‘
```

```
}
Note:MainActivity.java
public class MainActivity extends AppCompatActivity {
   Toolbar toolbar;
   CoordinatorLayout coordinatorLayout;
   RelativeLayout parentLayout;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      toolbar = (Toolbar) findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);
      ActionBar actionBar = getSupportActionBar();
      actionBar.setHomeAsUpIndicator(R.drawable.ic_menu);
      actionBar.setDisplayHomeAsUpEnabled(true);
      parentLayout = (RelativeLayout) findViewById(R.id.parentLayout);
      coordinatorLayout = (CoordinatorLayout) findViewById(R.id.coordinator);
   }
public void snackbarWithoutCoordinate(View view) {
   Snackbar snackbar = Snackbar
         .make(parentLayout, "I'm a Snackbar", Snackbar.LENGTH_LONG);
   snackbar.show();
}
public void snackbarWithCoordinate(View view) {
   Snackbar snackbar = Snackbar
         .make(coordinatorLayout, "I'm a Snackbar", Snackbar.LENGTH_LONG);
   snackbar.show();
}
public void customSnackBar(View view) {
   Snackbar snackbar = Snackbar
         .make(coordinatorLayout, "I'm a Snackbar", Snackbar.LENGTH_LONG
         .setAction("Action", new View.OnClickListener() {
           @Override
           public void onClick(View view) {
           }
         });
snackbar.setActionTextColor(Color.RED);
      View sbView = snackbar.getView();
      TextView textView = (TextView) sbView.findViewById(android.support.d
      textView.setTextColor(Color.YELLOW);
      snackbar.show();
   }
   public void actionSnackBar(View view) {
      Snackbar snackbar=Snackbar.make(coordinatorLayout,
"I'm a Snackbar", Snackbar.LENGTH_LONG).setAction
```



```
("Action", new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          //perform on click of snackbar action
           Toast.makeText(MainActivity.this, "Snackbar Action",
Toast.LENGTH_LONG).show();
        }
      });
      snackbar.show();
   }
}
```

Note:**activity_main.xml**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/parentLayout"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <android.support.design.widget.CoordinatorLayout
      android:id="@+id/coordinator"
       android:layout_width="match_parent"
       android:layout_height="match_parent">
       <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent"
         android:orientation="vertical">
         <include layout="@layout/tool_bar" />
           <Button
                    style="@style/ButtonStyle"
                android:onClick="snackbarWithoutCoordinate"
                 android:text="@string/snack_bar_without_cordinate" />
            <Button
                 style="@style/ButtonStyle"
                android:onClick="snackbarWithCoordinate"
                android:text="@string/snack_bar_with_cordinate" />
            <Button
                style="@style/ButtonStyle"
                android:onClick="customSnackBar"
                android:text="@string/custom_snack_bar" />
            <Button
                 style="@style/ButtonStyle"
                 android:onClick="actionSnackBar"
                 android:text="@string/snack_bar_action" />

</LinearLayout>
<android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_gravity="bottom|right"
        android:layout_marginBottom="@dimen/activity_vertical_margin"
        android:layout_marginRight="@dimen/activity_horizontal_margin"
        android:src="@drawable/ic_action_new" />
  </android.support.design.widget.CoordinatorLayout>
</RelativeLayout>
```

Note:**toolbar.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/toolbar"
  android:layout_width="match_parent"
  android:layout_height="?attr/actionBarSize"
  android:background="?attr/colorPrimary"
  android:theme="@style/ThemeOverlay.AppCompat.Dark" />
```

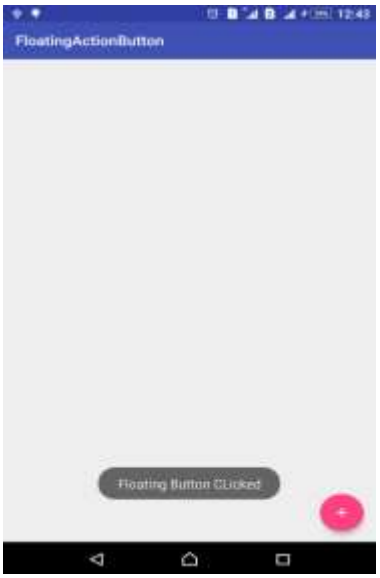
Android Material Design Floating Action Button :

- A floating action button is a round button denoting a primary action on your interface , FloatingActionButton is nothing but a view extending ImageViews. As the name suggest it floats on top of the user interface and used to display any promoted action, like adding a new item, compose mail, etc .

Note:

- **You Need add dependencies in gradle.bulild**

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  compile 'com.android.support:appcompat-v7:23.0.1'
  compile 'com.android.support:design:23.0.1‘
}
```



File:activitymain.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    app:titleTextAppearance="@style/AppTheme.Toolbar.Title" />
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="@dimen/activity_vertical_margin"
    android:layout_marginRight="@dimen/activity_horizontal_margin"
    android:src="@drawable/ic_add" />
</RelativeLayout>
```

File:style.xml

```
<resources>
  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="android:windowNoTitle">true</item>
         <item name="windowActionBar">false</item>
         <!-- Customize your theme here. -->
         <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
         <item name="colorAccent">@color/accent</item>
  </style>
  <style name="AppTheme.Toolbar.Title"
 parent="TextAppearance.Widget.AppCompat.Toolbar.Title">
     <item name="android:textColor">@color/white</item>
  </style>
</resources>
```

Android Spinner In Toolbar :

As the title of the post suggest in this tutorial we will see how to have spinner widget inside the toolbar in the previous series of tutorial we have seen many example on how to set up the **android spinner widget**and also we have seen how to have **android material design toolbar**

File:MainActivity.java

```
public class MainActivity extends AppCompatActivity {
  private Toolbar toolbar=null;
  private String[] category=null;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    category = getResources().getStringArray(R.array.category);
    toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setDisplayShowTitleEnabled(false);
    toolbar.setLogo(R.drawable.ic_drawer);
SpinnerAdapter spinnerAdapter = ArrayAdapter.createFromResource(getApplicationContext(),
R.array.category, R.layout.spinner_dropdown_item);
Spinner navigationSpinner = new Spinner(getSupportActionBar().getThemedContext());
navigationSpinner.setAdapter(spinnerAdapter);
```

```
toolbar.addView(navigationSpinner, 0);
navigationSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
   @Override
   public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
      Toast.makeText(MainActivity.this,
           "you selected: " + category[position],
           Toast.LENGTH_SHORT).show();
   }
@Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
      });

   }
}
```

File:activity_main.xml

```
 <LinearLayout xmlns:android="http://schemas.android.com
/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">
   <android.support.v7.widget.Toolbar
      android:id="@+id/toolbar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="?attr/colorPrimary"
      android:minHeight="?attr/actionBarSize" />
</LinearLayout>
```

File:spinner_dropdown_item.xml

```
 <?xml version="1.0" encoding="utf-8"?>
<TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@android:id/text1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@color/colorPrimary"
   android:gravity="center_vertical"
   android:minHeight="?android:attr/listPreferredItemHeightSmall"
   android:paddingLeft="12dp"
   android:paddingRight="12dp"
   android:textAppearance="?android:attr/textAppearanceListItemSmall" />
```

File:String.xml

```
 <resources>
   <string name="app_name">SpinnerToolbar</string>
```

```
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <!-- Category -->
    <string-array name="category">
        <item>Top News</item>
        <item>Business</item>
        <item>Politics</item>
        <item>Sports</item>
        <item>Technology</item>
        <item>Movies</item>
    </string-array>
</resources>
```

Android TextInputLayout :

EditText , has a hint attribute that will show text inside the EditText telling the user what to enter in this text field. That hint text disappears due to the user inputting text. However, TextInputLayout , "a layout wrapping an EditText", shows hint as floating label when the hint is hidden due to the user inputting text. TextInputLayout is part of *Design Support library*

Android Searchview On Toolbar

- There might be requirement in your application where you want to implement the search functionality by having searchView widget inside the toolbar for filter the Listview.

  Create Number of Files For this Example Follow Below List:

  1] Mainactivity.java (Already Created)
  2]ListviewAdapter.java (For CustomListview which contain Adapter)
  3]activity_main.xml (Already Created)
  4]itemlistview.xml (Create For Bind Controls on Listview)
  5]mainmenu.xml(Already Created,if not created then create in menu folder)

  Note:

- **You Need add dependencies in gradle.bulild**

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1‘
    compile 'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'
    compile 'com.android.support:design:24.0.1‘
    }
```

Put Listview in MainActivity LayoutFile

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context=".MainActivity"
   tools:showIn="@layout/activity_main">
   <ListView
      android:id="@+id/list_item"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:scrollbars="none"/>
</RelativeLayout>
```

Do Entry in menu folder in menu file.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context=".MainActivity">
   <item
      android:id="@+id/action_search"
      android:icon="@android:drawable/ic_menu_search"
      app:showAsAction="always"
      app:actionViewClass="android.support.v7.widget.SearchView"
      android:title="Search"/>
</menu>
```

Create One itemlistview.xml Layout file for Adapter class

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal"
   android:padding="10dp">
   <ImageView
      android:id="@+id/image_view"
      android:layout_width="0dp"
      android:layout_height="75dp"
      android:layout_weight="30"
      android:contentDescription="@string/app_name" />
         <TextView
      android:id="@+id/text"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:layout_weight="70"
      android:gravity="center"
      android:text=""
      android:textStyle="bold" />
</LinearLayout>
```

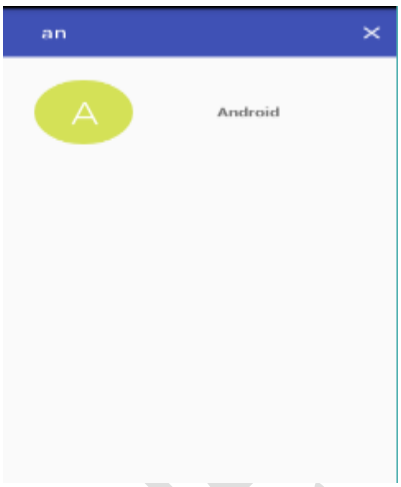
Create Adapter Class For Set data in Listview:

```
public class ListViewAdapter extends ArrayAdapter<String> {
   private MainActivity activity;
   private List<String> friendList;
   private List<String> searchList;
   public ListViewAdapter(MainActivity context, int resource, List<String> objects) {
      super(context, resource, objects);
      this.activity = context;
       this.friendList = objects;
       this.searchList = new ArrayList<>();
       this.searchList.addAll(friendList);
   }
   @Override
   public int getCount() {
      return friendList.size();
   }
@Override
   public String getItem(int position) {
      return friendList.get(position);
   }

   @Override
   public long getItemId(int position) {
      return position;
   }

   @Override
   public View getView(int position, View convertView, ViewGroup parent) {
      ViewHolder holder;
      LayoutInflater inflater = (LayoutInflater)
activity.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
      // If holder not exist then locate all view from UI file.
      if (convertView == null) {
         // inflate UI from XML file
         convertView = inflater.inflate(R.layout.item_listview, parent, false);
         // get all UI view
         holder = new ViewHolder(convertView);
         convertView.setTag(holder);
      } else {
         // if holder created, get tag from view
         holder = (ViewHolder) convertView.getTag();
      }
      holder.friendName.setText(getItem(position));
      //get first letter of each String item
      String firstLetter = String.valueOf(getItem(position).charAt(0));
      ColorGenerator generator = ColorGenerator.MATERIAL; // or use DEFAULT
      // generate random color
```

```
        int color = generator.getColor(getItem(position));
        TextDrawable drawable = TextDrawable.builder()
                .buildRound(firstLetter, color); // radius in px
        holder.imageView.setImageDrawable(drawable);
        return convertView;
    }
// Filter Class
    public void filter(String charText) {
        charText = charText.toLowerCase(Locale.getDefault());
        friendList.clear();
        if (charText.length() == 0) {
            friendList.addAll(searchList);
        } else {
            for (String s : searchList) {
                if (s.toLowerCase(Locale.getDefault()).contains(charText)) {
                    friendList.add(s);
                } }}
        notifyDataSetChanged();
    }
    private class ViewHolder {
        private ImageView imageView;
        private TextView friendName;
        public ViewHolder(View v) {
            imageView = (ImageView) v.findViewById(R.id.image_view);
            friendName = (TextView) v.findViewById(R.id.text);
        } }}
```

Do Entry in MainActivity.Java File:

```
public class MainActivity extends AppCompatActivity {
    private ListView listView;
    private ArrayList<String> stringArrayList;
    private ListViewAdapter adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listView = (ListView) findViewById(R.id.list_item);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        setData();
       adapter = new ListViewAdapter(this, R.layout.item_listview, stringArrayList);
       listView.setAdapter(adapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(MainActivity.this, (String)parent.getItemAtPosition(position),
```




```
Toast.LENGTH_SHORT).show();
   }
});
}
private void setData() {
   stringArrayList = new ArrayList<>();
   stringArrayList.add("Android");
   stringArrayList.add("iOS");
   stringArrayList.add("DOTNET");
   stringArrayList.add("PHP");
   stringArrayList.add("XZMARIN");
   stringArrayList.add("PHONEGAP");
   stringArrayList.add("UNITY");
   stringArrayList.add("CODE IGNITOR");
   stringArrayList.add("JAVA");
   stringArrayList.add("PYTHON");
   stringArrayList.add("RBUY&RAILS");
   stringArrayList.add("JSP");
   stringArrayList.add("HIBERNET");
   stringArrayList.add("MATERIAL DESIGN");
}
@Override
   public boolean onCreateOptionsMenu( Menu menu) {
      getMenuInflater().inflate( R.menu.menu_main, menu);

      MenuItem myActionMenuItem = menu.findItem( R.id.action_search);
      final SearchView searchView = (SearchView) myActionMenuItem.getActionView();
      searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
         @Override
         public boolean onQueryTextSubmit(String query) {
            return false;}
         @Override
         public boolean onQueryTextChange(String newText) {
            if (TextUtils.isEmpty(newText)) {
               adapter.filter("");
               listView.clearTextFilter();
            } else {
               adapter.filter(newText);   }
            return true;
         }      });       return true;
   }}
```

Butter knife and Android Annotation :

Butterknife installation :

Add dependencies into the build.gradle

compile **'com.jakewharton:butterknife:8.0.1'**

We are using butterknife 8.+ version so we will use

@BindView and ButterKnife.bind(*)

If we have Butterknife 7 or below

Then we have to use

@InjectView and ButterKnife.inject(*)

Activity with butterknife :

```
package com.example.tops.sqlitedatabase;

import ...

public class MainActivity extends AppCompatActivity {

    @BindView(R.id.edname)
    EditText ed1;

    @BindView(R.id.btninsert)
    Button btn;

    ArrayList<Pojo> list;

    @BindView(R.id.lv)
    ListView lv;

    DbHelp db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ButterKnife.bind(this);

        db = new DbHelp(MainActivity.this);

        list = db.getdata();

        Custad ad = new Custad(MainActivity.this,list);
        lv.setAdapter(ad);
```

Event with butterknife :

```
}
@OnClick(R.id.btninsert)
void buttonclick(){

    String sname = edl.getText().toString();

    if (sname.length()==0){

        edl.setError("Fill this field");
        list = db.getdata();

        Custad ad = new Custad(MainActivity.this,list);
        lv.setAdapter(ad);
    }
    else {

        Pojo p = new Pojo();
        p.setName(sname);
        String s = db.insertdata(p);
        Toast.makeText(MainActivity.this, s, Toast.LENGTH_LONG).show();
        edl.setText("");

        list = db.getdata();

        Custad ad = new Custad(MainActivity.this, list);
        lv.setAdapter(ad);
    }
```

```
@OnItemClick
void listitemselect(int position){

    final Pojo p = (Pojo)list.get(position);
    Toast.makeText(MainActivity.this,p.getName()+"  "+p.getId(),Toast.LENGTH_LONG).show();
    new AlertDialog.Builder(MainActivity.this).setTitle("Choose Options").setMessage("What do you want to do with "+p

            String s = db.deletedata(p.getId());

            Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();

            list = db.getdata();

            Custad ad = new Custad(MainActivity.this,list);
            lv.setAdapter(ad);
    }).setNegativeButton("update", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {


            Intent i = new Intent(MainActivity.this,UpdateData.class);

            i.putExtra("id",p.getId());
            i.putExtra("name",p.getName());

            startActivity(i);
```

Custom baseadapter with butterknife :

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View view = inflater.inflate(R.layout.singlelist,parent,false);

   ViewHold v = new ViewHold(view);


    Pojo p = (Pojo) list.get(position);

    v.tid .setText(p.getId());
    v.tname.setText(p.getName());


    return view;
}

class ViewHold{

    @BindView(R.id.txtid)
    TextView tid;

    @BindView(R.id.txtname)
    TextView tname;

    ViewHold(View view){

        ButterKnife.bind(this,view);

    }
```

Fragments with butterknife :

```
public class FragDemo extends Fragment {

    @BindView(R.id.edname)
    EditText edl;

    @BindView(R.id.btninsert)
    Button btn;



    @BindView(R.id.lv)
    ListView lv;


    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.activity_main,container,false);

        ButterKnife.bind(this,view);

        return view;
    }
}
```

PERSISTENCE IN ANDROID

The data-persistence techniques in Android provide options for balancing speed, efficiency, and robustness.

- **Shared Preferences :**

When storing UI state, user preferences, or application settings, you want a lightweight mechanism to store a known set of values. Shared Preferences let you save groups of key/value pairs of primitive data as named preferences.

- **Saved Application State :**

Activities include specialized event handlers to record the current UI state when your application is moved to the background.

- **Files**

It's not pretty, but sometimes writing to and reading from files is the only way to go. Android lets you create and load files on the device's internal or external media.
Using the Shared Preferencesclass you can create named maps of key/value pairs within your application that can be shared among application components running in the same application context.
Shared Preferences support the primitive types Boolean, string, float, long, and integer, making them an ideal means of quickly storing default values, class instance variables, the current UI state, and user preferences. They are most commonly used to persist data across user sessions and to share settings among application components.

Creating and Saving Preferences

To create or modify a Shared Preference, call getSharedPreferences on the application Context, passing in the name of the Shared Preference to change. Shared Preferences are shared across an application's components, but aren't available to other applications.

To modify a Shared Preference use the SharedPreferences.Editor class. Get the Editor object by calling edit on the Shared Preferences object you want to change. To save edits call commit on the Editor, as shown in fig.

Simple Example of Create, Save and Retrieve Shared Preferences

- In this example we will set preferences in one activity MainPreferencesActivity.java. Retrieve these values in the next activity – ViewSharedPrefs.java. The second activity displays my preferred preferences in a Text view.
- Create new Project first decide your UI. So in that res/layout/**main.xml.** in that we add only one button which is load Preferences.

Inactivity_main.xml file.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:hint="Username" >
      </EditText>
      <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
```

```
        android:layout_below="@+id/editText1"
        android:ems="10"
        android:hint="Password" />
<EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/editText2"
        android:ems="10"
        android:hint="Full Name" />
<CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/editText3"
        android:text="Agree Terms and Condition" />
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/checkBox1"
        android:text="Registration" />
</RelativeLayout>
```

InMainPreferencesActivity.java file.

```
public class SahredPrefrencesDemo extends Activity {
      private final String NAME = "name";
      private final String USER = "user";
      private final String PASS = "pass";
      private final String IS_REGISTER = "isREG";
      private Button btn;
      private EditText txt, txt2, txt3;
      private CheckBox chk;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_sahred_prefrences_demo);
            btn = (Button) findViewById(R.id.button1);
            txt = (EditText) findViewById(R.id.editText1);
            txt2 = (EditText) findViewById(R.id.editText2);
            txt3 = (EditText) findViewById(R.id.editText3);
            chk = (CheckBox) findViewById(R.id.checkBox1);
            btn.setOnClickListener(new OnClickListener() {
      public void onClick(View arg0) {
      // TODO Auto-generated method stub
      if (chk.isChecked()) {
      setPrefrance();
      Intent intent = new Intent(SahredPrefrencesDemo.this,
      ShowSharedData.class);
```

```
      startActivity(intent);
                        }
                  }
            });
      }
privatevoid setPrefrance()
{
      SharedPreferences sp = this.getSharedPreferences("MyPref",
                                    MODE_WORLD_READABLE);
      SharedPreferences.Editor spEditor = sp.edit();
      spEditor.putBoolean(IS_REGISTER, chk.isChecked());
      spEditor.putString(USER, txt.getText().toString());
      spEditor.putString(PASS, txt2.getText().toString());
      spEditor.putString(NAME, txt3.getText().toString());
      spEditor.commit();
      }
}
```

RETRIEVING SHARED PREFERENCES

Accessing Shared Preferences, like editing and saving them, is done using the getSharedPreferences method. Pass in the name of the Shared Preference you want to access, and use the type-safe get<type>methods to extract saved values.

Simple Example of Retrive Shared Preference

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<TextView
        android:id="@+id/txtHead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="Registration Data"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/txtHead"
        android:text="TextView" />
<TextView
        android:id="@+id/text3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/text2"
        android:text="TextView" />
<TextView
        android:id="@+id/text4"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/text3"
        android:text="TextView" />
<TextView
        android:id="@+id/text5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/text4"
        android:text="TextView" />
</RelativeLayout>
```

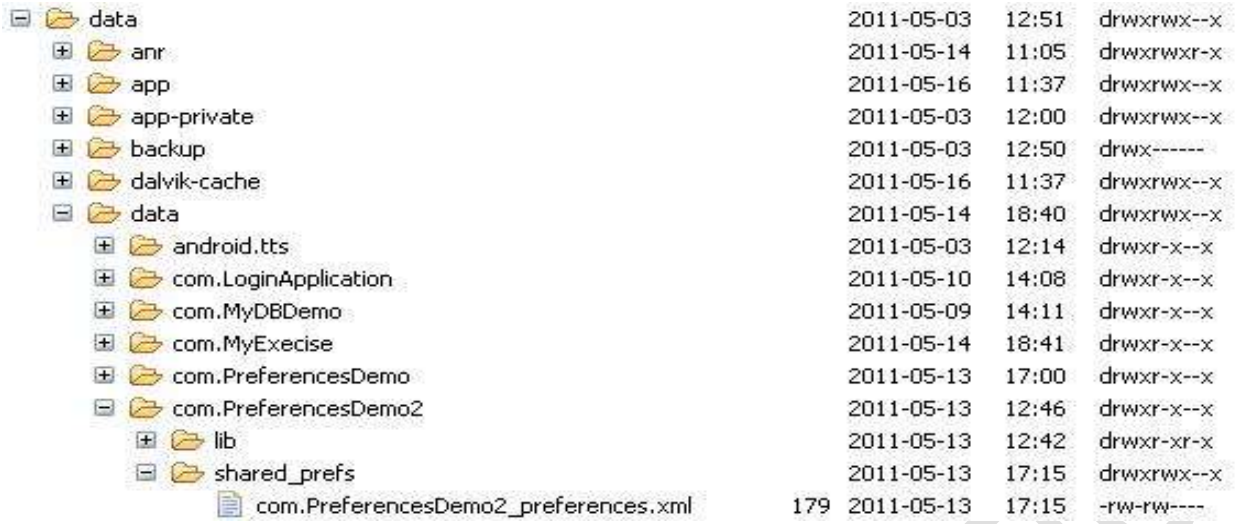
Create new activity LoadPreference.java but first create new layout which is use to set your Preference Values. In res/layout create new file name new_layout.xml. in that we silmple add five text view.

Now in LoadPreference.java file.

```
public class ShowSharedData extends Activity {
      private final String NAME = "name";
      private final String USER = "user";
      private final String PASS = "pass";
      private final String IS_REGISTER = "isREG";
      private TextView txt1, txt3, txt4, txt2;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_show_shared_data);
            txt1 = (TextView) findViewById(R.id.text2);
            txt2 = (TextView) findViewById(R.id.text3);
            txt3 = (TextView) findViewById(R.id.text4);
            txt4 = (TextView) findViewById(R.id.text5);
            getPrefrenceData();
      }
      private void getPrefrenceData() {
//Read Shared Preference Data
            SharedPreferences spf = this.getSharedPreferences("MyPref",
                        MODE_WORLD_READABLE);
            txt1.setText("UserName : " + spf.getString(USER, ""));
            txt2.setText("Password : " + spf.getString(PASS, ""));
            txt3.setText("Full Name : " + spf.getString(NAME, ""));
            txt4.setText("Is Registered : " + spf.getBoolean(IS_REGISTER,
false));
      }
}
```

Path to an application's saved preferences

The Android framework also takes care of persisting preferences. For example, when the user clicks Button, Android stores the selection in an XML file within the application's /data directory



The actual file path is /data/data/[PACKAGE_NAME]/shared_prefs/[PACKAGE_NAME]_PreferencesDemo2_preferences.xml

Preference Activity

Android offers an XML-driven framework to create system-style preference screens foryour applications. By using this framework you can ensure that the preference Activitiesin your applications are consistent with those used in both native and other third-partyapplications.

This has two distinct advantages:

- Users will be familiar with the layout and use of your application settings screen.
- You can integrate settings screens from other applications (including system settings such as location settings) into your application's settings screens.

The Preference Activity framework consists of three parts:

- **Preference Screen Layout**

  An XML file that defines the hierarchy displayed in your Preference Activity. It specifies the controls to display, the values to allow, and the Shared Preference keys to use for each UI control.

- **Preference Activity**

  An extension of Preference Activitythat will be used to host your application preference screens.

- **Shared Preference Change Listener**

  An implementation of the onSharedPreferenceChangeListener class used to listen for changes to Shared Preferences.

The Activity Preference framework is a powerful tool for creating fully customizable dynamic preference screens. The full range of possibilities available through this framework is beyond the scope of this book; however, the following sections will introduce it and demonstrate how to create and use each of the components described above.

Defining a Preference Screen Layout in XML

The most important part of the Preference Activity is the XML layout. Unlike in the standard UI layout, preference definitions are stored in the res/xml resources folder.

While conceptually they are similar to the UI layout resources, Preference Screen layouts use a specialized set of controls designed specifically to create preference screens like those used for system settings.

Each preference layout is defined as a hierarchy, beginning with a single Preference Screenelement

Within each Preference Screen you can include any combination of PreferenceCategory and Preference<control> elements. Preference Category elements, shown in the followingsnippet, are used to break each Preference Screen into subcategories using a title bar separator:

While the specific attributes available for each preference control vary, each of them includes at least the following four:

- **android:key** : The Shared Preference key the selected value will be recorded against.
- **android:title :** The text displayed to represent the preference.
- **android:summary** : The longer text description displayed in a smaller font below the title text.
- **android:defaultValue:** The default value that will be displayed (and selected) if no preference value has been assigned to this preference key.

Native Preference Controls

Android includes several preference controls to build yourPreference Screens:

Checkbox Preference:A standard preference checkbox control. Used to set preferences to true or false.

Edit Text Preference: Allows users to enter a string value as a preference. Selecting the preference text will display a text entry dialog.

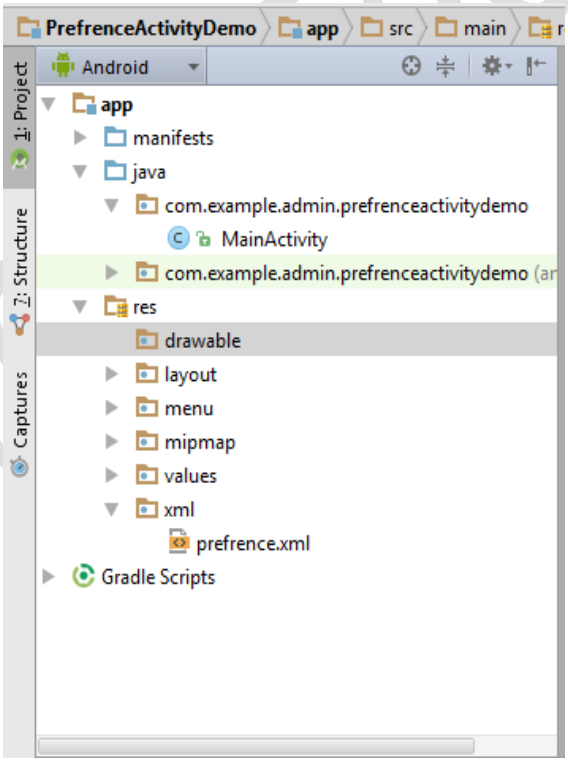
List Preference: The preference equivalent of a spinner. Selecting this preference will display a dialog box containing a list of values from which to select. You can specify different arrays to contain the display text and selection values.

Ringtone Preference:A specialized List Preference that presents the list of available ringtones for user selection. This is particularly useful when you're constructing a screen to configure notification settings.

Each of these preference controls can be used to construct your Preference Screen hierarchy. Alternatively, you can create your own specialized preference controls by extending the Preference class (or any of these subclasses).

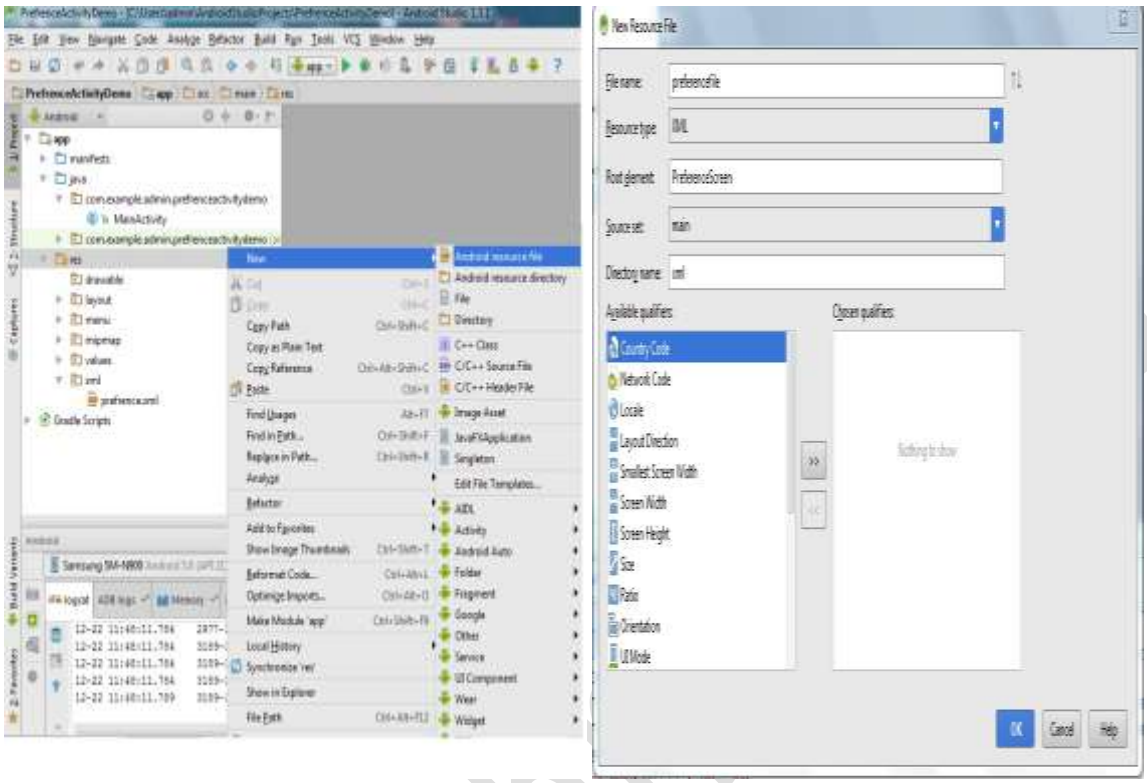
Simple Shared Preferences example using xml file

- Create new project and create folder in res/xml.

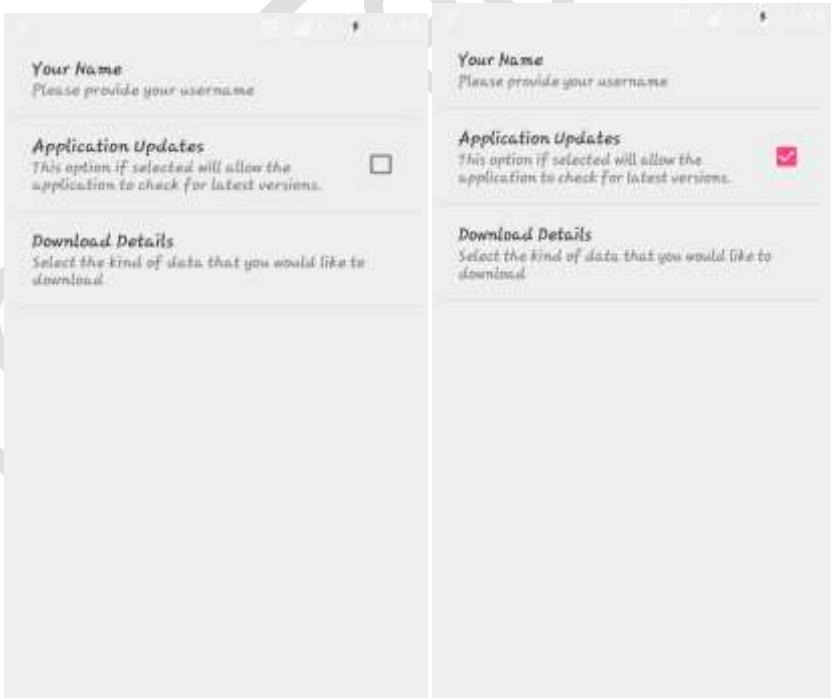


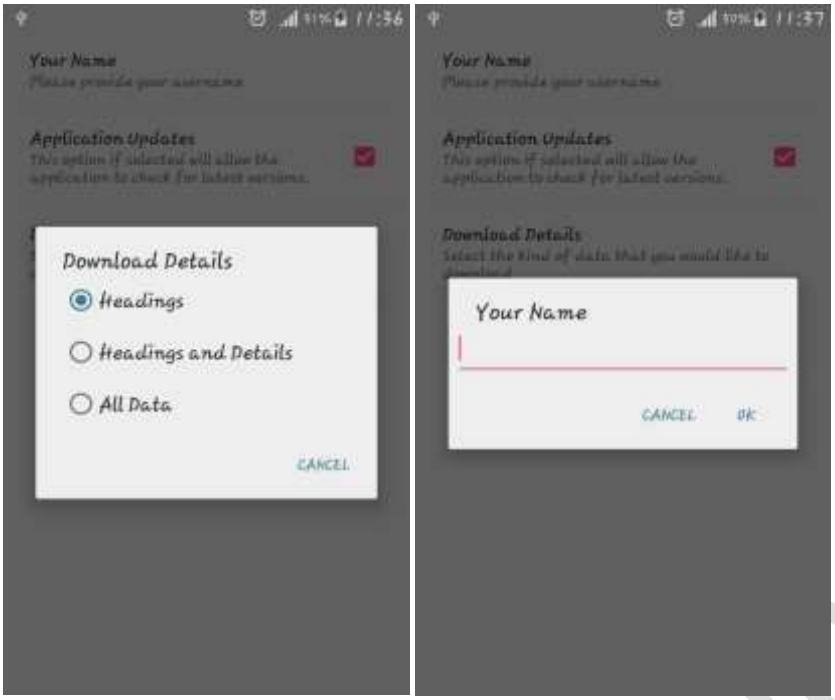
- Create new file **Android xml file.**

- Create new xml file, select res->new-> Android Resource File and select resource type as xml :



Preference Layout





In new_pref.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
<EditTextPreference
        android:key="username"
        android:summary="Please provide your username"
        android:title="Your Name"></EditTextPreference>

<CheckBoxPreference
        android:defaultValue="false"
        android:key="applicationUpdates"
        android:summary="This option if selected will allow the
application to check for      latest versions."
        android:title="Application Updates" />

<ListPreference
        android:defaultValue="1"
        android:entries="@array/listArray"
        android:entryValues="@array/listValues"
        android:key="downloadType"
        android:summary="Select the kind of data that you would like to
download"
        android:title="Download Details" />
</PreferenceScreen>
```

In mainActivity.java file

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      getFragmentManager().beginTransaction().replace(android.R.id.content,      new
MyPreferenceFragment()).commit();
   }
   public static class MyPreferenceFragment extends PreferenceFragment {
      @Override
      public void onCreate(final Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         addPreferencesFromResource(R.xml.prefrence);
 }
   }
}
```

Android Requesting Permission at run time(Android 6.0)

Android Marshmallow Permissions Example. One of the major changes in Android Marshmallow is the new permission system. In earlier versions we were declaring the permission in the AndroidManifest.xml file. But with Android Marshmallow we need to ask the permission at run time.

1. In AndroidMenifest file :

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

2. activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="net.simplifiedcoding.runtimepermission.MainActivity">


   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Request Storage Permission"
      android:id="@+id/buttonRequestPermission"
```

```
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />
</RelativeLayout>
```

3. In MainActivity.xml

```
public class MainActivity extends AppCompatActivity {
private Button btnRequestPermission;
//Permision code that will be checked in the method onRequestPermissionsResult
private int STORAGE_PERMISSION_CODE = 23;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btnRequestPermission = (Button) findViewById(R.id.buttonRequestPermission);
btnRequestPermission.setOnClickListener(new View.OnClickListener() {
@Override
        public void onClick(View v) {
        //First checking if the app is already having the permission
        if(isReadStorageAllowed()){
                //If permission is already having then showing the toast
                Toast.makeText(MainActivity.this,"You already have
        permission",Toast.LENGTH_LONG).show();
                //Existing the method with return
                return;
        }
        //If the app has not the permission then asking for the permission
        requestStoragePermission();
        }
        });
}
//We are calling this method to check the permission status
private boolean isReadStorageAllowed() {
        //Getting the permission status
        int result = ContextCompat.checkSelfPermission(this,
        Manifest.permission.READ_EXTERNAL_STORAGE);
        //If permission is granted returning true
        if (result == PackageManager.PERMISSION_GRANTED)
                return true;
                //If permission is not granted returning false
        return false;
}
//Requesting permission
private void requestStoragePermission(){
        if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.permission.READ_EXTER
NAL_STORAGE)){
```

```
        }
        ActivityCompat.requestPermissions(this,new
        String[]{Manifest.permission.READ_EXTERNAL_STORAGE},STORAGE_PERMISSION_CODE);
}
//This method will be called when the user will tap on allow or deny
@Override
        public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

        //Checking the request code of our request
        if(requestCode == STORAGE_PERMISSION_CODE){
        //If permission is granted
        if(grantResults.length >0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                //Displaying a toast
        Toast.makeText(this,"Permission granted now you can read the
        storage",Toast.LENGTH_LONG).show();
        }else{
                //Displaying another toast if permission is not granted
Toast.makeText(this,"Oops you just denied the permission",Toast.LENGTH_LONG).show();
                }
        }     } }
```

Using the filesystem

Android's file system is based on Linux and supports mode-based permissions. You canaccess this file system in several ways. You can create and read files from within applications, you can access raw resource files, and you can work with specially compiled custom XML files. In this section, we'll explore each approach.

Creating files

You can easily create files in Android and store them in your application's data path. The following listing demonstrates how to open a FileOutputStream and use it to create a file.

Simple example of files in android.

In your activity layout.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
  android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"  tools:context=".MainActivity">
</RelativeLayout>
```

- **In your activity java file**

```
public class MainActivity extends AppCompatActivity {
    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FragmentTransaction ft =getFragmentManager().beginTransaction();
        SDCardFragment fragment =new SDCardFragment();
        ft.replace(android.R.id.content,fragment);
        ft.commit();
    }
}
```

In Fragment.xml.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.admin.sdcarddemo.SDCardFragment">

<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="66dp"
        android:text="Enter Text to Write" />
<EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:inputType="textNoSuggestions" />
<Button
        android:id="@+id/btRead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/editText1"
        android:layout_below="@+id/editText1"
        android:text="Read" />
```

```
<TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignRight="@+id/textView1"
      android:layout_below="@+id/btRead"
      android:layout_marginTop="126dp"
      android:text="Click Read"
      android:textAppearance="?android:attr/textAppearanceLarge" />
<Button
      android:id="@+id/btWrite"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editText1"
      android:layout_below="@+id/editText1"
      android:text="Write" />

</RelativeLayout>
```

In Android Manifest

Add Uses Permission to Manifest file

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

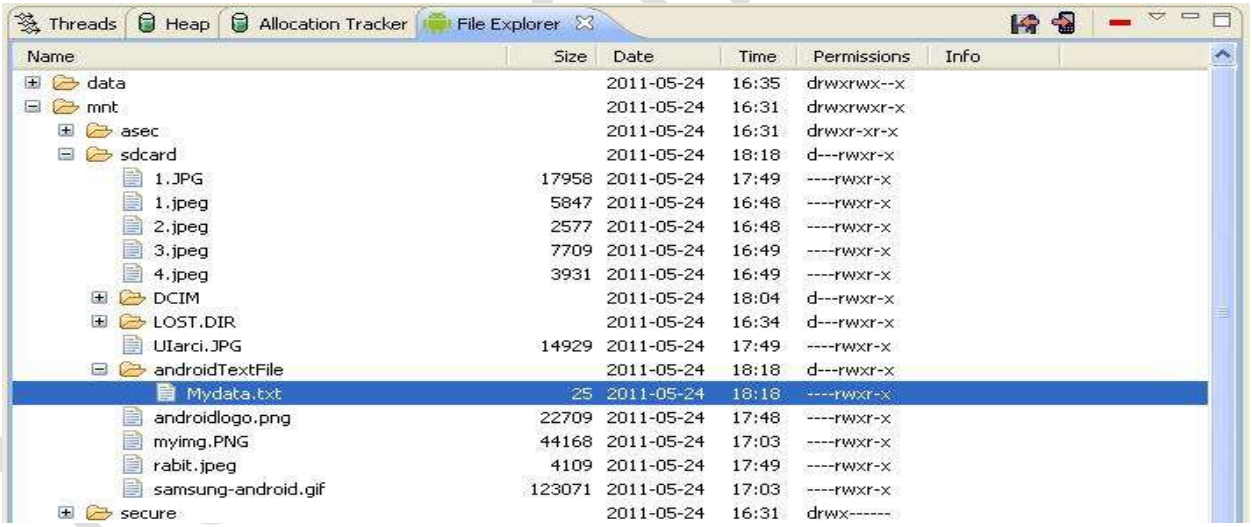
- Creating a file in Android from **Your first Activity** in **fileDemoOneMain.java**

```
public class SDCardFragment extends Fragment {
private Button btnWrite;
private Button btnRead;
private EditText txtWrite;
private TextView tvRead;
public SDCardFragment() {
// Required empty public constructor
}
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view =inflater.inflate(R.layout.fragment_sdcard, container, false);
btnWrite = (Button) view.findViewById(R.id.btWrite);
btnRead = (Button) view.findViewById(R.id.btRead);
txtWrite = (EditText) view.findViewById(R.id.editText1);
tvRead = (TextView) view.findViewById(R.id.textView2);
    btnWrite.setOnClickListener(new View.OnClickListener() {
public void onClick(View arg0) {
// Get External Storage
File SDCard = Environment.getExternalStorageDirectory();
      Toast.makeText(getActivity(), "Get SD Card" ,Toast.LENGTH_LONG).show();
if(SDCard.exists()) {
//Get Folder
File MyDir = new File(SDCard.getAbsolutePath() + "/MyDir");
        MyDir.mkdir();
```

```
        Toast.makeText(getActivity(), "Directory Created",
            Toast.LENGTH_LONG).show();
if (MyDir.exists()) {
            File MyFile = new File(MyDir.getAbsolutePath()+ "/MyFile.txt");
            Toast.makeText(getActivity(), "File Careated",
                Toast.LENGTH_LONG).show();
try {
//Create File
MyFile.createNewFile();
              FileOutputStream fos = null;
             if (MyFile.exists() && MyFile.canWrite()) {
try {
//Write File
fos = new FileOutputStream(MyFile);
                    String str = txtWrite.getText().toString();
                    fos.write(str.getBytes());
                    Toast.makeText(getActivity(),"Writing Complete",
                        Toast.LENGTH_LONG).show();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                    finally {
if (fos != null) {
try {
                        fos.flush();
                        fos.close();
                      } catch (IOException e) {
                        e.printStackTrace() }}}
              } else {
tvRead.setText("Unable to Write");
              }
            } catch (IOException e) {
             e.printStackTrace();}
        } else {
tvRead.setText("directory not available");}
      } else {
tvRead.setText("Unable to Write in directory");
      }   }});
btnRead.setOnClickListener(new View.OnClickListener() {
public void onClick(View arg0) {
            File SDCard = Environment.getExternalStorageDirectory();
            File dir = new File(SDCard.getAbsoluteFile()+ "/MyDir/MyFile.txt");
            FileInputStream fis = null;
if (dir.exists()) {
```

```
try {
                fis = new FileInputStream(dir);
byte[] reader = new byte[fis.available()];
while (fis.read(reader) != -1) {
tvRead.setText("Data From File: "+ new String(reader));
                }
              } catch (Exception e) {
                 e.printStackTrace();
              } finally {
try {
if (fis != null) {
                      fis.close();
                    }
                 } catch (IOException e) {
                    e.printStackTrace();
                 } }
            } else {
tvRead.setText("Unable to Read");
            }
          }
      }); return view}}
```

IN SDCARD FILE STRUCTURE:



DATABASES WITH SQLITE

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte) which makes it a good candidate from being embedded into other runtimes.

- SQLite is embedded into every Android device. Using a SQLite database in Android does not require a setup procedure or administration of the database.
- You only have to define the SQL statements for creating and updating the database. Afterwards the database is automatically managed for you by the Android platform.

- Access to a SQLite database involves accessing the file system. This can be slow. Therefore it is recommended to perform database operations asynchronously.
- If your application creates a database, this database is by default saved in the directoryDATA/data/APP_NAME/databases/FILENAME.
- *Slate*is a well regarded relational database management system (RDBMS). It is:
    - Open-source
    - Standards-compliant
    - Lightweight
    - Single-tier
- It has been implemented as a compact C library that's included as part of the Android software stack.
- SQLite is an embedded relational database engine. Its developers call it a self-contained, server less, zero-configuration and transactional SQL database engine.
- SQLite is created in C programming language and has bindings for many languages like C++, Java, C#, Python, Perl, Ruby, Visual Basic, Tcl and others. The source code of SQLite is in public domain.

SQLite Data Types

Most SQL database engines use static typing

The data type value is determined by the column it is stored in

SQLITE uses dynamic typing

The data type of a value is associated with the value itself

In order to maximize compatibility between SQLite and other database engines, SQLite supports "type affinity" of columns

- **NULL**
- **INTEGER**
    - 1,2,3,4,6, or 8 bytes, depending on magnitude
- **REAL**
    - 8 byte floating point value
- **TEXT**
- **BLOB**
    - Binary data such as images

Cursor

- ContentValues are used to insert new rows into tables. Each Content Values object represents a single table row as a map of column names to values.
- Queries in Android are returned as Cursor objects. Rather than extracting and returning a copy of the result values, Cursors are pointers to the result set within the underlying data.
- The Cursor class includes a number of navigation functions including, but not limited to, the following:
    - **moveToFirst** Moves the cursor to the first row in the query result
    - **moveToNext** Moves the cursor to the next row
    - **moveToPrevious** Moves the cursor to the previous row
    - **getCount** Returns the number of rows in the result set
    - **getColumnIndexOrThrow** Returns the index for the column with the specified name
    - **getColumnName** Returns the name of the specified column index
    - **getColumnNames** Returns a string array of all the column names in the current Cursor

- **moveToPosition** Moves the Cursor to the specified row
- **getPosition** Returns the current Cursor position

SQLiteOpenHelper

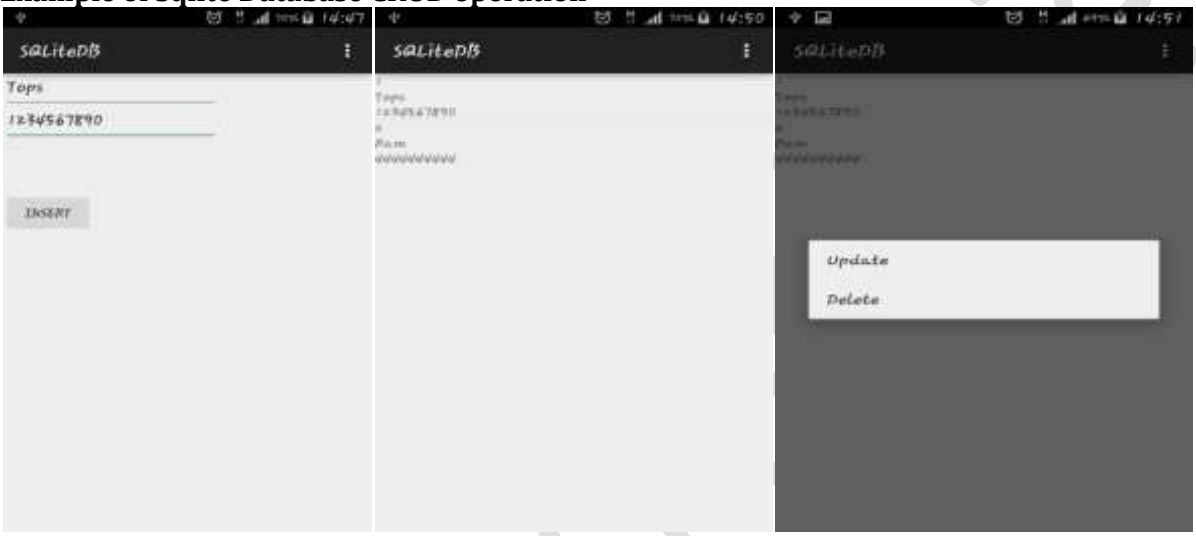
SQLiteOpenHelper is an abstract class used to implement the best practice pattern for creating, opening, and upgrading databases. By implementing a SQLite Open Helper you hide the logic used to decide if a database needs to be created or upgraded before it's opened.

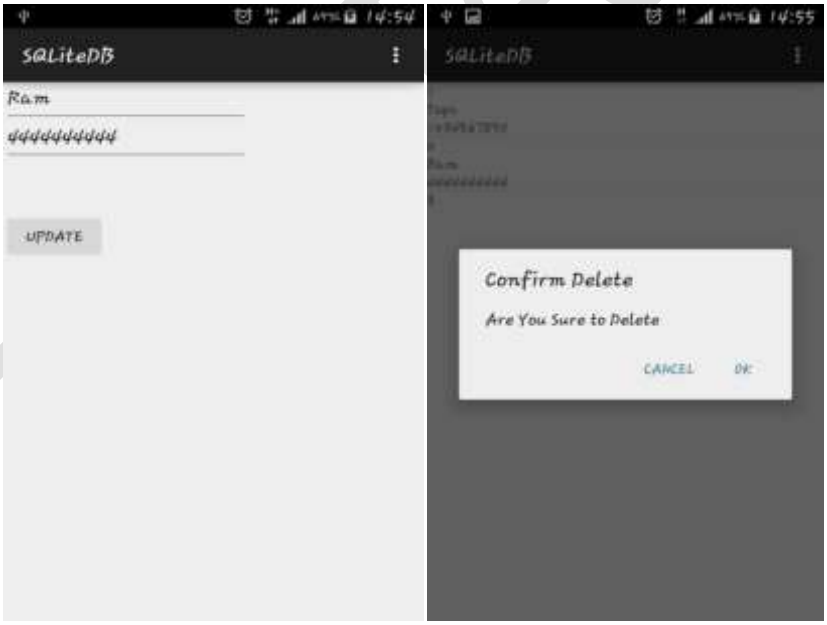
Create SqliteOpenHelper Class

To use an implementation of the helper class, create a new instance, passing in the context, database name, and current version, and a CursorFactory (if you're using one).

Call getReadableDatabase or getWritableDatabase to open and return a readable/writable instance of the underlying database.

Example of Sqlite Database CRUD operation





Contact.java

```
public class Contact {
	int _id;
	String _Name;
	String _phNo;
```

```
	public Contact(int id, String name, String phNo) {
		_id = id;
		_Name = name;
		_phNo = phNo;
	}
	public Contact(String name, String phNo) {
		_Name = name;
		_phNo = phNo;
	}
	public Contact() {
	}
	public int getID() {
		return this._id;
	}
	public void setID(int id) {
		this._id = id;
	}
	public String getName() {
		return this._Name;
	}
	public void setName(String name) {
		this._Name = name;
	}
	public String getPhNo() {
		return this._phNo;
	}
	public void setPhNo(String phNo) {
		this._phNo = phNo;
	}
}
```

DatabaseHandler.java

```
package com.example.arpandb;
public class DatabaseHandler extends SQLiteOpenHelper {
	private static final int DB_VERSION = 1;
	private static final String DB_NAME = "ContactManager";
	private static final String TABLE_NAME = "Contacts";
	private static final String KEY_ID = "id";
	private static final String KEY_NAME = "name";
	private static final String KEY_PH_NO = "phNo";
	public DatabaseHandler(Context context) {
		super(context, DB_NAME, null, DB_VERSION);
		// TODO Auto-generated constructor stub
	}
	public void onCreate(SQLiteDatabase db) {
		// TODO Auto-generated method stub
		String createContact = "Create Table " + TABLE_NAME + " (" + KEY_ID
				+ " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT," + KEY_PH_NO
				+ " TEXT)";
		db.execSQL(createContact);
```

```
        }
        @Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   String sql=("drop table if exist "+TABLE_NAME);
   db.execSQL(sql);
   onCreate(db);
}
public void insert(Contact contact){
   SQLiteDatabase db= getWritableDatabase();
   ContentValues values= new ContentValues();
   values.put(KEY_NAME, contact.get_Name());
   values.put(KEY_PH_NO, contact.get_phNo());

   db.insert(TABLE_NAME, KEY_ID, values);
}
public List<Contact> show(){
   ArrayList<Contact> contactlist= new ArrayList<Contact>();
   SQLiteDatabase db=getReadableDatabase();

   String[] columns = {KEY_ID,KEY_NAME,KEY_PH_NO};
   Cursor c=db.query(TABLE_NAME, columns, null, null, null, null,null);
while (c.moveToNext()) {
int id=c.getInt(0);
      String name=c.getString(1);
      String num=c.getString(2);

Contact u= new Contact();
         u.set_id(id);
         u.set_Name(name);
     u.set_phNo(num);
   contactlist.add(u);
   }
return contactlist;
}
void update(Contact contact)
{
   SQLiteDatabase db= getReadableDatabase();
   ContentValues values= new  ContentValues();
   values.put(KEY_NAME,contact.get_Name());
   values.put(KEY_PH_NO, contact.get_phNo());
   String whereClause=KEY_ID+" = "+contact.get_id();
   db.update(TABLE_NAME, values, whereClause, null);
}
public void delete(int id)
   {
```

```
    SQLiteDatabase db= getWritableDatabase();
    String whereClause=KEY_ID+" = "+id;
    db.delete(TABLE_NAME, whereClause, null);
  }
}
```

activity_data_demo.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ListView android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/lstContact">
</ListView>
</RelativeLayout>
```

- **In Main Activity java file**

```
public class MainActivity extends AppCompatActivity {

  @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    InsertData data= new InsertData();
    FragmentTransaction ft=getFragmentManager().beginTransaction();
    ft.replace(android.R.id.content, data);
    ft.commit();
  }
}
```

- **In Main Activity xml file**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
  android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
</RelativeLayout>
```

- **Create Contact in InsertData.java Fragment**

```
public class InsertData extends Fragment {
  EditText name,num;
  Button insert;
  FragmentTransaction ft;
  @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    View view=inflater.inflate(R.layout.fragment_insert,container,false);
```

```
name=(EditText)view.findViewById(R.id.edit_name);
num=(EditText)view.findViewById(R.id.edit_num);
insert=(Button)view.findViewById(R.id.insert_btn);
insert.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            String name_data = name.getText().toString();
            String num_data = num.getText().toString();

            Contact contact = new Contact();
            contact.set_Name(name_data);
            contact.set_phNo(num_data);
DBHandler db = new DBHandler(getActivity());
        db.insert(contact);

            ShowDataFragment sat = new ShowDataFragment();
ft = getFragmentManager().beginTransaction();
ft.replace(android.R.id.content, sat);
ft.commit();
        }      });
return view;}}
```

In Fragment Show Data xml file:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.example.admin.sqlitedb.ShowDataFragment">
<ListView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/listView"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />
</RelativeLayout>
```

In Fragment Show Data java file:

```
public class ShowDataFragment extends Fragment {
   ListView lv;
   List<Contact>list;
   DBHandler dbHandler;
   ArrayList<HashMap<String,String>>hmlist = new ArrayList<>();
```

```
public ShowDataFragment() {
// Required empty public constructor
}
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view= inflater.inflate(R.layout.fragment_show_data, container, false);;
lv=(ListView)view.findViewById(R.id.listView);
      refresh();
return view;
}
void refresh() {
dbHandler = new DBHandler(getActivity());
list=dbHandler.show();
for(Contact contact:list)
   {
      HashMap<String,String> hm = new HashMap<>();
      hm.put("id",String.valueOf(contact.get_id()));
      hm.put("name",contact.get_Name());
      hm.put("no",contact.get_phNo());
hmlist.add(hm);
   }
   registerForContextMenu(lv);
   String[] from = {"id","name","no"};
int[] to ={R.id.textView,R.id.textView2,R.id.textView3};
   SimpleAdapter adapter = new SimpleAdapter(getActivity(),hmlist,R.layout.cust,from,to);
lv.setAdapter(adapter);
   adapter.notifyDataSetChanged();
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
menuInfo) {
   MenuItem mi=menu.add(0,1,0,"Update");
   MenuItem mi1=menu.add(0,2,0,"Delete");
super.onCreateContextMenu(menu, v, menuInfo);
}
@Override
public boolean onContextItemSelected(MenuItem item) {
   AdapterView.AdapterContextMenuInfo
acmi=(AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
```

```
int position=acmi.position;
final Contact contact=list.get(position);

if(item.getItemId()==1){
      UpdateFragment fragment= new UpdateFragment();
      Bundle bundle=new Bundle();
      bundle.putInt("id",contact.get_id());
      bundle.putString("name",contact.get_Name());
      bundle.putString("num",contact.get_phNo());
 fragment.setArguments(bundle);
      getFragmentManager().beginTransaction().replace(android.R.id.content,fragment).commit();
   }
if(item.getItemId()==2){
         DBHandler db = new DBHandler(getActivity());
  db.delete(contact.get_id());
         AlertDialog.Builder alert = new AlertDialog.Builder(getActivity());
         alert.setTitle("Confirm Delete");
         alert.setMessage("Are You Sure to Delete ");
         alert.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
public void onClick(DialogInterface dialog, int which) {
               refresh();
               FragmentTransaction ft =getFragmentManager().beginTransaction();
               Fragment fragment = Fragment.instantiate(getActivity(),
ShowDataFragment.class.getName());
               ft.replace(android.R.id.content,fragment);
  ft.commit(); }          });
         alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
public void onClick(DialogInterface dialog, int which) {
}});
 alert.show();}
```

- **In Cust xml layout file**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent"
   android:orientation="vertical">
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Text"
      android:id="@+id/textView" />
```

```
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Text"
      android:id="@+id/textView2" />
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Text"
      android:id="@+id/textView3" />
</LinearLayout>
```

In Fragment Update Data xml file:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent"
tools:context="com.example.admin.sqlitedb.UpdateFragment">
<EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/update_name"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />
<EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:inputType="phone"
      android:ems="10"
      android:id="@+id/update_num"
      android:layout_below="@+id/update_name"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />
<Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Update"
      android:id="@+id/update"
      android:layout_below="@+id/update_num"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
```

```
    android:layout_marginTop="55dp" />
</RelativeLayout>
```

In Update Fragment java file:

```
public class UpdateFragment extends Fragment {
   EditText ed_name,ed_num;
   Button update;
int id;
   String name,num;
   FragmentTransaction ft;

public UpdateFragment() {
// Required empty public constructor
}
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view =inflater.inflate(R.layout.fragment_update, container, false);
ed_name=(EditText)view.findViewById(R.id.update_name);
ed_num=(EditText)view.findViewById(R.id.update_num);
update=(Button)view.findViewById(R.id.update);

final int id=getArguments().getInt("id");
final String name=getArguments().getString("name");
final String num=getArguments().getString("num");
ed_name.setText(name);
ed_num.setText(num);

update.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            String new_name = ed_name.getText().toString();
            String new_num = ed_num.getText().toString();
            Contact user= new Contact();
            user.set_id(id);
            user.set_Name(new_name);
      user.set_phNo(new_num);
            DBHandler dbHandler =  new DBHandler(getActivity());
      dbHandler.update(user);
            ShowDataFragment sat = new ShowDataFragment();
```

```
ft = getFragmentManager().beginTransaction();
ft.replace(android.R.id.content, sat);
ft.commit();}
     });return view;}}
```

```
package com.example.arpandb;

public class DataDemo extends Activity {

      public ArrayList<HashMap<String, String>> ContactList = new
ArrayList<HashMap<String, String>>();

      private static final String TAG_ID = "id";
      private static final String TAG_NAME = "name";
      private static final String TAG_PHNO = "phNo";

      private static int itemIndex;
      private ListView lst;
      private List<Contact> contacts;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_database_demo);
            lst = (ListView) findViewById(R.id.lstContact);
      }
//Create context menu to edit and delete
      @Override
      public void onCreateContextMenu(ContextMenu menu, View v,
                  ContextMenuInfo menuInfo) {
            // TODO Auto-generated method stub
            super.onCreateContextMenu(menu, v, menuInfo);
            if (v.getId() == R.id.lstContact) {
                  menu.setHeaderIcon(R.drawable.message);
                  menu.setHeaderTitle("Contacts");
                  menu.add(0, 1, Menu.NONE, "Edit Contact");
                  menu.add(0, 2, Menu.NONE, "Delete Contact");
                  menu.add(0, 3, Menu.NONE, "Cancel");
            }

      }

      @Override
      public boolean onContextItemSelected(MenuItem item) {
            // TODO Auto-generated method stub
            AdapterView.AdapterContextMenuInfo menuInfo;
            menuInfo = (AdapterContextMenuInfo) item.getMenuInfo();
            itemIndex = menuInfo.position;
            final Contact con = contacts.get(itemIndex);

            switch (item.getItemId()) {
            case 1:
```

```
	//Open intent to edit data
			Intent editIntent = new Intent(DataDemo.this,
UpdateContact.class);
			editIntent.putExtra(TAG_ID, con.getID());
			editIntent.putExtra(TAG_NAME, con.getName());
			editIntent.putExtra(TAG_PHNO, con.getPhNo());
			startActivity(editIntent);
			break;
		case 2:
//Delete Confirmation Dialog

			new AlertDialog.Builder(this)
				.setIcon(R.drawable.document_delete)
				.setTitle("Confirm Delete")
				.setMessage("Are you sure?")
				.setPositiveButton("Yes",
					new DialogInterface.OnClickListener()
{
	public void onClick(DialogInterface dialog,int which) {
		// TODO Auto-generated method stub
		DatabaseHandler db = new DatabaseHandler(DataDemo.this);
		db.DeleteContact(con); //Delete Data
		Toast.makeText(DataDemo.this,con.getName() + " Deleted",
			Toast.LENGTH_SHORT).show();
				fillList();
			}
		}).setNegativeButton("No", null).show();
			break;
		default:
			break;
		}
		return super.onContextItemSelected(item);
	}
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuItem addNew = menu.add(0, 1, 0, "Create Contact");

		addNew.setOnMenuItemClickListener(new OnMenuItemClickListener() {
			public boolean onMenuItemClick(MenuItem item) {
		// TODO Auto-generated method stub
		Intent iCreate = new Intent(DataDemo.this, CreateContact.class);
			startActivity(iCreate);
			return true;
			}
		});
		return true;
	}
	protected void fillList() {
		DatabaseHandler db = new DatabaseHandler(this);
		ListAdapter adap;
		ContactList.clear();
		contacts = db.ReadAllContact();

		for (Contact cn : contacts) {
			HashMap<String, String> hmap = new HashMap<String, String>();
```

```
                  hmap.put(TAG_NAME, cn.getName());
                  hmap.put(TAG_PHNO, cn.getPhNo());
                  ContactList.add(hmap);
                  adap = new SimpleAdapter(this, ContactList,
                              R.layout.list_item_view,
                              new String[] { TAG_NAME, TAG_PHNO }, new int[] {
                                          R.id.lstName, R.id.lstPhNo });
                  lst.setAdapter(adap); //Set adapter to fill Data
                  registerForContextMenu(lst);
            }
      }
      @Override
      protected void onResume() {
            // TODO Auto-generated method stub
            super.onResume();
            fillList();
      }
}
```

activity_create_contact.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

<EditText
        android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Your Name"
        android:inputType="textNoSuggestions" />

<EditText
        android:id="@+id/txtPhNo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtName"
        android:hint="Enter Phone No"
        android:inputType="number" />

<Button
        android:id="@+id/btnCreate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtPhNo"
        android:text="Create Contact" />

</RelativeLayout>
```

CreateContact.java

```
package com.example.arpandb;

public class CreateContact extends Activity {

      private EditText etName, etPhNo;
```

```
	private Button btAdd;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_create_contact);

		final DatabaseHandler dHabdler = new DatabaseHandler(this);

		etName = (EditText) findViewById(R.id.txtName);
		etPhNo = (EditText) findViewById(R.id.txtPhNo);
		btAdd = (Button) findViewById(R.id.btnCreate);

		btAdd.setOnClickListener(new OnClickListener() {

			public void onClick(View v) {
				// TODO Auto-generated method stub
				String name = etName.getText().toString();
				String phno = etPhNo.getText().toString();

				if (name.length() == 0 && phno.length() == 0) {
					etName.setError("Enter Name");
					etPhNo.setError("Enter Phone No");
				} else if (name.length() == 0) {
					etName.setError("Enter Name");
				} else if (phno.length() == 0) {
					etPhNo.setError("Enter Phone No");
				} else {
		//Call method to Insert Data
					dHabdler.AddContact(new Contact(name, phno));
					finish();
				}
			}
		});

	}
}
```

UpdateContact.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

<EditText
        android:id="@+id/txtNname"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="textNoSuggestions" />

<EditText
        android:id="@+id/txtNPhNo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtNname"
        android:inputType="number" />
```

```
<Button
        android:id="@+id/btnUpdate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtNPhNo"
        android:text="Update Contact" />

<TextView
        android:id="@+id/txtId"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/btnUpdate" />

</RelativeLayout>
```

UpdateContact.java

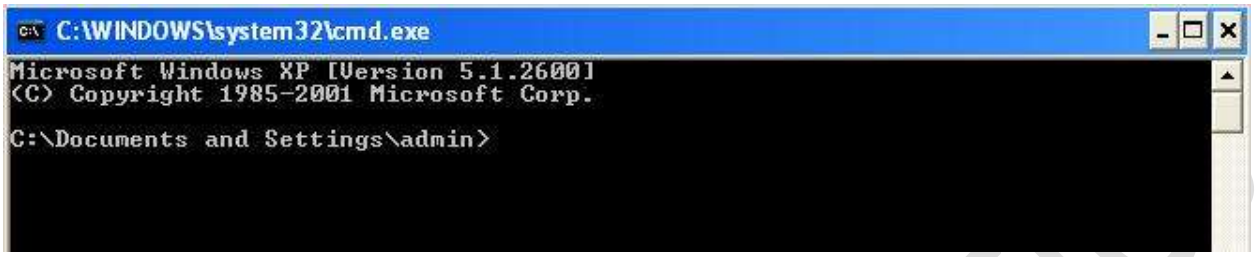
```
package com.example.arpandb;

public class UpdateContact extends Activity {

      private EditText txtName,txtPh;
      private Button btnUpdate;
      private TextView txt;
      private static final String TAG_ID="id";
      private static final String TAG_NAME = "name";
      private static final String TAG_PHNO = "phNo";
      private static int id;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_contact);
        txtName=(EditText)findViewById(R.id.txtNname);
        txtPh=(EditText)findViewById(R.id.txtNPhNo);
        btnUpdate=(Button)findViewById(R.id.btnUpdate);
        txt=(TextView)findViewById(R.id.txtId);

        Intent i=getIntent();
        id=i.getIntExtra(TAG_ID,0);
        txtName.setText(i.getStringExtra(TAG_NAME));
        txtPh.setText(i.getStringExtra(TAG_PHNO));

        btnUpdate.setOnClickListener(new OnClickListener() {

            public void onClick(View arg0) {
            // TODO Auto-generated method stub
            DatabaseHandler handler=new DatabaseHandler(UpdateContact.this);
                        Contact contact=new Contact();
                        contact.setID(id);
                        contact.setName(txtName.getText().toString());
                        contact.setPhNo(txtPh.getText().toString());
                  // Call method to update data
                        handler.UpdateContact(contact);
                        finish();
                  }
            });
    }
```

```
}
```

Exploring Databases on the Emulator and Available Devices

From your Start menu, click Run, type **CMD** or **COMMAND** in the Run dialog box, and click OK.

Executing this command launches the command window shown next. This window is the equivalent of the older DOS operating environments.



Many of these tools reside in the \android-sdk-installdirectory\tools subdirectory.

One of the tools is a remote shell on the device that allows you to execute a **command line** SQLite tool against a specified database. You'll see in this section how to use thiscommand-line utility to examine the built-in Android databases.

Android uses another command-line tool called Android Debug Bridge (adb), which isavailable as

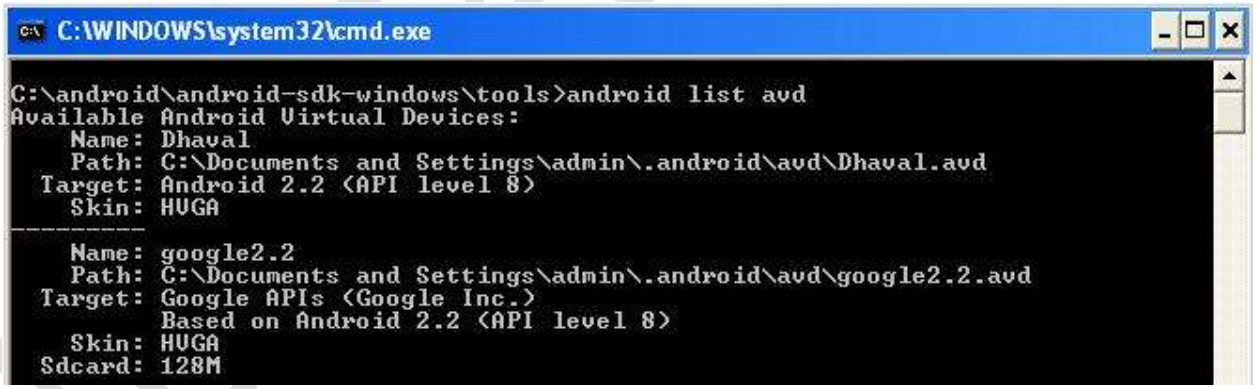
tools\adb.exe

adb is a special tool in the Android toolkit that most other tools go through to get to thedevice. However, you must have an emulator running or an Android device connectedfor adb to work.

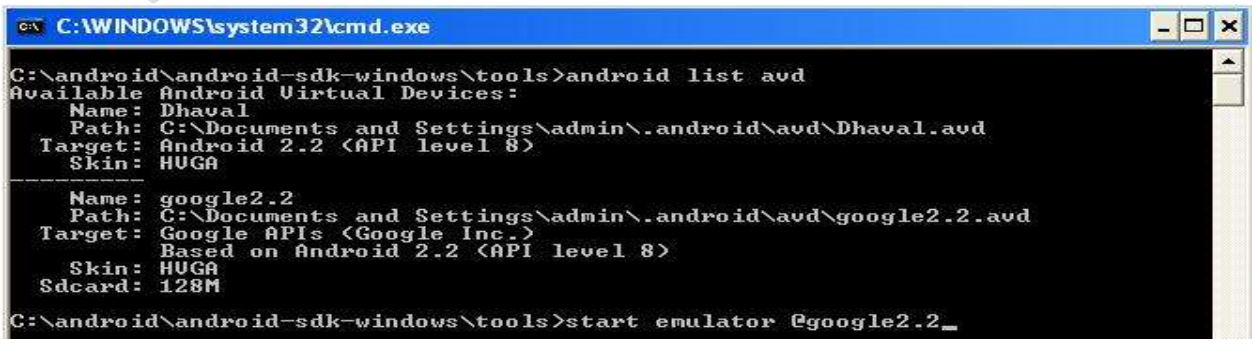
To find out what virtual devices you already have you can run the following command:

\tools\android list avd

This command will list the available AVD. If you have developed and run any Android applications through Eclipse ADT then you will have configured at least one virtual device. The above command will list at least that one virtual device.



If the emulator is not running, you can start the emulator by typing this at the command line: **Start emulator @name [name of emulator]**



You can see the many options and commands that you can run with adb by typing thisat the command line: **adb help**

You can use adb to open a shell on the connected device by typing this: **adb shell**

You can see the available command set in the shell by typing this at the shell prompt:

#ls /system/bin (The # sign is the prompt for the shell.)

To see a list of root-level directories and files, you can type the following in the shell:

ls -l

You'll need to access this directory to see the list of databases:

ls /data/data

You can invoke sqlite3 on one of these databases inside the adb shell by typing this:

#sqlite3 /data/data/com.android.providers.contacts/databases/contacts.db

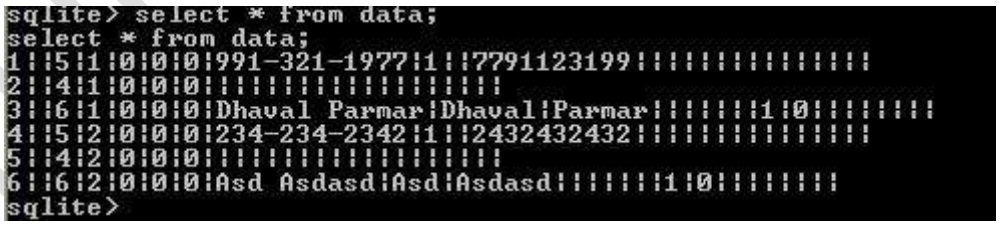
You can exit sqlite3 by typing this:

sqlite>.exit;

Notice that the prompt for adb is # and the prompt for sqlite3 is sqlite>. You can read about the various sqlite3 commands by visiting http://www.sqlite.org/sqlite.html.However, we will list a few important commands here so that you don't have to make atrip to the web.

You can see a list of tables by typing.

sqlite> .tables



Create a simple table from the command prompt:

Create table syntax:

Create table statement use to create table in sqlite.

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
```

```
column_name3 data_type,
.......
)
```

The following example shows how you can create a table named "person", with three columns. The column names will be "First Name", "Last Name" and "Age".

```
create table person (id INTEGERPRIMARY KEY, Firstname TEXT, Lastname TEXT, Age INTEGER);
```

Show in command prompt.

```
sqlite> create table person (id INTEGER PRIMARY KEY, firstname TEXT, lastname TE
XT, Age INTEGER);
create table person (id INTEGER PRIMARY KEY, firstname TEXT, lastname TEXT, Age
INTEGER);
sqlite>
```

Insert Data In to a Table

The INSERT INTO statement is used to add new records to a database table .

Syntax:

```
INSERT INTO table_name (column1, column2,...) VALUES (value1, value2,....) ;
```

Example:

```
insert into person (id, Firstname, Lastname) values (1,'dhaval','parmar');
```

```
sqlite> insert into person(id,firstname,lastname) values(1,'dhaval','parmar');
insert into person(id,firstname,lastname) values(1,'dhaval','parmar');
sqlite>
```

Select Data From a Table

```
      The SELECT statement is used to select data from a database.
```

Syntax:

```
SELECT column_name(s) FROM table_name ;
```

Example : select * from person;

```
sqlite> select * from person;
select * from person;
1|dhaval|parmar|
sqlite>
```

Realm-No SQL Database :

1. Add gradle to the project :

```
dependencies {
        compile 'io.realm:realm-android:0.87.4'
}
```

2. Create instance :

```
public class MainActivity extends AppCompatActivity {
        Realm realm;
         @Override
        protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
        realm = Realm.getInstance(MainActivity.this);
}
```

3. Add data to realm :

Put this code into button click

Void adddata()

```
{
        realm.beginTransaction();
        RealmPojo rm = realm.createObject(RealmPojo.class);
        rm.setName(ed.getText().toString());
        realm.commitTransaction();
}
```

4.Get all data :

```
void getallname()
{
 list = new ArrayList<>();
 RealmResults<RealmPojo> results = realm.where(RealmPojo.class).findAll();
 realm.beginTransaction();
 for (int i = 0; i < results.size(); i++) {
    list.add(results.get(i));
 }
 CustAdapter ad= new CustAdapter(MainActivity.this,list);
 lv.setAdapter(ad);
 realm.commitTransaction();
}
```

5. Remove data from realm :

```
void removerealm(String i){
 RealmResults<RealmPojo> results =  realm.where(RealmPojo.class).equalTo("name",i).findAll();

 realm.beginTransaction();
 results.remove(0);
 realm.commitTransaction();
}
```

6. Update realm database :

```
void updaterealm(String name,String find)
{
 RealmPojo  rmp = realm.where(RealmPojo.class).equalTo("name", find).findFirst();
 realm.beginTransaction();
 rmp.setName(name);
 realm.commitTransaction();
 }
```

7. Get single data using parameters :

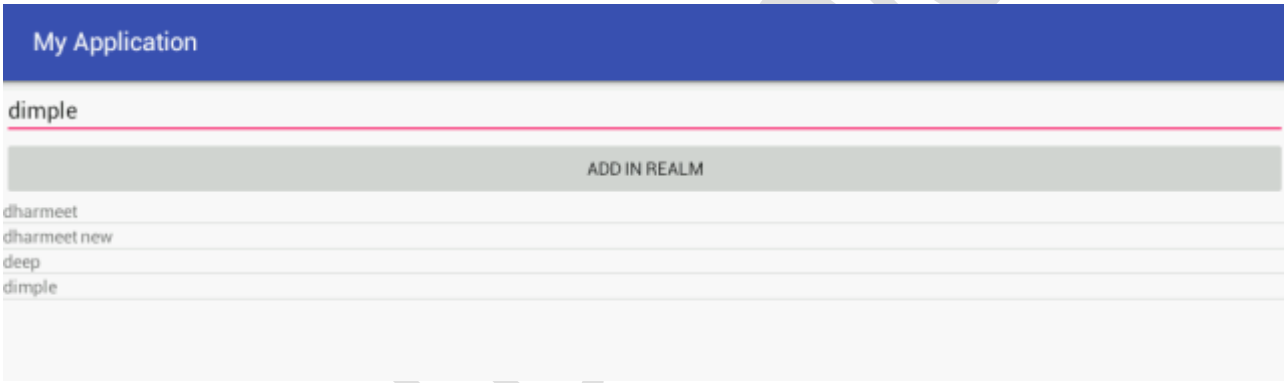
```
void getsignle(String name){
 list = new ArrayList<>();
RealmResults<RealmPojo>results=
                realm.where(RealmPojo.class).equalTo("name", i).findAll();
 realm.beginTransaction();
 for (int i = 0; i < results.size(); i++) {
    list.add(results.get(i));
```

```
 }

CustAdapter ad= new CustAdapter(MainActivity.this,list);
  lv.setAdapter(ad);
  realm.commitTransaction();
}
```

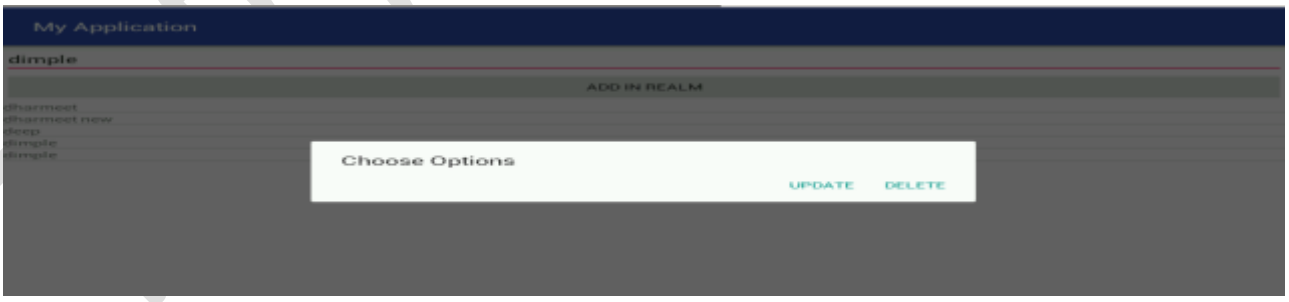
Results :

Add data :

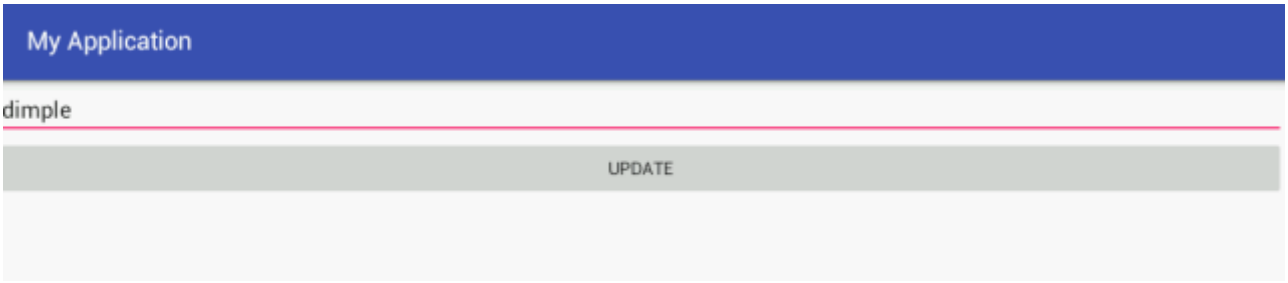
Getall data :



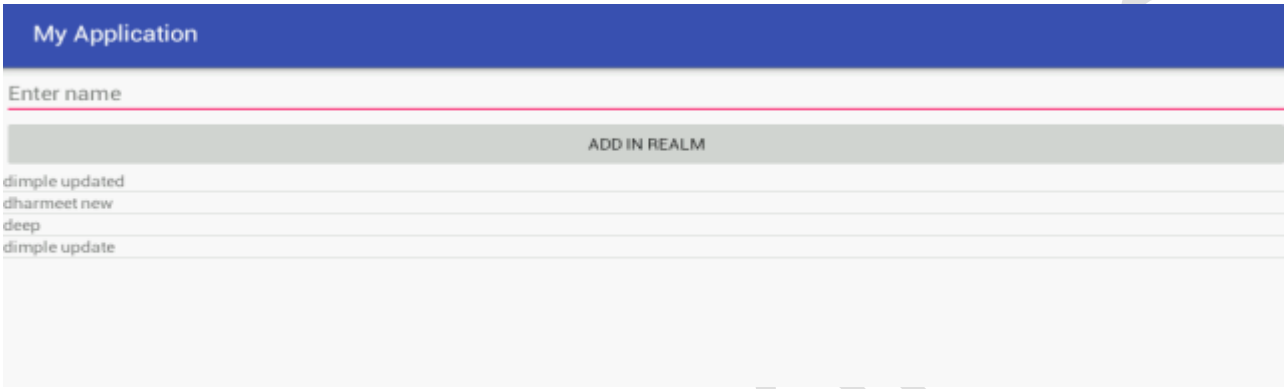
Options :



Update data :

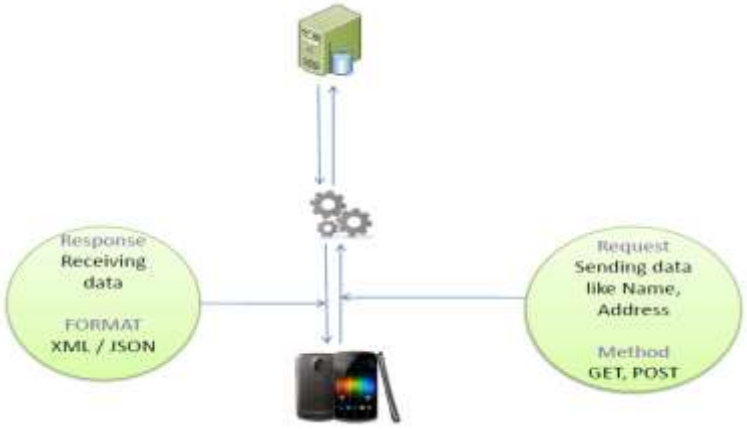


Updated data :



Web Services and Data Parsing

"web service" as "a software system designed to support interoperablemachine-to-machine interaction over a network.



Web Service Integration

As more and more mobile applications are dependent on the web in order to function properly (e.g. Twitter, Facebook, news apps etc.), integrating web services into mobile applications is an increasingly common scenario. One of the most common functionalities required in mobile applications is to call a web service to retrieve data. This process involves requesting the web service with parameters, receiving the response and parsing it to obtain data. Today the most common web services types are **SOAP** and **REST**. Android does not provide a built in SOAP client, there are many third party libraries that can be used, but we'll see how to call a SOAP web service with native android APIs.

Soap Web Service

- Simple Object Access Protocol
- An XML-based messaging protocol.
- Communication between the web service and client happens using XML messages

• SOAP defines the rules for communication like what are all the tags that should be used in XML and their meaning

XML PARSING

DOM parsing
Sax Parsing
Json Parsing

Pull Parsing :

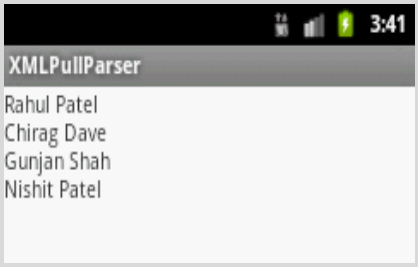
The term *XML resources* sometimes confuse new Android developers. XML resources might mean resources in general that are defined in XML—such as layout files, styles, arrays, and the like—or it can specifically mean res/xml XML files.
In this section, we'll deal with res/xml XML files. These files are different from raw files in that you don't use a stream to access them because they're compiled into an efficient binary form when deployed. They're different from other resources in that they can be of any custom XML structure.
To demonstrate this concept, we're going to use an XML file named people.xml that defines multiple <person>elements and uses attributes for firstname and lastname. We'll grab this resource and display its elements in *last-name, first-name* order, as shown in
Figure.

XML File

```
<?xml version="1.0" encoding="utf-8"?>
<people>
<person fname="Rahul" lname="Patel"/>
<person fname="Chirag" lname="Dave"/>
<person fname="Gunjan" lname="Shah"/>
<person fname="Nishit" lname="Patel"/>
</people>
```

Layout XML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<TextView
        android:id="@+id/txtXML"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```



Activity Java File

```
public class XMLPullActivity extends AppCompatActivity {

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_xmlpull);

        PullFragment fragment = new PullFragment();
        FragmentTransaction ft =getFragmentManager().beginTransaction();
        ft.replace(android.R.id.content, fragment);
        ft.commit();
    }
}
```

In fragment java file:

```
public class PullFragment extends Fragment {
   ListView lv;
   ArrayList<String>list;
public PullFragment() {
// Required empty public constructor
}
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view = inflater.inflate(R.layout.fragment_pull, container, false);
      StrictMode.ThreadPolicy policy =new StrictMode.ThreadPolicy.Builder().permitAll().build();
      StrictMode.setThreadPolicy(policy);
lv = (ListView) view.findViewById(R.id.listView);
list = new ArrayList<String>();
      XmlPullParser parser = getResources().getXml(R.xml.dataxml);
try {
while (parser.next() != XmlPullParser.END_DOCUMENT) {
            String fname = null, lname=null , data;
            String tagName = parser.getName();
int attrcount = parser.getAttributeCount();
for (int i = 0; i < attrcount; i++) {
               String attrname = parser.getAttributeName(i);

if (attrname.equals("fname")) {
                  fname = parser.getAttributeValue(i);
               }
if (attrname.equals("lname")) {
                  lname = parser.getAttributeValue(i);
                  data = fname + " " + lname;

list.add(data);
                  System.out.println();
               }}}
lv.setAdapter(new ArrayAdapter<String>(getActivity(), android.R.layout.simple_list_item_1, list));

      } catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
      }
return view; }}
```

If you're using Eclipse, it'll automatically detect a file in the res/xml path and compile it into a resource asset. You can then access this asset in code by parsing its binary XML, as shown in the following listing.

Dom parsing

DOM parsing on Android is fully supported. It works exactly as it works in Java code that you would run on a desktop machine or a server
Fetch data from xml (remote)file using DOM parser.

In your main Activity.

```
public class DOMParsing extends AppCompatActivity {
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      DOMFragment fragment = new DOMFragment();
      FragmentTransaction ft =getFragmentManager().beginTransaction();
      ft.replace(android.R.id.content,fragment);
      ft.commit();
   }
}
```

In your Fragment.

```
public class DOMFragment extends Fragment {

   ListView lv;
   URL url;
private ArrayList<String>list;

public DOMFragment() {
// Required empty public constructor
}
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view = inflater.inflate(R.layout.fragment_dom, container, false);
      StrictMode.ThreadPolicy policy =new StrictMode.ThreadPolicy.Builder().permitAll().build();
      StrictMode.setThreadPolicy(policy);

lv=(ListView)view.findViewById(R.id.listView);
list =new ArrayList<String>();

try {
url=new URL("http://www.w3schools.com/aspnet/countries.xml");

          DocumentBuilderFactory dbf =DocumentBuilderFactory.newInstance();
          DocumentBuilder db = dbf.newDocumentBuilder();
          Document document = db.parse(url.openStream());
          document.normalize();

          NodeList nodeList = document.getElementsByTagName("country");
for(int i=0;i<nodeList.getLength();i++)
          {
             Node node = nodeList.item(i);
             Element element =(Element)node;

             String text =getDataDOM(element,"text");
             String value = getDataDOM(element, "value");

             String data =text+" "+value;
list.add(data);
          }
      } catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
      }

lv.setAdapter(new ArrayAdapter<String>(getActivity(),    android.R.layout.simple_list_item_1,list));;
return view;
   String getDataDOM(Element element, String string) {
// TODO Auto-generated method stub
NodeList nList = element.getElementsByTagName(string);
      Node chilenode= nList.item(0);
      Element ele = (Element)chilenode;
      String textdata = ele.getTextContent();
return textdata;
   }
}
```

SAX Parsing

Fetch data from xml (remote)file using SAX parser.

Xml file

In your main Activity.

```
public class MainActivity extends AppCompatActivity {
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      SAXFragment fragment = new SAXFragment();
      FragmentTransaction ft =getFragmentManager().beginTransaction();
      ft.replace(android.R.id.content, fragment);
      ft.commit();    }
}
```

In your Fragment class

```
public class SAXFragment extends Fragment {
   ArrayList<String>list = new ArrayList<String>();
   ListView lv;
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment_sax, container, false);
lv=(ListView)view.findViewById(R.id.listView);
StrictMode.ThreadPolicy policy =new StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
String URL="http://www.w3schools.com/aspnet/countries.xml";
SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser parser;
try {
  parser = spf.newSAXParser();
```

```
   DefaultHandler dh = new DefaultHandler() {
      String text, value, data;
boolean textb, valueb;

      @Override
public void startElement(String uri, String localName,
                        String qName, Attributes attributes)
throws SAXException {
// TODO Auto-generated method stub
super.startElement(uri, localName, qName, attributes);
if (localName.equals("text")) {
textb = true;
         }
if (localName.equals("value")) {
valueb = true;
         }
      }
public void endElement(String uri, String localName, String qName) throws SAXException {
// TODO Auto-generated method stub
super.endElement(uri, localName, qName);
if (localName.equals("text")) {
textb = false;
                }
if (localName.equals("value")) {
valueb = false;
          }
             }
public void characters(char[] ch, int start, int length)
throws SAXException {
// TODO Auto-generated method stub
super.characters(ch, start, length);
if (textb) {
text = new String(ch, start, length);
                }
if (valueb) {
value = new String(ch, start, length);
data = text + " - " + value;
list.add(data);
                }
        } };
parser.parse(URL, dh);
lv.setAdapter(new ArrayAdapter<String>(getActivity(), android.R.layout.simple_list_item_1, list));
      } catch (Exception e) {
// TODO Auto-generated catch block
```

```
e.printStackTrace();
    }
return view ;
  }}
```

JSON PARSING

What is JSON?

JSON is the best alternative to XML for storing data in files. It is easy to parse and access data stored in JSON format.

- JSON stands for **JavaScript Object Notation**
- JSON is lightweight text-data interchange format
- JSON is language independent .
- JSON is "self-describing" and easy to understand
- JSON uses JavaScript syntax for describing data objects, but JSON is still language and platform independent. JSON parsers and JSON libraries exists for many different programming languages.
- JSON is a very short data exchange format. Android includes the json.org libraries which allow to work easily with JSON files.

JSON Example

Json Data is look like shown in the following snippet.

```
{
myObject : {
 "first": "John",
 "last": "Doe",
 "age": 39,
 "sex": "M",
 "salary": 70000,
 "registered": true,
 "interests": [ "Reading", "Mountain Biking", "Hacking" ]
} }
```

JSON Object

- A JSONObject is an unordered collection of name/value pairs. Its external form is a string wrapped in curly braces with colons between the names and values, and commas between the values and names.
- Ex : `{Name: Alpa, Surname: Detval}`

JSON Array

- A JSONArray is an ordered sequence of values. Its external text form is a string wrapped in square brackets with commas separating the values.

```
EX : [{Name : Alpa, Surname : Detval},
      { Name :Yug,  Surname:Detval }];
```

Get JSON Data From Web service

Read Data From Following Url.

http://api.androidhive.info/contacts/

Create JSON Parser Class to process the Request

```
public class JSONParser {
        static JSONObject jObj = null;
    static String json = "";
    private OkHttpClient client;
    public JSONObject getJSONFromUrl(String url) {
        try {
            client = new OkHttpClient();
           Request request = new Request.Builder()
     .url(url)
     .build();
            Response response = client.newCall(request).execute();
            json = response.body().string();
           jObj = new JSONObject(json);
        } catch (Exception e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }
        return jObj;
    } }
```

Note: For OkHttpClient add jar dependency in android studio using following steps

- In android studio, go to Project Structure.
- In Modules Section goto -> app -> select Dependencies tab
- Click on + sign on right side, choose 1)Library Dependency
- And Select OkHttp.

Step 2:To create Customized List View Create below XML file under res/layout Folder

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
            <!-- Name Label -->
<TextView
            android:id="@+id/name"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="#43bd00"
            android:textSize="16sp"
            android:textStyle="bold"
            android:paddingTop="6dip"
            android:paddingBottom="2dip" />
            <!-- Description label -->
<TextView
            android:id="@+id/email"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="#acacac"
            android:paddingBottom="2dip">
</TextView>
<TextView
            android:id="@+id/mobile"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="#acacac"
            android:paddingBottom="2dip">
</TextView>
<!-- Linear layout for cost and price Cost: Rs.100 -->
</LinearLayout>
</LinearLayout>
```

Step : 3 To View the content of XML file , Create ListView in main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">
<ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
</ListView>
</LinearLayout>
```

Step :4 Write code in your Main Activity:

```
public class AndroidJSONParsingActivity extends ListActivity {
	// url to make request
	private static String url = "http://api.androidhive.info/contacts/";

	// JSON Node names
	private static final String TAG_CONTACTS = "contacts";
	private static final String TAG_ID = "id";
	private static final String TAG_NAME = "name";
	private static final String TAG_EMAIL = "email";
	private static final String TAG_ADDRESS = "address";
	private static final String TAG_GENDER = "gender";
	private static final String TAG_PHONE = "phone";
	private static final String TAG_PHONE_MOBILE = "mobile";
	private static final String TAG_PHONE_HOME = "home";
	private static final String TAG_PHONE_OFFICE = "office";
	// contacts JSONArray
	JSONArray contacts = null;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

	// Hashmap for ListView
	ArrayList<HashMap<String, String>> contactList = new ArrayList<HashMap<String, String>>();
	// Creating JSON Parser instance
	JSONParser jParser = new JSONParser();
	// getting JSON string from URL
	JSONObject json = jParser.getJSONFromUrl(url);
try {
	// Getting Array of Contacts
	contacts = json.getJSONArray(TAG_CONTACTS);
	// looping through All Contacts
	for(int i = 0; i < contacts.length(); i++){
		JSONObject c = contacts.getJSONObject(i);
		// Storing each json item in variable
		String id = c.getString(TAG_ID);
		String name = c.getString(TAG_NAME);
		String email = c.getString(TAG_EMAIL);
		String address = c.getString(TAG_ADDRESS);
		String gender = c.getString(TAG_GENDER);
		// Phone number is again JSON Object
		JSONObject phone = c.getJSONObject(TAG_PHONE);
		String mobile = phone.getString(TAG_PHONE_MOBILE);
		String home = phone.getString(TAG_PHONE_HOME);
		String office = phone.getString(TAG_PHONE_OFFICE);
```

```
                // creating new HashMap
                HashMap<String, String> map = new HashMap<String,
        String>();
                // adding each child node to HashMap key => value
                map.put(TAG_ID, id);
                map.put(TAG_NAME, name);
                map.put(TAG_EMAIL, email);
                map.put(TAG_PHONE_MOBILE, mobile);
                // adding HashList to ArrayList
                contactList.add(map);
        }
}
catch (JSONException e) {
        e.printStackTrace();
}
/**
 * Updating parsed JSON data into ListView
 * */
ListAdapter adapter = new SimpleAdapter(this, contactList,
                        R.layout.list_item,new String[] { TAG_NAME,
        TAG_EMAIL, TAG_PHONE_MOBILE }, new int[] {
                        R.id.name, R.id.email, R.id.mobile });
setListAdapter(adapter);
// selecting single ListView item
ListView lv = getListView();
        }
}
```

Step:6 To Access Internet set permission in android.manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jsonparserdemo"
    android:versionCode="1"
    android:versionName="1.0" >
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />
      <uses-permission  android:name="android.permission.INTERNET"/>
<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
<activity
            android:name="com.example.jsonparserdemo.MainActivity"
            android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

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

Async Task

- **AsyncTask** enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.
- **AsyncTask is a generic class**, it uses 3 types: AsyncTask<Params, Progress, Result>.
    - **Params** – the input. what you pass to the AsyncTask
    - **Progress** – if you have any updates, passed to onProgressUpdate()
    - **Result** – the output. what returns doInBackground()

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
<Button
        android:id="@+id/btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="readWebpage"
        android:text="Load Webpage" >
</Button>
<Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="dummyFunc"
        android:text="Dummy Button" />
<WebView
        android:id="@+id/webView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

Java File

```
public class MainActivity extends Activity {
      final Context context = this;
      /** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
      }
      private class LoadWebPageASYNC extends AsyncTask<String, Void, String> {
            @Override
            protected String doInBackground(String... urls) {
                  WebView webView = (WebView) findViewById(R.id.webView);
                  webView.getSettings().setJavaScriptEnabled(true);
                  webView.loadUrl(urls[0]);
                  return null;
            }
            @Override
```

```
            protected void onPostExecute(String result) {
            }
      }
      public void dummyFunc(View view) {
            Toast.makeText(MainActivity.this, "Button Clicked",
Toast.LENGTH_SHORT)
                        .show();
      }
      public void readWebpage(View view) {
            LoadWebPageASYNC task = new LoadWebPageASYNC();
            task.execute(new String[] { "http://www.javacodegeeks.com" });
      }
}
```

Third Party Libraries :

- **OkHttp :**

  we alreday have seen example of OkHttp with JSON Parsing.

- **Retrofit :**

  Retrofit is RestClient in android to send and retrive JSON data using webservices.

  Retrofit relied on the Gson library to serialize and deserialize JSON data.

  For implementing Retrofit we need to add below dependencies :

  compile 'com.squareup.retrofit2:retrofit:2.0.0

  compile 'com.squareup.retrofit2:converter-gson:2.0.0

  Add permission in Menifest.xml :

```
<uses-permission android:name="android.permission.INTERNET"/>
```

  JSON URL Result :

```
{
            "emp": [
             {
            "e_id": "1",
             "e_name": "Jay",
             "email": "jay@ab.com",
             },
             {
            "e_id": "2",
              "e_name": "amit",
              "email": "amit@ab.com",
            }
           ]
}
```

  Make one Model class (Based on Your Json URL Response):

```
public class User {
  @SerializedName("email")
  private  String email;
  @SerializedName("e_name")
  private  String name;
```

```
  public String getEmail() {
  return email;
  }
public void setEmail(String email) {
 this.email = email;
}
public String getName() {
 return name;
  }
 public void setName(String name) {
  this.name = name;
 }
 @Override
public String toString() {
return name+" "+email;
 }
}
```

Now Create UserResponse Model class which handles your whole arrayList :

```
public class UserResponse {
 @SerializedName("emp")
 private List<User> empList;

 public List<User> getEmpList() {
 return empList;
}
public void setEmpList(List<User> empList) {
 this.empList = empList;
 }
}
```

Now Create One Interface for define URL END Points :

```
public interface ApiInterface {
 @GET("getdata.php")
 Call<UserResponse> getEmployees();

 @FormUrlEncoded
 @POST("postdata.php")
Call<ResponseBody> insertEmployee(@Field("e_name") String ename,
        @Field("email") String email);
}
```

Now Create one ApiClient to Define BaseURL with Retrofir class :

```
public class ApiClient {
        public static final String BASE_URL="http://www.dummydomain.890m.com/";
        static Retrofit retrofit=null;

        public static Retrofit getClient() {

        if (retrofit == null) {
                retrofit = new Retrofit.Builder()
                        .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create()).build();
        }
         return retrofit;
}

public static Retrofit getClientForString() {

        if (retrofit == null) {
        retrofit = new Retrofit.Builder()
```

```
                        .baseUrl(BASE_URL)
                        .build();
                }
                return retrofit;
        }
```

Now in MainActivity.java :

```
    public class MainActivity extends AppCompatActivity {
      EditText et_name,et_email,et_pass;
      Button btn,btnpost;
      ApiInterface inter;
            RecyclerView recyclerView;
            RecyclerViewAdapter adapter;
             ArrayList<UserDetail> arrayList=new ArrayList<>();
       protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et_email=(EditText)findViewById(R.id.et_email);
        et_name=(EditText)findViewById(R.id.et_name);
        et_pass=(EditText)findViewById(R.id.et_pass);
        btnpost=(Button) findViewById(R.id.btnpost);
        btn=(Button)findViewById(R.id.btn);
        inter=ApiClient.getClient().create(ApiInterface.class);
        btn.setOnClickListener(new View.OnClickListener() {
              public void onClick(View v) {
                    fetchData();
          }
        });
        btnpost.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            postData();
          }
        });
      }

       private void postData() {
          Call<ResponseBody>
       call=inter.insertEmployee(et_email.getText().toString(),
                          et_name.getText().toString(),
                         et_pass.getText().toString());

        call.enqueue(new Callback<ResponseBody>() {
          @Override
          public void onResponse(Call<ResponseBody> call, Response<ResponseBody>
                  response) {

                  try {
              Toast.makeText(MainActivity.this, response.body().string(),
                            Toast.LENGTH_SHORT).show();
         } catch (IOException e) {
                e.printStackTrace();
            }
         }

          @Override
          public void onFailure(Call<ResponseBody> call, Throwable t) {
                  }
        });

      }
```

```
    private void fetchData() {

      Call<UserResponse> call=inter.getEmployees();

     call.enqueue(new Callback<UserResponse>() {
     @Override
     public void onResponse(Call<UserResponse> call, Response<UserResponse>
        response)
                {
                        arrayList= (ArrayList<User>) response.body().getEmpList();
                         adapter=new RecyclerViewAdapter(MainActivity.this,arrayList);
                                recyclerView.setAdapter(adapter);
      }

      @Override
     public void onFailure(Call<UserResponse> call, Throwable t) {

                Toast.makeText(MainActivity.this, ""+t.toString(),      Toast.LENGTH_SHORT).show();
     }
     });
     }
  }
```

activity.xml File :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/activity_main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.admin.retrofitexample.MainActivity">
  <android.support.v7.widget.RecyclerView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/recycler"/>
</RelativeLayout>
```

At last we need to implement only recycler adapter class.

- **Glide :**

  Glide is an open source media management framework for Android that wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface. Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide includes a flexible API that allows developers to plug in to almost any network stack.

Update build.gradle file :

```
dependencies {
        compile 'com.github.bumptech.glide:glide:3.6.1'
        compile 'com.android.support:support-v4:19.1.0'
 }
```

Add INTERNET permission in manifest file :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Loading Images from server :

```
Glide.with(this)
.load("IMAGE URL HERE")
 .into(imageView);
```

Placeholder And Error Fallback And Image Resizing:

```
Glide.with(this)
    .load("IMAGE URL HERE")
    .placeholder(R.drawable.placeholder)
    .error(R.drawable.imagenotfound)
    .override(200, 200);
    .centerCrop();
    .into(imageView);
```

- **Picasso :**

Picasso is Android Image Loader librabry. If we want to display images into imageview which we are fatching from database, we can load it with Picasso.

To use Picasso Library in your project you need toadd dependencies in your gradle file.

```
dependencies {
  compile "com.squareup.picasso:picasso:2.4.0"
}
```

To load images use below code.

```
Picasso.with(this)
  .load("YOUR IMAGE URL HERE")
  .into(imageView);
```

Placeholder and Error Handling use below code.

```
Picasso.with(this)
  .load("YOUR IMAGE URL HERE")
  .placeholder(Your Drawable Resource) //this is optional the image to display while
                                       the url image is downloading
  .error(Your Drawable Resource)       //this is also optional if some error has
        occurred in downloading the image this image would be displayed
  .into(imageView);
```

Re-sizing and Rotating :

```
Picasso.with(this)
        .load("YOUR IMAGE URL HERE")
        .placeholder(DRAWABLE RESOURCE)   // optional
        .error(DRAWABLE RESOURCE)     // optional
        .resize(width, height)                // optional
        .rotate(degree)                   // optional
        .into(imageView);
```

Advance Android development

LOCATION AND MAPPING

Google Map v2

- Google provides Maps via Google play a library for using Google Maps in your application. The following description is based on the Google Maps Android API v2 which provides significant improvements to the older API version.

- If you have developed any app that contains Google Maps v1, It's time to upgrade it to Google Maps V2 as Google maps version 1 deprecated officially on December 3rd, 2012 and it won't work anymore.
- Before starting a new project, we need to go through some pre required steps. These steps involve importing required library, generating SHA1 fingerprint and configuring maps in Google console.

Google Map Project

1. Downloading Google Play Services

Open Studio⇒ Android SDK Manager and check whether you have already downloaded Google Play Services or not under Extras section. If not select play services and install the package.

2. Importing Google Play Services into Eclipse

After downloading play services we need to import it to Eclipse which will be used as a library for our maps project.

1. In Eclipse go to **File ⇒ Import ⇒ Android ⇒ Existing Android Code into Workspace**

2. Getting the Google Maps API key

1. Same as in maps v1 we need to generate SHA-1 fingerprint using java **keytool**. Open your terminal and execute the following command to generate SHA-1 fingerprint.

On Windows

```
keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
```

On Linux or Mac OS

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

In the output you can see SHA 1 finger print.

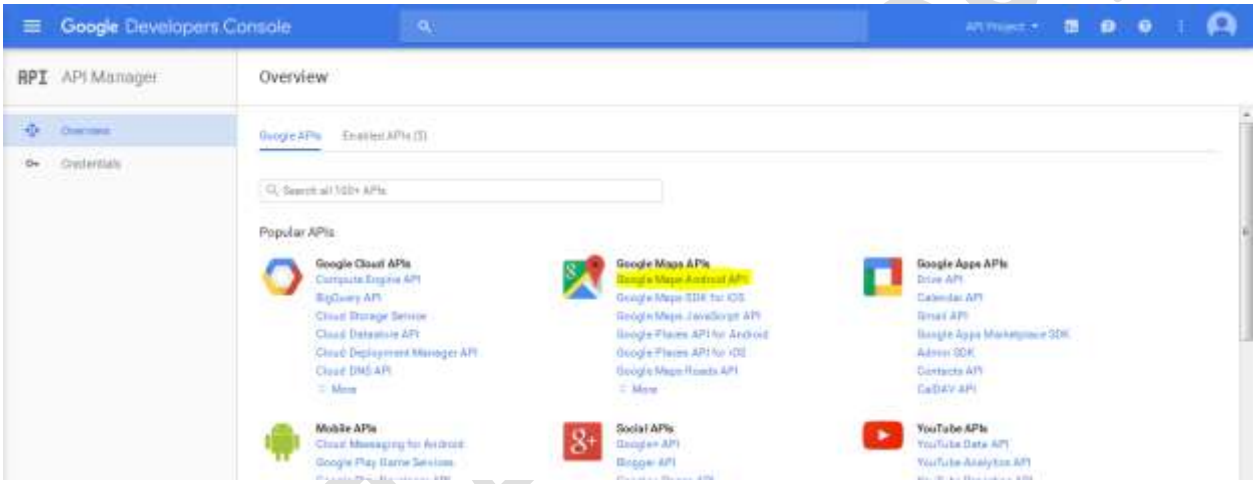
Generating SHA 1 fingerprint using keytool

```
Alias name: androiddebugkey
Creation date: May 13, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 6f3eb719
Valid from: Mon May 13 02:13:08 IST 2013 until: Wed May 06 02:13:08 IST 2043
Certificate fingerprints:
         MD5:  7F:21:AF:F4:0B:67:4F:88:12:90:54:69:5E:6D:BE:DE
         SHA1: E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2
         SHA256: 3E:23:20:84:9A:77:66:52:1E:1B:E2:D5:EB:13:BE:35:FA:A8:9F:B3:86:97:F6:15:E
:13:EE:AD:F9
         Signature algorithm name: SHA256withRSA
         Version: 3
```

2. Now open https://code.google.com/apis/console/

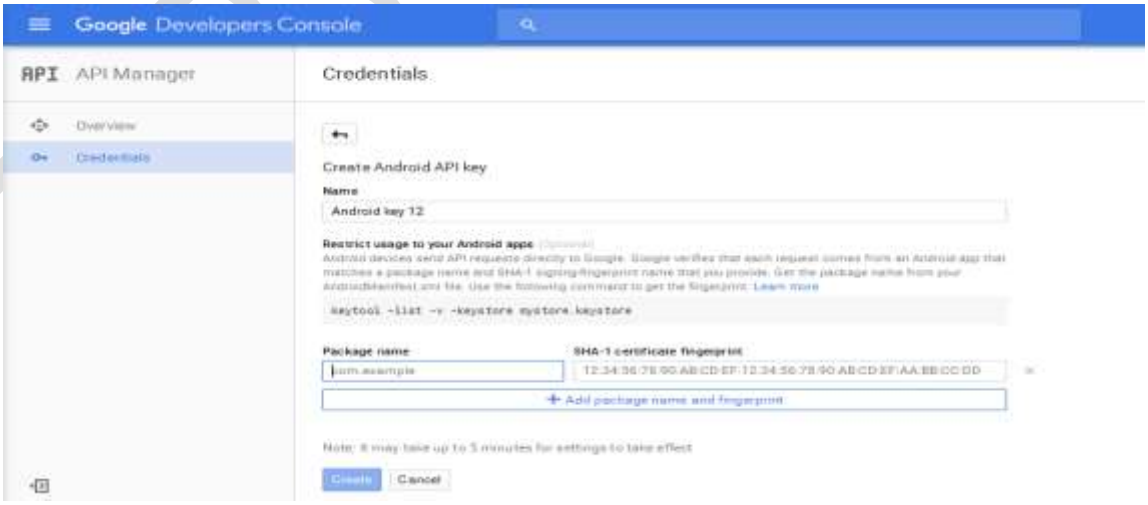
3. Now create and select a project .

4 Select **Google Maps Android API and Enable API**

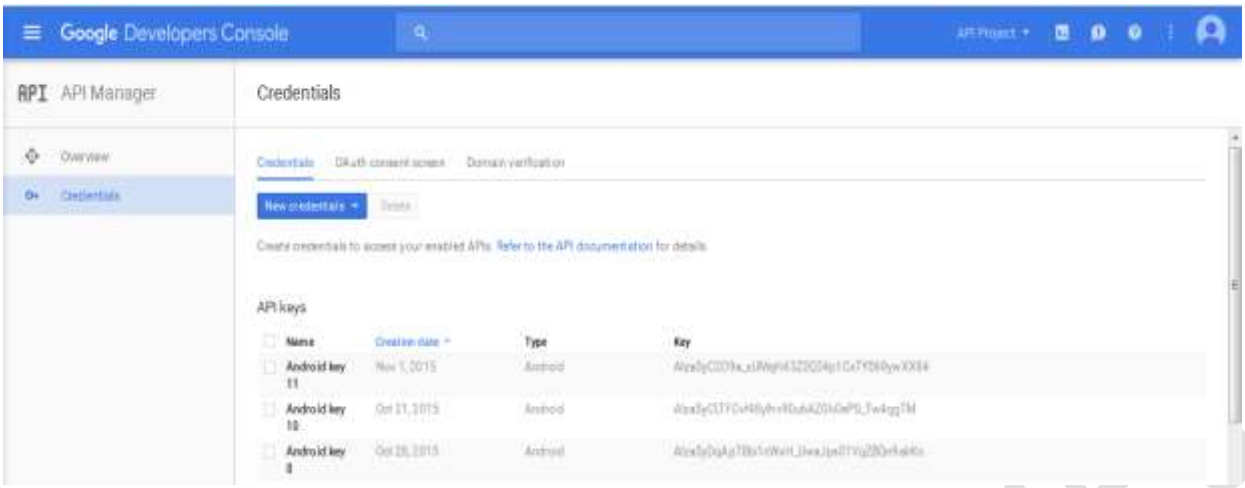


4. Now select credentials on left side and on the right side click on New Credentials - > API Key -> Android Key.

5. Add the SHA1 and package name. Enter your **SHA 1** and your **android project package name** separated by semicolon **;** and click on create.



And note down the API key which required later in our project.



Note: For google-play-services add jar dependency in android studio using following steps

- In android studio, go to Project Structure.
- In Modules Section goto -> app -> select Dependencies tab
- Click on + sign on right side, choose 1)Library Dependency

And Select google-play-services.

3.Add the Map Key in the manifest file. Open**AndroidManifest.xml** file and add the following code before tag. Replace the **android:value** with your map key which you got from google console.

```
<!-- Goolge Maps API Key -->
<meta-dataandroid:name  "com.google.android.maps.v2.API_KEY"
   android:value="AIzaSyBZMlkOv4sj-M5JO9p6wksdax4TEjDVLgo" />
```

4. Google maps needs following permissions and features.

ACCESS_NETWORK_STATE – To check network state whether data can be downloaded or not
INTERNET – To check internet connection status
WRITE_EXTERNAL_STORAGE – To write to external storage as google maps store map data in external storage
ACCESS_COARSE_LOCATION – To determine user's location using WiFi and mobile cell data
ACCESS_FINE_LOCATION – To determine user's location using GPS
OpenGL ES V2 – Required for Google Maps V2
Finally my **AndroidManifest.xml** file looks like this (Replace the package name with your project package)

In Manifest File:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.admin.mymaptest" >
<!--
      The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
      Google Maps Android API v2, but you must specify either coarse or fine
      location permissions for the 'MyLocation' functionality.
   -->
<permission android:name="com.example.admin.mymaptest.permission.MAPS_RECEIVE"
android:protectionLevel="signature"/>
<uses-permission android:name="com.example.admin.mymaptest.permission.MAPS_RECEIVE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
android:theme="@style/AppTheme">

<uses-feature
        android:glEsVersion="0x00020000"
    android:required="true"/>

<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="@string/google_maps_key"/>

<meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
<activity
        android:name=".MapsActivity"
        android:label="@string/title_activity_maps" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

5. It will pop up a window asking the SHA1 and package name. Enter your **SHA 1** and your **android project package name** separated by semicolon; and click on create.

6. New google maps are implemented using **MapFragments** which is a sub class of **Fragments** class. Open your main activity layout file**activity_main.xml** file and add following code. I used**RelativeLayout** as a parent element. You can remove it and use MapFragment directly.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:map="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
    android:layout_height="match_parent" android:id="@+id/map"
tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends Activity {
      // Google Map
private GoogleMap mMap;
private double latitude=23.0300;
private double longitude=72.5800;
      @Override
      protected void onCreate(Bundle savedInstanceState)
       {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_maps);

// Obtain the SupportMapFragment and get notified when the map
 is ready to be used.
final SupportMapFragment mapFragment =(SupportMapFragment)
         getSupportFragmentManager()
         .findFragmentById(R.id.map);
         mapFragment.getMapAsync(this);
}
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
// Add a marker in Sydney and move the camera
LatLng sydney = new LatLng(latitude, longitude);
mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
mMap.animateCamera(CameraUpdateFactory.zoomTo(12));
  }
```

Google Map V2 is only targeted to Google Play Service we must need to install Google Play service on emulator.

Note: Don't run your application before below steps

- **Create AVD with 4.1 or above for best performance**
- **Run AVD**
- **Download below given two Google Play's .APK from Internet**
  Google Play (com.android.vending.apk)
  Google Play Service (com.google.android.gms.apk)
- **Here are steps for installing these apks**

1. Execute the emulator **(SDK Manager.exe->Tools->Manage AVDs...->New then Start)**
2. Start the console (Windows XP), Run -> type **cmd**, and move to the **platform-tools** folder of **SDK** directory.
3. Paste the *APK* file in the 'android-sdk\tools' or 'platform-tools' folder.
4. Then type the following command.

adb install [apk file name]

Example:

adb install com.android.vending.apk

Placing a Marker

You can place a marker on the map by using following code.

```
// latitude and longitude
double latitude = ;
double longitude = ;

// create marker
MarkerOptions marker = new MarkerOptions().position(new LatLng(latitude,
longitude)).title("Hello Maps ");

// adding marker
googleMap.addMarker(marker);
```

Moving Camera to a Location with animation

You may want to move camera to a particular position. Google maps provides set of functions to achieve this.

```
CameraPosition cameraPosition = new CameraPosition.Builder().target(
               new LatLng(17.385044,78.486671)).zoom(12).build();

googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

Changing Map Type

Google provides 4 kinds of map types **Normal**, **Hybrid**, **Satellite** and**Terrain**. You can toggle to any kind of map using**googleMap.setMapType()** method.

```
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
googleMap.setMapType(GoogleMap.MAP_TYPE_NONE);
```

Showing Current Location

You can show user's current location on the map by calling **setMyLocationEnabled()**. Pass true / false to enable or disable this feature

```
googleMap.setMyLocationEnabled(true); // false to disable
```

Zooming Buttons

You can call **setZoomControlsEnabled()** function to get rid of those zooming controls on the map. By disabling these buttons map zooming functionality still work by pinching gesture.

```
googleMap.getUiSettings().setZoomControlsEnabled(false);
```

My Location Button

My location button will be used to move map to your current location. This button can be shown / hidden by calling **setMyLocationButtonEnabled()** function

```
googleMap.getUiSettings().setMyLocationButtonEnabled(true);
```

Location Based Service

Android gives your applications access to the location services supported by the device through classes in theandroid.location package. The central component of the location framework is the LocationManagersystem service, which provides APIs to determine location and bearing of the underlying device (if available).

As with other system services, you do not instantiate a Location Manager directly. Rather, you request an instance from the system by calling getSystemService(Context.LOCATION_SERVICE). The method returns a handle to a new Location Manager instance.

- Once your application has a LocationManager, your application is able to do three things:
- Query for the list of all LocationProviders for the last known user location.
- Register/unregister for periodic updates of the user's current location from a location provider (specified either by criteria or name).
- Register/unregister for a given Intent to be fired if the device comes within a given proximity (specified by radius in meters) of a given lat/long.

SELECTING A LOCATION PROVIDER:

- Depending on the device, there may be several technologies that Android can use to determine the current location. Each technology, or Location Provider, will offer different capabilities, including differences in power consumption, monetary cost, accuracy, and the ability to determine altitude, speed, or heading information.

FINDING YOUR LOCATION :

- The purpose of location-based services is to find the physical location of the device.

To access the Location Manager, request an instance of the LOCATION_SERVICE using the getSystemService method, as shown in the following snippet:

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(serviceString);
```

- Before you can use the Location Manager you need to add one or more uses-permission tags to your manifest to support access to the LBS hardware.

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

```
private long minTime = 1000;//ms
private float minDistance = 1000;//meters
TextView tv;
LocationManager manager =(LocationManager) getSystemService(LOCATION_SERVICE);
LocationListener listener =new LocationListener() {
public void onStatusChanged(String provider, int status, Bundle extras) {
// TODO Auto-generated method stub }
```

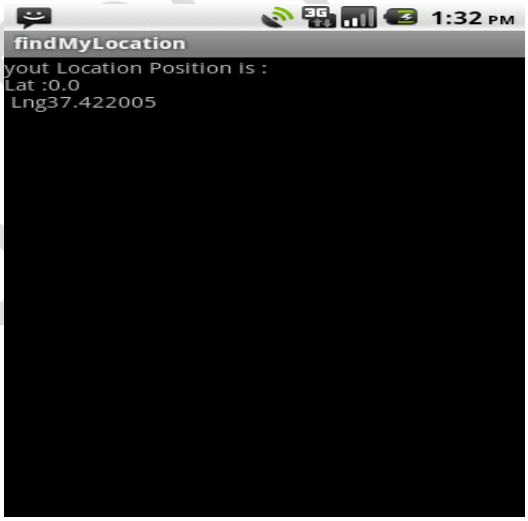
```
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub }
public void onProviderDisabled(String provider) {
// TODO Auto-generated method stub
}
public void onLocationChanged(Location location) {
latitude = location.getLatitude();
longitude =location.getLongitude();
tv.setText("Latitude is : "+latitude+" Longitude is : "+longitude);
  } };
manager.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTime, minDistance, listener);
```

Example:

- **Create new android project and add INTERNET permission in android manifest file.**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.findMyLocation"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION">
                                                        </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".findMyLocationMain"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Output:



Note : This application will run on a real device

USING THE GEOCODER:

- Geocoding lets you translate between street addresses and longitude/latitude map coordinates. This can give you a recognizable context for the locations and coordinates used in location-based services and map-based Activities.
- The Geocoder class provides access to two geocoding functions:
    - **Forward geocoding**Finds the latitude and longitude of an address
    - **Reverse geocoding**Finds the street address for a given latitude and longitude

- The results from these calls are contextualized by means of a locale (used to define your usual location and language). The following snippet shows how you set the locale when creating your Geocoder. If you don't specify a locale, it will assume your device's default.
- Geocoder geocoder = new Geocoder(getApplicationContext(), Locale.getDefault());
- Both geocoding functions return a list of Address objects. Each list can contain several possible results, up to a limit you specify when making the call.

Example :

- **Create your new android project in main.xml.**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:id="@+id/myLocationMap"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />

</LinearLayout>
```

- **In your manifest**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="com.findLocation2"
     android:versionCode="1"
     android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION">
                                                        </uses-permission>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".findLocationMain"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <uses-library android:name="com.google.android.maps" android:required="true">
                                                        </uses-library>
    </application>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION">
                                                        </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
</manifest>
```

- **Geocoding API**

- On Google Developers Console, select use Google API
- Select Google Maps Geocoding API
- Enable API
- In the left panel select Credentials
- New Credentials - > API Key -> select Browser key ->select create (create blank key).

- REVERSE Geocoding
- Link for Reverse Geocoding.
- String url = “https://maps.googleapis.com/maps/api/geocode/json?latlng=latitude,longitude&API_KEY";
- In latitude and longitude provide the co-ordinates whose street address is required
- For example - https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY.
- Json is returned of the following format.
- For address you need to fetch formatted-address key from json


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "StradaProvinciale 20",
               "short_name" : "SP20",
               "types" : [ "route" ]
            },
            {
               "long_name" : "CologneSpiaggia",
               "short_name" : "CologneSpiaggia",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Rosetodegli Abruzzi",
               "short_name" : "Rosetodegli Abruzzi",
               "types" : [ "administrative_area_level_3", "political" ]
            },
            {
               "long_name" : "ProvinciadiTeramo",
               "short_name" : "TE",
               "types" : [ "administrative_area_level_2", "political" ]
            },
            {
               "long_name" : "Abruzzo",
               "short_name" : "Abruzzo",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "Italy",
               "short_name" : "IT",
               "types" : [ "country", "political" ]
            },
         ],
"formatted_address" : "SP20, 64026 CologneSpiaggia TE, Italy",
         "geometry" : {
            "bounds" : {
               "northeast" : {
                  "lat" : 42.7160841,
                  "lng" : 13.9611359
               },
```


- **FORWARD Geocoding**
  - Link for Forward Geocoding.
  - String url = “https://maps.googleapis.com/maps/api/geocode/json?address=YOUR_ADDRESS,+CA&key=YOUR_API_KEY.
  - In address provide the address whose co-ordinates (latitude, longitude) are required

- For example - https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY
- Json is returned of the following format.
- For latitude and longitude you need to fetch lat and lng.


```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "1600",
               "short_name" : "1600",
               "types" : [ "street_number" ]
            },
            {
               "long_name" : "Amphitheatre Parkway",
               "short_name" : "Amphitheatre Pkwy",
               "types" : [ "route" ]
            },
            {
               "long_name" : "Mountain View",
               "short_name" : "Mountain View",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Santa Clara County",
               "short_name" : "Santa Clara County",
               "types" : [ "administrative_area_level_2", "political" ]
            },
            {
               "long_name" : "California",
               "short_name" : "CA",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "United States",
               "short_name" : "US",
               "types" : [ "country", "political" ]
            },
            {
               "long_name" : "94043",
               "short_name" : "94043",
               "types" : [ "postal_code" ]
            }
         ],
         "formatted_address" : "1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA",
         "geometry" : {
            "location" : {
"lat" : 37.4223607,
               "lng" : -122.0841964
            },
```


ANDROID GRAPHICS AND MULTIMEDIA

DRAWING 2D GRAPHICS

This chapter is about graphics, and therefore about the View part of the Model-View-Controller pattern. Widgets also contain Controller code, which is good design because it keeps together all of the code relevant to a behavior and its representation on the screen.
Concentrating on graphics, then, we can break the tasks of this chapter into two essentialparts: finding space on the screen and drawing in that space. The first task is known as ***layout***. A leaf widget can assert its space needs by defining an onMeasure method that the Android framework will call at the right time. The second task, actually rendering the widget, is handled by the widget's onDraw method.

Canvas Drawing

Now that we've explored how widgets allocate the space on the screen in which theydraw themselves, we can turn to coding some widgets that actually do some drawing.
Eventually, when that event is processed, the framework calls the draw method at thetop of the view tree. This time the call is propagated preorder, with each view drawingitself before it calls its children. This means that leaf views are drawn after their parents,which are, in turn, drawn after their parents. Views that are lower in the tree appear tobe drawn on top of those nearer the root of the tree.
The draw method calls onDraw, which a subclass overrides to implement its customrendering. When your widget's onDraw method is called, it must render itself accordingto the current application state and return.
There are two essential rules:

• Drawing code should be inside the onDraw method. Your widget should draw itself completely, reflecting the program's current state, when onDraw is invoked.

• A widget should draw itself as quickly as possible when onDraw is invoked. All the state you need to draw should be cached and ready for use at drawing time.

The Android UI framework uses four main classes in drawing. If you are going to implement custom widgets and do your own drawing, you will want to become veryfamiliar with them:

Canvas (a subclass of android.graphics.Canvas)

The canvas has no complete analog in real-life materials. You might think of it asa complex easel that can orient, bend, and even crumple the paper on which you are drawing in interesting ways. It maintains the clip rectangle, the stencil throughwhich you paint. It can also scale drawings as they are drawn, like a photographicenlarger. It can even perform other transformations for which material analogs aremore difficult to find: mapping colors and drawing text along paths.

Paint (a subclass of android.graphics.Paint)

This is the medium with which you will draw. It controls the color, transparency,and brush size for objects painted on the canvas. It also controls font, size, andstyle when drawing text.

Bitmap (a subclass of android.graphics.Bitmap)

This is the paper you are drawing on. It holds the actual pixels that you draw.

Drawables (likely a subclass of android.graphics.drawable.Drawable)

This is the thing you want to draw: a rectangle or image. Although not all of thethings that you draw are Drawables (text, for instance, is not), many, especially themore complex ones, are.

Working with Paint

Before we can do any drawing, we need to construct a Paint object. The Paint objectwill allow us to define the color, size of the stroke, and style of the stroke used whendrawing. We can therefore think of the Paint as both paint and brush.

```
Paint paint = new Paint(); //Initialize paint class
paint.setColor(Color.GREEN); //Set Color
paint.setStyle(Paint.Style.STROKE); //Set Style
paint.setStrokeWidth(10); //Set Storack
```

The foregoing snippet of code creates a Paint object, sets its color to be green, definesthat we want to draw the outline of shapes rather than fill them in, and sets the width ofthat stroke to be 10 pixels.

Color

Using the setColor method on the Paint object, we can pass in a Color object. TheColor class defines a series of colors represented as 32bit integers as constants:

- Color.BLACK
- Color.BLUE
- Color.RED

```
Paint paint = new Paint();
paint.setColor(Color.GREEN);
```

We can also construct a specific color by calling the static method Color.argb, passingin a value between 0 and 255 for alpha, red, green, and blue. This method returns a32bit integer representing that color that we then pass to setColor.

```
Paint paint = new Paint();
int myColor = Color.argb(255,128,64,32);
paint.setColor(myColor);
```

We can actually skip the color creation step completely if we are defining the exactvalues:

```
Paint paint = new Paint();
paint.setARGB(255,128,64,32);
```

Style

When defining the style of the paint through the setStyle method, we are determiningwhether the shapes drawn will be filled or simply outlined. The possible styles aredefined as constants in the Paint.Style class.

- ***Paint.Style.STROKE*:** Only draw the outline of the shapes

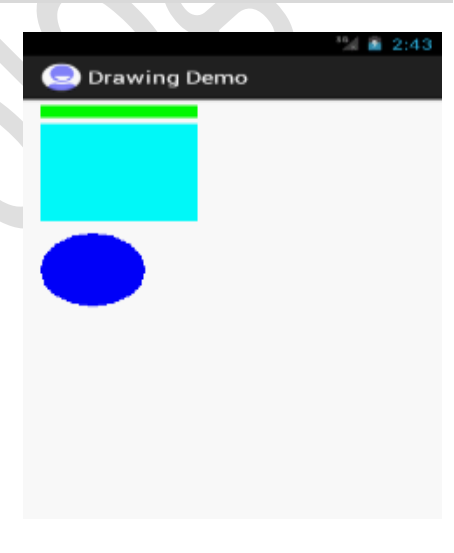
- ***Paint.Style.FILL*:** Only fill the shapes
- ***Paint.Style.FILL_AND_STROKE*:** Fill and draw the outline of the shapes

Drawing Shapes

The Canvas class defines several drawing methods. Let's go through a couple of these.

Example of different shapes.

```
publicclass DrawingDemo extends Activity {
	@Override
	publicvoid onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(new myLayout(this));
	}
	publicclass myLayout extends View {
		public myLayout(Context context) {
			super(context);
			// TODO Auto-generated constructor stub
		}
		Paint paint = new Paint();
		@Override
		protectedvoid onDraw(Canvas canvas) {
			// TODO Auto-generated method
stub
			super.onDraw(canvas);
			canvas.drawColor(Color.WHITE);
			// Paint Line
			paint.setStrokeWidth(10);
			paint.setColor(Color.GREEN);
		canvas.drawLine(10, 10, 100, 10,
paint);
			// Paint Ract
			paint.setStrokeWidth(5);
			paint.setColor(Color.CYAN);
			canvas.drawRect(10, 20, 100, 100, paint);
			// Paint Circle
			paint.setStrokeWidth(2);
			paint.setColor(Color.BLUE);
			canvas.drawCircle(40, 140, 30, paint);
		}
	}
}
```



Drawing text

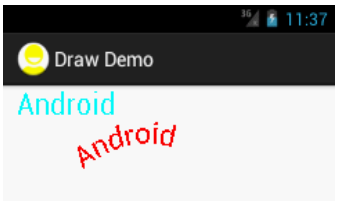
The most important Canvas methods are those used to draw text. Although some Canvasfunctionality is duplicated in other places, text-rendering capabilities are not.In order to put text in your widget, you will have to use the Canvas.Canvas methods for rendering text come in pairs: three sets of two signatures.

A pair of text drawing methods

public void drawText(String text, float x, float y, Paint paint)public void drawText(char[] text, int index, int count, float x, float y, Paint paint)

There are several pairs of methods. In each pair, the first of the two methods in the pairuses String, and the second uses three parameters to describe the text: an array ofchar, the index indicating the first character in that array to be drawn, and the numberof total characters in the text to be rendered. In some cases, there are additional convenience methods.

Example of Three ways of drawing text




```
public class DrawDemo extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(new DrawLayout(this));
	}
	public class DrawLayout extends View {
		public DrawLayout(Context context) {
			super(context);
			// TODO Auto-generated constructor stub
		}
		Paint paint = new Paint();
		Path path = new Path();
		@Override
		protected void onDraw(Canvas canvas) {
			// TODO Auto-generated method stub
			super.onDraw(canvas);
			// Text Writing
			paint.setColor(Color.CYAN);
			paint.setTextSize(20);
			paint.setStrokeWidth(2);
			canvas.drawText("Android", 10, 20, paint);

			// Arc Text
			path.addArc(new RectF(10, 40, 200, 300), 240, 90);
			canvas.drawTextOnPath("Android", path, 0, 0, paint);
		}
	}
}
```

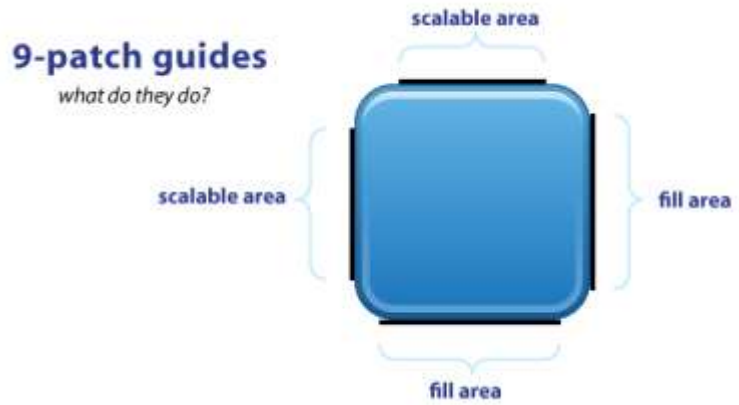
Bitmaps

The Bitmap is the last member of the four essentials for drawing: something to draw (aString, Rect, etc.), Paint with which to draw, a Canvas on which to draw, and theBitmap to hold the bits. A common use for a Bitmap is as a way to cache a drawing that is time-consuming todraw but unlikely to change frequently.

```
public class bitmap extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new layoutD(this));
    }
    public class layoutD extends View
    {
        public layoutD(Context context) {
            super(context);
        }
        public void onDraw(Canvas canvas)
        {
            Bitmap bitimage1 = BitmapFactory.decodeResource
                               (getResources(), R.drawable.icon2);
            Bitmap bitimage2 = BitmapFactory.decodeResource
                               (getResources(), R.drawable.myimg);
            canvas.drawColor(Color.WHITE);
            canvas.drawBitmap(bitimage1, 0, 0, null);
            canvas.drawBitmap(bitimage2, 0, 120, null);
        }
    }
}
```

9-patch for Android UI

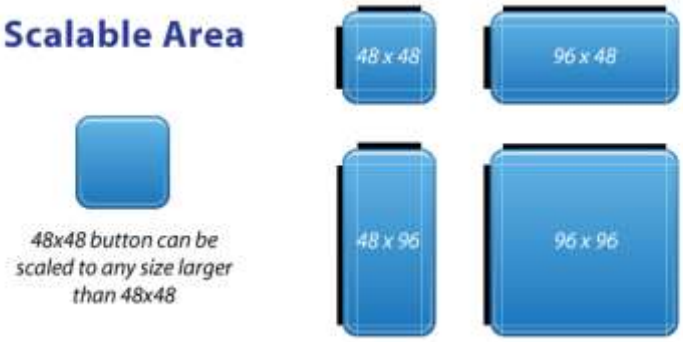
Basically, 9-patch uses png transparency to do an advanced form of 9-slice or scale9. The guides are straight, 1-pixel black lines drawn on the edge of your image that define the scaling and fill of your image. By naming your image file *name.9.png*, Android will recognize the *9.png* format and use the black guides to scale and fill your bitmaps.



can see, you have guides on each side of your image. The TOP and LEFT guides are for scaling your image (i.e. 9-slice), while the RIGHT and BOTTOM guides define the fill area.

The black guide lines are cut-off/removed from your image – they won't show in the app. Guides must only be one pixel wide, so if you want a 48×48 button, your png will actually be 50×50. Anything thicker than one pixel will remain part of your image. (My examples have 4-pixel wide guides for better visibility. They should really be only 1-pixel).
Your guides must be solid black (#000000). Even a slight difference in color (#000001) or alpha will cause it to fail and stretch normally. This failure won't be obvious either*, it fails silently! Yes. Really.

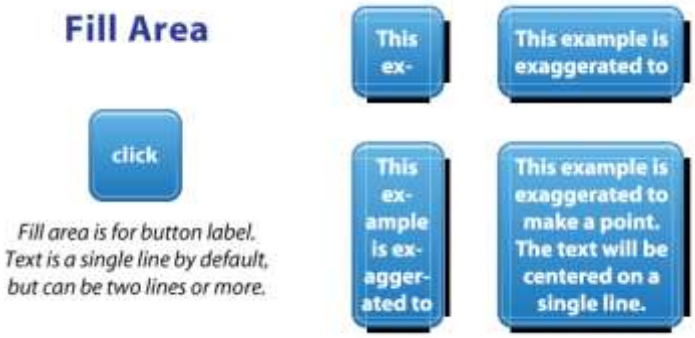
Scalable Area



The TOP and LEFT guides are used to define the scalable portion of your image – LEFT for scaling height, TOP for scaling width. Using a button image as an example, this means the button can stretch horizontally and vertically within the black portion and everything else, such as the corners, will remain the same size. The allows you to have buttons that can scale to any size and maintain a uniform look.

It's important to note that 9-patch images don't scale down – they only scale up. So it's best to start as small as possible.

Fill Area

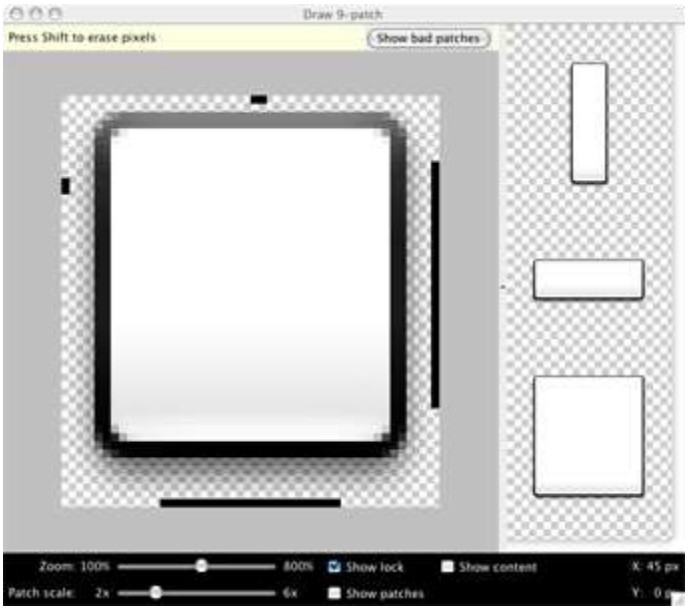


Fill area guides are *optional* and provide a way define the area for stuff like your text label. Fill determines how much room there is within your image to place text, or an icon, or other things. 9-patch isn't just for buttons, it works for background images as well.

The above button & label example is exaggerated simply to explain the idea of fill – the label isn't completely accurate. To be honest, I haven't experienced how Android does multi-line labels since a button label is usually a single row of text.

Draw 9-patch Images

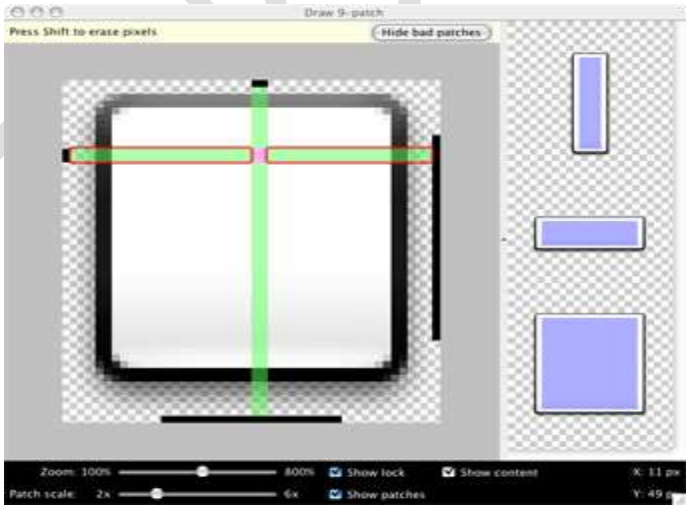
The Draw 9-patch tool allows you to easily create a NinePatch graphic using a WYSIWYG editor.



Here's a quick guide to create a Nine-patch graphic using the Draw 9-patch tool. You'll need the PNG image with which you'd like to create a NinePatch.

1. From a terminal, launch the `draw9patch` application from your SDK `/tools` directory.
2. Drag your PNG image into the Draw 9-patch window (or **File**>**Open 9-patch...** to locate the file). Your workspace will now open.
   The left pane is your drawing area, in which you can edit the lines for the stretchable patches and content area. The right pane is the preview area, where you can preview your graphic when stretched.
3. Click within the 1-pixel perimeter to draw the lines that define the stretchable patches and (optional) content area. Right-click (or hold Shift and click, on Mac) to erase previously drawn lines.
4. When done, select **File**>**Save 9-patch...**
   Your image will be saved with the `.9.png` file name.

Note: A normal PNG file (`*.png`) will be loaded with an empty one-pixel border added around the image, in which you can draw the stretchable patches and content area. A previously saved 9-patch file (`*.9.png`) will be loaded as-is, with no drawing area added, because it already exists.



Optional controls include:

- **Zoom**: Adjust the zoom level of the graphic in the drawing area.
- **Patch scale**: Adjust the scale of the images in the preview area.
- **Show lock**: Visualize the non-drawable area of the graphic on mouse-over.
- **Show patches**: Preview the stretchable patches in the drawing area (pink is a stretchable patch).
- **Show content**: Highlight the content area in the preview images (purple is the area in which content is allowed).
- **Show bad patches**: Adds a red border around patch areas that may produce artifacts in the graphic when stretched. Visual coherence of your stretched image will be maintained if you eliminate all bad patches.

2D Animation

Animation is a process by which an object on a screen changes its color, position, size, ororientation over time. Android supports three types of animation:

frame-by-frame animation, which occurs when a series of frames is drawn one after the other at regular intervals;
layoutanimation, in which you animate views inside a container view such as lists and tables;
view animation, in which you animate any general-purpose view. The latter two types fall into the category of *tweening animation*, which involves the drawings in between the key drawings.

Frame-by-Frame Animation

Frame-by-frame animation is the simple process of showing a series of images in succession at quick intervals so that the end effect is that of an object moving. This is how movie or film projectors work. We'll explore an example in which we'll design an image and save that image as a number of distinct images, where each one differs from the other slightly. Then we will take the collection of those images and run them through the sample code to simulate animation.

Planning for Frame-by-Frame Animation

Before you start writing code, you first need to plan the animation sequence using a series of drawings. Give the image a base name of colored-ball. Then you can store these images
in the /res/drawable subdirectory so that you can access them using their resource IDs. The name of each image will have the pattern colored-images-N, where N is the digit representing the image number.
The primary area in this activity is used by the animation view. We have included a button to start and stop the animation to observe its behavior.

Creating the Activity

Start by creating the basic XML layout file for our test-animation activity screen.

XML Layout File for the Animation


```
<RelativeLayout xmlns:android="http://schemas.android.com
/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<Button
        android:id="@+id/btnStart"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Start" />
<ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnStart"
        android:layout_alignParentLeft="true"
```

```
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true" />
</RelativeLayout>
```

In Android, you accomplish frame-by-frame animation through a class in thegraphics package called AnimationDrawable. This class can take a list of Drawable resources (like images) and render them at specified intervals. This class is really a thin wrapper around the animation support provided by the basic Drawable class.

The Drawable class enables animation by asking its container or view to invoke a Runnable class that essentially redraws the Drawable using a different set of parameters. Note that youdon't need to know these internal implementation details to use the AnimationDrawable class.But if your needs are more complex, you can look at the AnimationDrawable source code forguidance in writing your own animation protocols.

To make use of the AnimationDrawable class, start with a set of Drawable resources placed in the /res/drawable subdirectory. You will then construct an XML file that defines the list of frames.

XML File Defining the List of Frames to be Animated

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list android:oneshot="false"
    xmlns:android="http://schemas.android.com/apk/res/android">
<item android:drawable="@drawable/mickey_1" android:duration="200"/>
<item android:drawable="@drawable/mickey_2" android:duration="200"/>
<item android:drawable="@drawable/mickey_3" android:duration="200"/>
<item android:drawable="@drawable/mickey_4" android:duration="200"/>
<item android:drawable="@drawable/mickey_5" android:duration="200"/>
</animation-list>
Complete Code for the Frame-by-Frame Animation
public class AnimationDemo extends Activity {
      private Button btnP;
      private ImageView imgAnm;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_animation_demo);
            btnP = (Button) findViewById(R.id.btnStart);
      imgAnm = (ImageView) findViewById(R.id.imageView1); // Set Image view
            btnP.setOnClickListener(new OnClickListener() {
                  public void onClick(View arg0) {
            // Set animation Property
            imgAnm.setVisibility(ImageView.VISIBLE);
            imgAnm.setBackgroundResource(R.anim.animation);
      AnimationDrawable ani = (AnimationDrawable) imgAnm.getBackground();
      if (ani.isRunning()) {
            ani.stop(); //Stop if Started
            btnP.setText("Start");
            } else {
            ani.start(); // Start if Stoped
            btnP.setText("Stop");
            }
                  }
            });   }
```

Layout Animation

frame-by-frame animation is a quick and dirty way to add visual effects toyour Android applications. Layout animation is almost as simple. You'll use layout animation with the List View and GridView, which are the two most commonly, used controls in Android.

Layout animation works by applying ***tweening***principles to each view that is part of the layout being animated. Tweening is a process in which a number of the view's properties are changed at regular intervals. Every view in Android has a matrix that maps the view to the screen. By changing this matrix in a number of ways, you can accomplish scaling, rotation, and movement (translation) of the view. By changing the transparency of the view from 0 to 1, for example, you can accomplish what is called an ***alpha***animation.

Tweening Animation Types

Before we design the test harness to apply the various tweening animations, we'll give yousome detail on the basic types of tweening animation:

- ***Scale animation*:** You use this type of animation to make a view smaller or larger either on the x axis or on the y axis. You can also specify the pivot point around which you want the animation to take place.
- ***Rotate animation*:** You use this to rotate a view around a pivot point by a certain number of degrees.
- ***Translate animation*:** You use this to move a view along the x axis or the y axis.
- ***Alpha animation*:** You use this to change the transparency of a view.

All of the parameter values associated with these animations have a formand a to flavorbecause you must specify the starting values and ending values for when the animation starts and ends. Each animation also allows duration as an argument and a time interpolator as an argument. We'll cover interpolators at theend of this section on layout animation, but for now, know that interpolators determine the rate of change of the animated argument during animation.

Animating the List View example

XML Layout File Defining the ListView

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<Button android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btn1"
        android:text="Start Animation"/>
<ListView android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/lst1"
        android:persistentDrawingCache="animation|scrolling"
        android:layout_below="@id/btn1">
</ListView>
</RelativeLayout>

Code for the Layout-Animation Activity

Defining Scale Animation in an XML File (These animation-definition files
reside in the /res/anim subdirectory.)

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator" >
<scale
        android:duration="1000"
        android:fromXScale="1"
        android:fromYScale="0.1"
        android:pivotX="50%"
        android:pivotY="50%"
```

```
        android:startOffset="100"
        android:toXScale="1"
        android:toYScale="1.0" >
</scale>
</set>
```

Definition for a Layout-Controller XML File

```
<?xml version="1.0" encoding="utf-8"?>
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
    android:delay="30%"
    android:animationOrder="reverse"
    android:animation="@anim/scale">
</layoutAnimation>
```

Complet Code For Activity Class

```
publicclass LayoutAnimationDemo extends Activity {
      private ListView lst;
      private Button btn;
      private String[] items = { "Honda", "Suzuki", "Maruti", "Hundai",
                  "Mercedes", "Scoda", "Rols Royas", "Audi", "Bugati" };
      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_layout_animation_demo);
            lst = (ListView) findViewById(R.id.lst1);
            btn = (Button) findViewById(R.id.btn1);
            lst.setAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, items));
            btn.setOnClickListener(new OnClickListener() {
                  publicvoid onClick(View arg0) {
                        // Start Animation
                        animateView();
                  }
                  privatevoid animateView() {
                        // TODO Auto-generated method stub
                        lst.startAnimation(new myAnimation());
                  }
            });
      }

publicclass myAnimation extends Animation {
            public myAnimation() {
            }
            @Override
      publicvoid initialize(int width, int height, int parentWidth,
                        int parentHeight) {
      // TODO Auto-generated method stub
      super.initialize(width, height, parentWidth, parentHeight);
                  setDuration(2500);
                  setFillAfter(true);
                  setInterpolator(new LinearInterpolator());
            }
      @Override
```

```
      protectedvoid applyTransformation(float interpolatedTime,
                        Transformation t) {
                  // TODO Auto-generated method stub
                  super.applyTransformation(interpolatedTime, t);
                  final Matrix matrix = t.getMatrix();
                  matrix.setScale(interpolatedTime, interpolatedTime);
            }
      }
}
```

View Animation

Now that you're familiar with frame-by-frame animation and layout animation, you're ready to tackle view animation—the most complex of the three animation types. View animation allows you to animate any arbitrary view by manipulating the transformation matrix that is in place for displaying the view.

Understanding View Animation

When a view is displayed on a presentation surface in Android, it goes through a transformation matrix. In graphics applications, you use transformation matrices to transform a view in some way. The process involves taking the input set of pixel coordinates and color combinations and translating them into a new set of pixel coordinates and color combinations. At the end of a transformation, you will see an altered picture in terms of size, position, orientation, or color.

You can achieve all of these transformations mathematically by taking the input set of coordinates and multiplying them in some manner using a transformation matrix to arrive at a new set of coordinates. By changing the transformation matrix, you can impact how a view will look. A matrix that *doesn't* change the view when you multiply by it is called an *identity matrix*. You typically start with an identity matrix and apply a series of transformations involving size, position, and orientation. You then take the final matrix and use that matrix to draw the view.

Android exposes the transformation matrix for a view by allowing you to register an animation object with that view. The animation object will have a callback that lets it obtain the current matrix for a view and change it in some manner to arrive at a new view.

The view-animation activity

XML Layout File for the View-Animation Activity

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#3399CC"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:textStyle="bold"
    />
    <Button
        android:id="@+id/btn_animate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Start Animation"
    />
    <ListView
        android:id="@+id/list_view_id"
        android:persistentDrawingCache="animation|scrolling"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    />
</LinearLayout>
```

Code for the View-Animation Activity

```
import android.app.Activity;

public class twoDthird extends Activity {
    private ListView LV;
    private Button btn;
    private String[] p = {"Android","iPhone","Java","Php",".Net","Testing"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        LV = (ListView)findViewById(R.id.list_view_id);
        btn = (Button)findViewById(R.id.btn_animate);
        LV.setAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,p));
        btn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                animateListView();
            }
        });
    }
    private void animateListView()
    {
        ListView lv = (ListView)this.findViewById(R.id.list_view_id);
        lv.startAnimation(new viewtwoD());
    }
}
```

New class for animation

```
public class viewtwoD extends Animation {

    public viewtwoD(){}
    @Override
    public void initialize(int width, int height, int parentWidth,
    int parentHeight)
    {
    super.initialize(width, height, parentWidth, parentHeight);
    setDuration(2500);
    setFillAfter(true);
    setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t)
    {
    final Matrix matrix = t.getMatrix();
    matrix.setScale(interpolatedTime, interpolatedTime);
    }
}
```

http://stuffthathappens.com/blog/2008/11/13/android-animation-101/

MULTIMEDIA IN ANDROID

Play Audio on Android

Any smartphone worth its name these days has audio playback capabilities on par withdedicated portable media devices or MP3 players. Of course, Android-based devicesare no different. These capabilities allow for the building of music player, audio book,podcast, or just about any other type of application that is centered around audioplayback.

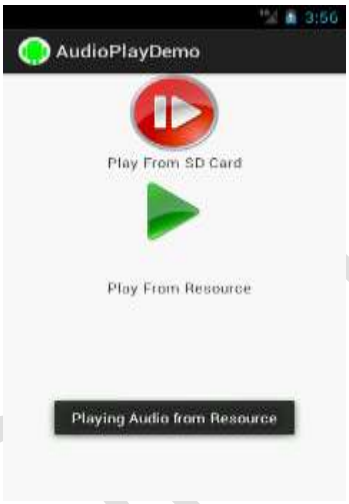
Supported Audio Formats

1. *AAC*: Advanced Audio Coding codec (as well as both profiles of HEAAC, High Efficiency AAC), .m4a (audio/m4a) or.3gp (audio/3gpp) files.
2. *MP3*: MPEG-1 Audio Layer 3, .mp3 (audio/mp3) files. MP3, probably the most widely used audio codec, is supported.
3. *AMR*: Adaptive Multi-Rate codec (both AMR Narrowband, AMR-NB, and AMR Wideband, AMR-WB), .3gp (audio/3gpp) or .amr (audio/amr) files.
4. *Ogg*: Ogg Vorbis, .ogg (application/ogg) files. Ogg Vorbis is an open source, patent-free audio codec with quality that is comparable to commercial and patent-encumbered codecs such as MP3 and AAC.
5. *PCM*: Pulse Code Modulation commonly used in WAVE or WAV files (Waveform Audio Format), .wav (audio/x-wav) files. PCM is the technique used for storing audio on computers and other digital audio devices.

- **Custom Audio-Playing Application**
- Android includes a MediaPlayer class. This class is used for the playback and control of both audio and video. Right now we'll just be using the audio playback capabilities.
- The simplest MediaPlayer example is to play back an audio file that is packaged with the application itself. In order to do that, an audio file should be placed within the application's raw resources. To do this using the Android Developer Tools on Eclipse, we need to create a new folder in our Project's res folder called raw as illustrated in fig.

Built-in Audio Player via an Intent

- The generic **android.content.Intent.ACTION_VIEW** intent with the data set to a Uri to the audio file and the MIME type specified allows Android to pick the appropriate application for playback. This should be the Music application, but the user may be presented with other options if he or she has other audio playback software installed.



Start a new project named *audioDemo*. Open the res/layout/main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ImageButton
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:background="@null"
        android:contentDescription="play"
        android:src="@drawable/play_r" />
```

```
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_centerHorizontal="true"
        android:text="Play From SD Card" />
<ImageButton
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/textView1"
        android:background="@null"
        android:contentDescription="play"
        android:src="@drawable/play_g" />
<TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_centerVertical="true"
        android:text="Play From Resource" />
</RelativeLayout>
```

In your project *audioDemo*. java file

```
package com.example.audioplaydemo;

publicclass AudioPlayDemo extends Activity {
      private ImageButton btn, btnM;
      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_audio_play_demo);
            btn = (ImageButton) findViewById(R.id.button1);
            btnM = (ImageButton) findViewById(R.id.button2);
            btn.setOnClickListener(new OnClickListener() {

            publicvoid onClick(View arg0) {
                        // Select Intent to Set Action
            Intent i = new Intent(android.content.Intent.ACTION_VIEW);
            File sdcard = Environment.getExternalStorageDirectory();//Get File
            File audioFile = new File(sdcard.getPath()
                                    + "/koi_mil_gaya.mp3");
      // Set Data and Type
            i.setDataAndType(Uri.fromFile(audioFile), "audio/mp3");
            startActivity(i); //Start Playing
            Toast.makeText(AudioPlayDemo.this,
                                    "Playing Audio from SD Card",
Toast.LENGTH_LONG).show();
                  }
            });
            btnM.setOnClickListener(new OnClickListener() {

                  publicvoid onClick(View arg0) {
            // TODO Auto-generated method stub
```

```
            MediaPlayer mediaplay = MediaPlayer.create(AudioPlayDemo.this,
                                    R.raw.koi_mil_gaya);
            mediaplay.start();//Start Playing
            Toast.makeText(AudioPlayDemo.this,
                        "Playing Audio from Resource",
Toast.LENGTH_LONG).show();
                  }
            });
      }
}
```

Introduction to Video

- Continuing on our journey through Android's media capabilities, we'll now turn our attention to video. In this chapter, we'll explore the various means we can use for video playback on Android as well as what formats are supported.
- **Video Playback**
- Technically, some mobile phones have had video capabilities previous to 2004. In reality, though, video on mobile phones didn't really take off in the US until the introduction of the iPhone in 2007. Since then, every smart phone worth its name has supported video playback, if not video capture.

Supported Formats

- Android supports playing back a variety of video formats and the types it can play back is slowly increasing, it certainly doesn't cover the wide range of video formats available.
- In general Android's support is consistent with other mobile phones. It supports the 3GP (.3gp) and MPEG-4 (.mp4) file formats. 3GP is a video standard derived from MPEG-4 specifically for use by mobile devices.
- Android also supports MPEG-4 Simple Profile in 3GP files (.3gp) as well as H.264.

Video Playback Using an Intent

Start a new project named *videoDemo*. Open the res/layout/main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

<Button
        android:id="@+id/btn"
        android:layout_width="100dp"
        android:layout_height="100dp"
```

```
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="102dp"
        android:background="@drawable/play"
        android:paddingTop="100dp" />

</RelativeLayout>
```

In your project *videoDemo*. java file

```
package com.videodemo;
public class Main extends Activity implements OnClickListener {
      Button btn;
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            btn = (Button) findViewById(R.id.btn);
            btn.setOnClickListener(this);
      }
      @Override
      public void onClick(View arg0) {
            // Set Intent for Playing video
            Intent i = new Intent(android.content.Intent.ACTION_VIEW);
            Uri data = Uri.parse(Environment.getExternalStorageDirectory()
                        .getPath() + "/tom_jerry.3gp");
            i.setDataAndType(data, "video/3gp");// Set data and type
            Toast.makeText(getApplicationContext(), data.toString(),
100).show();
            startActivity(i);
      }

}
```

WORKING IN BACKGROUND

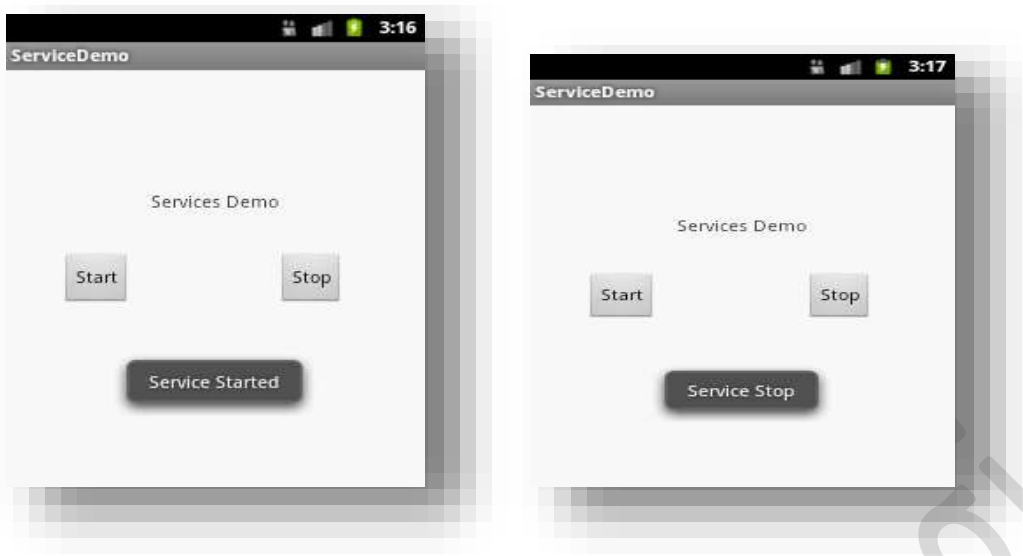
Android supports the concept of services. Services are components that run in the background, without a user interface. You can think of these components as Windows services or Unix services. Similar to these types of services, Android services are always available but don't have to be actively doing something. Android supports two types of services: ***local services***and ***remote services***. A local service is a service that is not accessible from other applications running on the device. Generally, these types of services simply support the application that is hosting the service. A remote service is accessible from other applications in addition to the application hosting the service. Remote services define themselves to clients using Android Interface Definition Language (AIDL).

Creating a Simple Service

To build a service, you extend the abstract class android.app.Service and put a serviceconfiguration entry in your application's manifest file.

A Simple Android Service Definition

Simple Example of service



Create new apps. In your layouts/main.xml file.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="108dp"
        android:text="Services Demo" />

<Button
        android:id="@+id/btnStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_marginRight="14dp"
        android:layout_marginTop="37dp"
        android:layout_toLeftOf="@+id/textView1"
        android:text="Start" />

<Button
        android:id="@+id/btnStop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnStart"
        android:layout_alignBottom="@+id/btnStart"
        android:layout_toRightOf="@+id/textView1"
        android:text="Stop" />

</RelativeLayout>
```

In Main Activity call service fragment

```
public class MainActivity extends AppCompatActivity {

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ServiceFragment fragment =new ServiceFragment();
        FragmentTransaction ft = getFragmentManager().beginTransaction();
        ft.replace(android.R.id.content, fragment);
        ft.commit();
    }
}
```

AndiService.java

```
public class AndiService extends Service {
  @Nullable
  @Override
public IBinder onBind(Intent intent) {
return null;
  }
  @Override
public void onCreate() {
super.onCreate();
    Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
  }
  @Override
public void onDestroy() {
super.onDestroy();
    Toast.makeText(this, "Service Stopped", Toast.LENGTH_LONG).show();
  }
}
```

In Manifest file -> Application tab, register service
<service android:name=".AndiService"></service>

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tops.arpan.servicedemo"
    android:versionCode="1"
    android:versionName="1.0" >

<uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="15" />

<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
<activity
            android:name=".ServiceDemo"
```

```
            android:label="@string/title_activity_service_demo" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<service android:name="AndiService" android:label="MyService"></service>
</application>

</manifest>
```

ServiceDemo.java

```
public class ServiceFragment extends Fragment {
   Button Start,Stop;
@Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view = inflater.inflate(R.layout.fragment_, container, false);
      Start = (Button) view.findViewById(R.id.button);
      Stop  =(Button)view.findViewById(R.id.button2);
      Start.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            getActivity().startService(new Intent(getActivity(), AndiService.class));
         }
      });
      Stop.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            getActivity().stopService(new Intent(getActivity(), AndiService.class));
    }
      });
      return view;
   }
}
```

Android Notification Services:

- A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer.
- To create a status bar notification, we'll need to use
  two classes: **Notification** and **NotificationManager**.
- **Notification**
  Defines the properties of the status bar notification

like the icon to display, the test to display when the notification first appears on the status bar and the time to display.

- **NotificationManager**

  NotificationManager is an android system service that executes and manages all notifications. Hence we cannot create an instance of the NotificationManager but we can retrieve a reference to it by calling the getSystemService() method.

- **In Main Activity**

```
public class MainActivity extends AppCompatActivity {
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      NotificationFragment fragment =new NotificationFragment();
      FragmentTransaction ft = getFragmentManager().beginTransaction();
      ft.replace(android.R.id.content, fragment);
      ft.commit();
   }
}
```

In Fragment

```
public class NotificationFragment extends Fragment {
private Notification notification;
private NotificationManager mNM;
   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                  Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view= inflater.inflate(R.layout.fragment_notification, container, false);
mNM=(NotificationManager)getActivity().getSystemService(Context.NOTIFICATION_SERVICE);
      NotificationCompat.Builder builder = new NotificationCompat.Builder(getActivity());
      Intent notifyIntent = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse("http://www.tops-
int.com"));
PendingIntent contentIntent= PendingIntent.getActivity(getActivity(), 0, notifyIntent, 0);

notification = builder.setContentIntent(contentIntent)
            .setSmallIcon(R.mipmap.ic_launcher).setTicker("NewNotification")
.setWhen(System.currentTimeMillis())
            .setAutoCancel(true).setContentTitle("Notification Title")
            .setContentText("Notification called").build();

mNM.notify(1, notification);
return view;}}
```

Broadcast Receivers

As a system-level message-passing mechanism, Intents are capable of sending structured messages across process boundaries. So far you've looked at using Intents to start new application components, but they can also be used to broadcast messages anonymously *between* components with the sendBroadcast method. You can implement Broadcast Receivers to listen for, and respond to, these broadcast Intents within your applications.

Broadcast Intents are used to notify listeners of system or application events, extending the eventdriven programming model between applications.

Broadcasting Intents helps make your application more open; by broadcasting an event using an Intent, you let yourself and third-party developers react to events without having to modify your original application. Within your applications, you can listen for Broadcast Intents to replace or enhance native (or third-party) applications or react to system changes and application events.

For example, by listening for the incoming call broadcast, you can modify the ringtone or volume based on the caller. Android uses Broadcast Intents extensively to broadcast system events like battery-charging levels, network connections, and incoming calls.

Example of checking battery status using Broadcast Receiver

Activity_broadcast_receiver.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/progressBar1"
        android:layout_centerHorizontal="true"
android:text="BroadCast Receiver Example"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

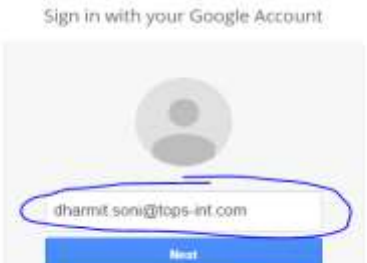
In Main Activity

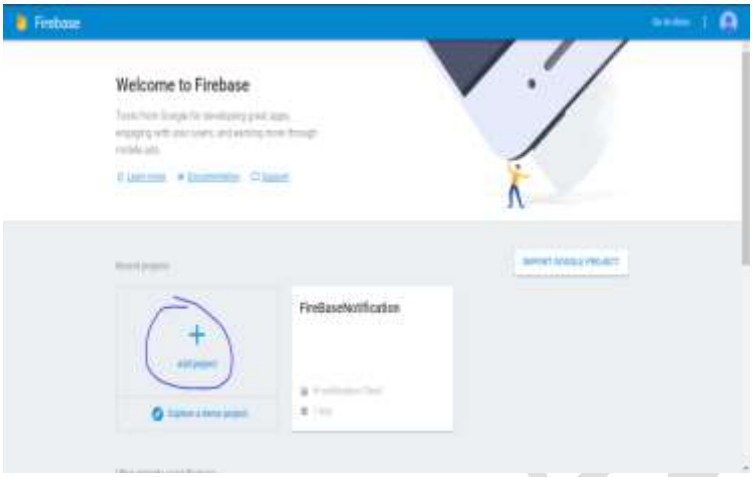
```
public class MainActivity extends AppCompatActivity {
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        RecFragment fragment =new RecFragment();
        FragmentTransaction ft = getFragmentManager().beginTransaction();
        ft.replace(android.R.id.content, fragment);
        ft.commit();
    } } }
```

Firebase:

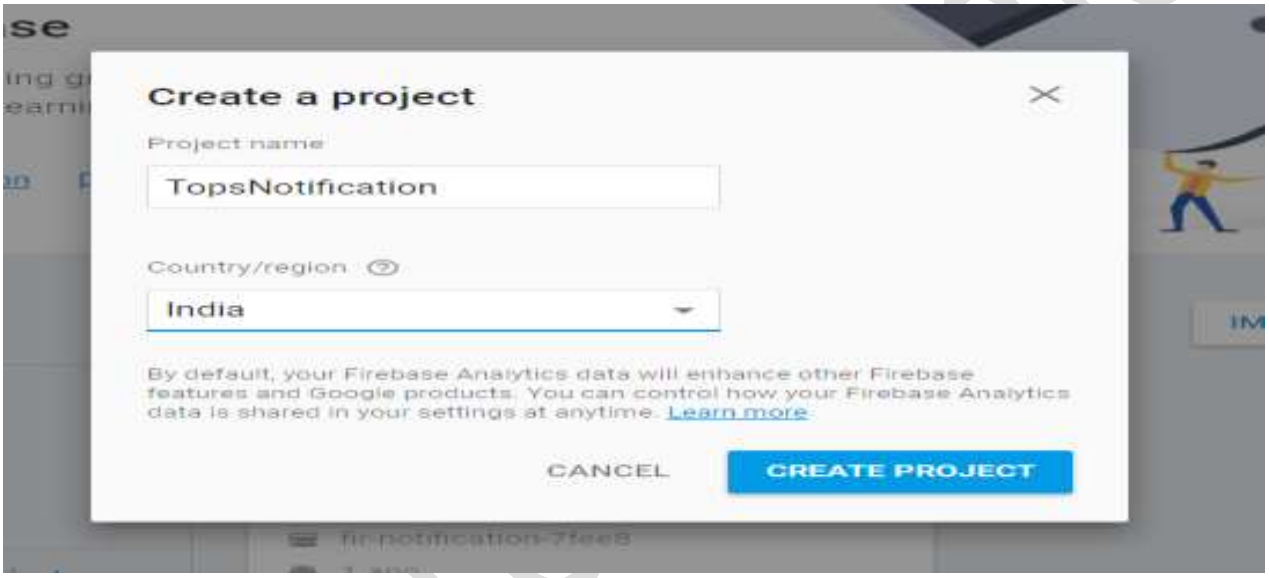
Android firebase push notification

1. login into firebase console with your google account.
2. create new project of the firebase.

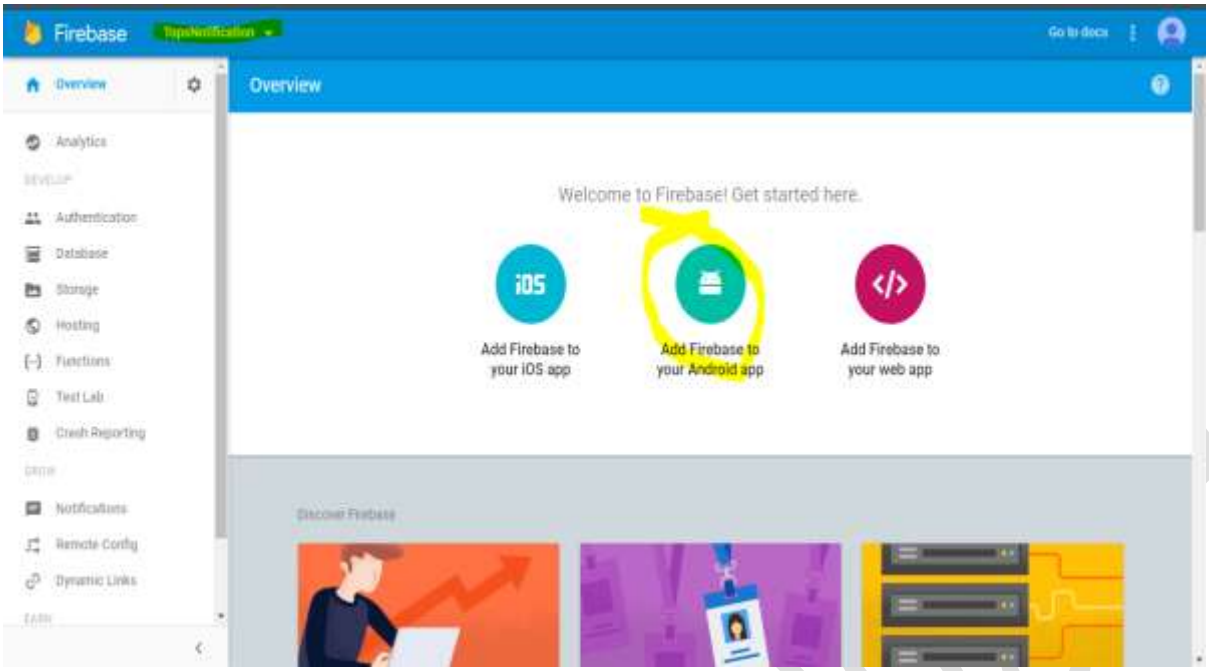




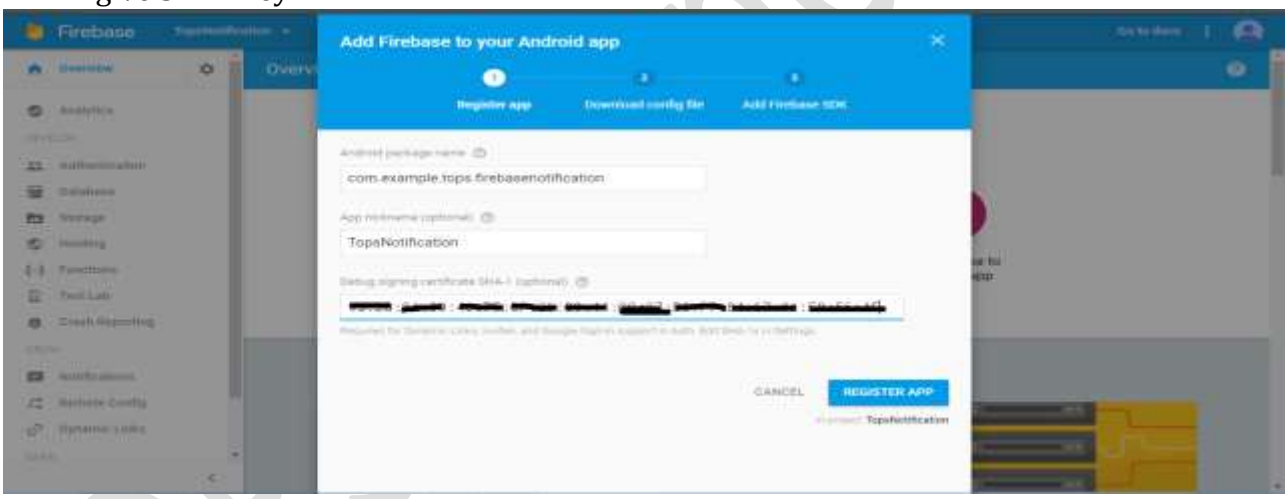
3. give name and select country.



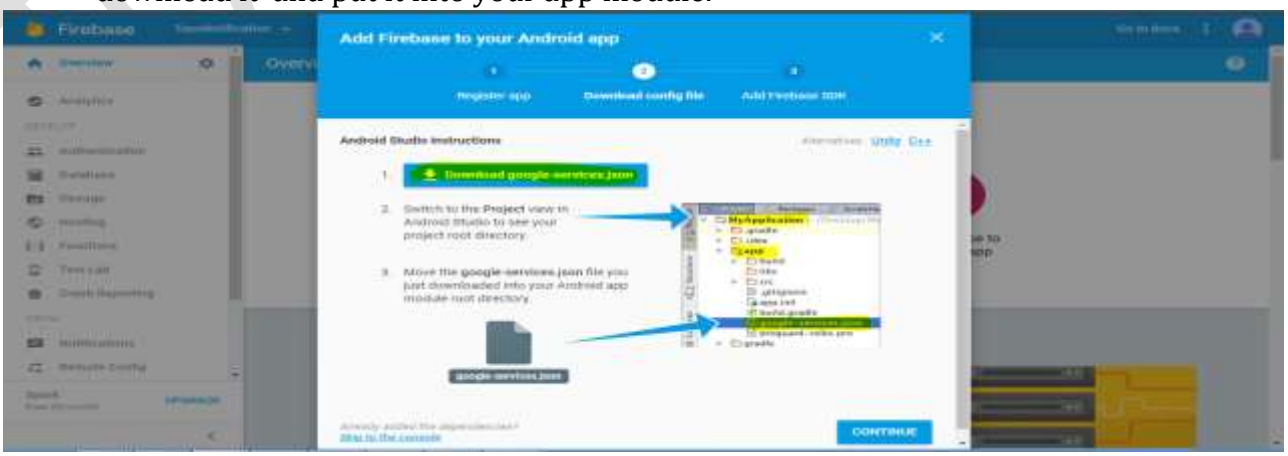
4. now you can see your dashboard of your project
now click on "Add Firebase to your Android App"



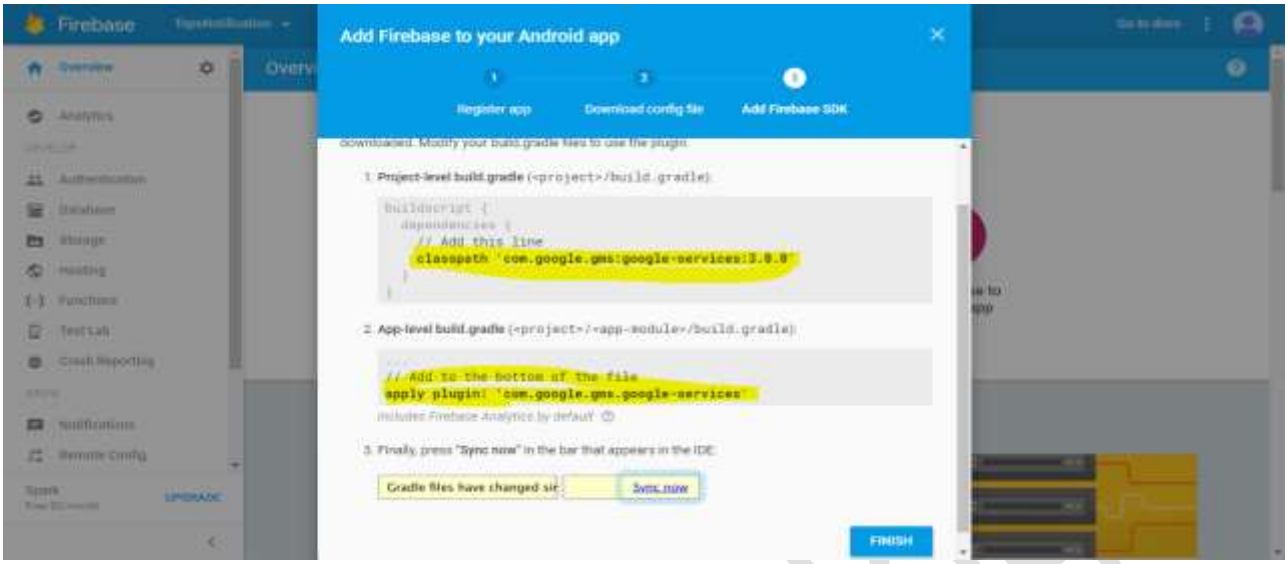
5. now give the package of the application give short name give SHA1 key



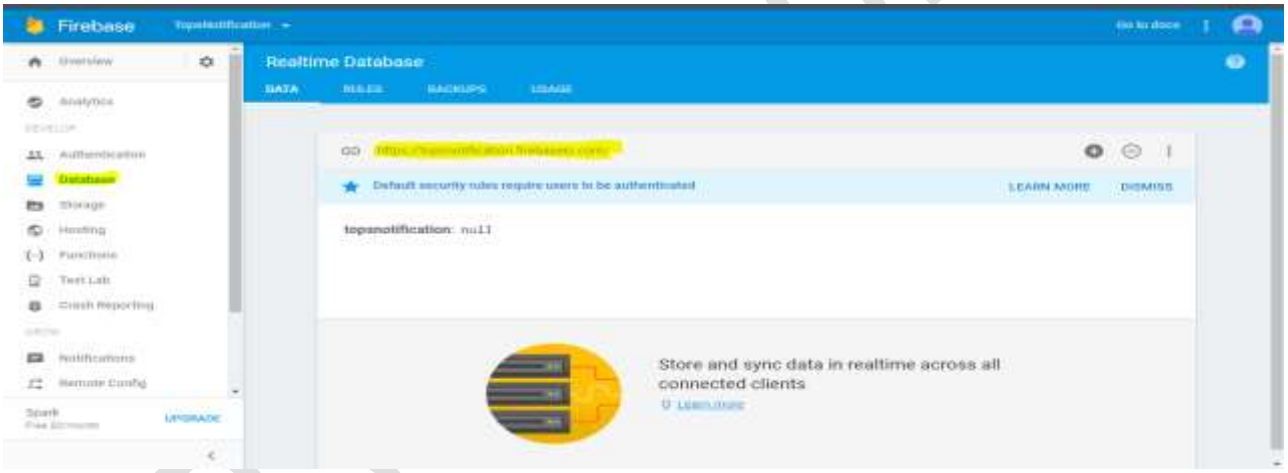
6. You will find the google-service.json file download it and put it into your app module.



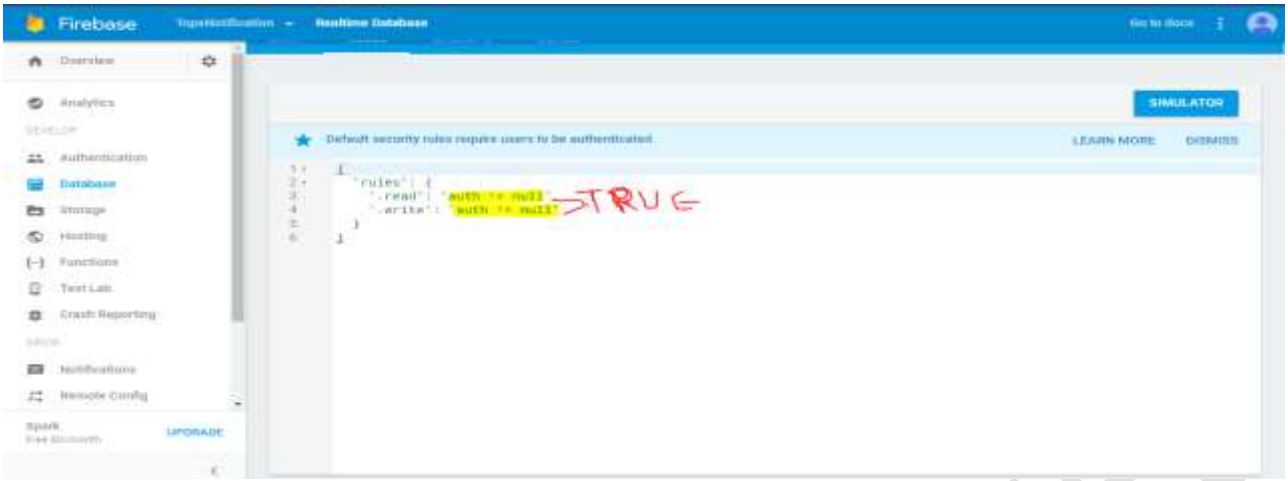
7. add dependencies and plugins to your application "build.gradle" file



8. now click on "database" tab from your dashboard
You will found a link of the firebase database we will use it in the php and android



9. now read and write permission to your firebase project for that click on "database" tab
now click on "rules"
now you will found ".read" and ".write"
change the permission of both to "true" and remove "auth != null"



10. add permission to your manifest files

11. Add this things into you app module gradle

Demo :

Create Constant.java File

```
public class Constants {
    //Firebase app url
    public static final String FIREBASE_APP = "https://fir-notification-
7fee8.firebaseio.com/";
    //Constant to store shared preferences
    public static final String SHARED_PREF = "mynotificationapp";
    //To store boolean in shared preferences for if the device is registered to
not
    public static final String REGISTERED = "registered";
    //To store the firebase id in shared preferences
    public static final String UNIQUE_ID = "uniqueid";
    //register.php address in your server
    public static final String REGISTER_URL =
"https://demoweapp.000webhostapp.com/firebasedata/register.php";

}
```

In activity_main.xml :

```
  <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:ems="10"
        android:hint="enter email"
        android:id="@+id/editTextEmail"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAllCaps="false"
        android:text="Register"
        android:id="@+id/buttonRegister"
        android:layout_below="@+id/editTextEmail"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAllCaps="false"
        android:text="Deregisters"
        android:id="@+id/btnderegister"
        android:layout_below="@+id/editTextEmail"
        android:layout_centerHorizontal="true" />

</LinearLayout>
```

In MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    //Creating Views
    private Button button,btndelete;
```

```
    private EditText editTextEmail;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //if the device is registered
        if(isRegistered()){
            startService(new Intent(this, NotificationListener.class));
        }
        //Initializing views
        button = (Button) findViewById(R.id.buttonRegister);
        btndelete = (Button) findViewById(R.id.btnderegister);
        editTextEmail = (EditText) findViewById(R.id.editTextEmail);

        //Attaching an onclicklistener
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                registerDevice();
                //if the device is not already registered
                if (!isRegistered()) {
                    //registering the device
                    registerDevice();
                } else {
                    //if the device is already registered
                    //displaying a toast
                    Toast.makeText(MainActivity.this, "Already registered...",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        btndelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            }
        });
    }

    private boolean isRegistered() {
        //Getting shared preferences
        SharedPreferences sharedPreferences =
getSharedPreferences(Constants.SHARED_PREF, MODE_PRIVATE);
        //Getting the value from shared preferences
        //The second parameter is the default value
        //if there is no value in sharedprference then it will return false
        //that means the device is not registered
        return sharedPreferences.getBoolean(Constants.REGISTERED, false);
    }

    private void registerDevice() {

        Firebase.setAndroidContext(this);
        //Creating a firebase object
        Firebase firebase = new Firebase(Constants.FIREBASE_APP);

        //Pushing a new element to firebase it will automatically create a unique
id
        Firebase newFirebase = firebase.push();
        //Creating a map to store name value pair
        Map<String, String> val = new HashMap<>();
        //pushing msg = none in the map
        val.put("msg", "none");
        //saving the map to firebase
        newFirebase.setValue(val);
        //Getting the unique id generated at firebase
        String uniqueId = newFirebase.getKey();
        //Finally we need to implement a method to store this unique id to our
```

```
server
        sendIdToServer(uniqueId);
    }

    private void sendIdToServer(final String uniqueId) {
        //Creating a progress dialog to show while it is storing the data on
server
        final ProgressDialog progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Registering device...");
        progressDialog.show();

        //getting the email entered
        final String email = editTextEmail.getText().toString().trim();

        //Creating a string request

        StringRequest req = new StringRequest(Request.Method.POST,
Constants.REGISTER_URL,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        //dismissing the progress dialog
                        progressDialog.dismiss();

                        //if the server returned the string success
                        if (response.trim().equalsIgnoreCase("success")) {
                            //Displaying a success toast
                            Toast.makeText(MainActivity.this, "Registered
successfully", Toast.LENGTH_SHORT).show();

                            //Opening shared preference
                            SharedPreferences sharedPreferences =
getSharedPreferences(Constants.SHARED_PREF, MODE_PRIVATE);

                            //Opening the shared preferences editor to save values
                            SharedPreferences.Editor editor =
sharedPreferences.edit();

                            //Storing the unique id
                            editor.putString(Constants.UNIQUE_ID, uniqueId);

                            //Saving the boolean as true i.e. the device is
registered
                            editor.putBoolean(Constants.REGISTERED, true);

                            //Applying the changes on sharedpreferences
                            editor.apply();

                            //Starting our listener service once the device is
registered
                            startService(new Intent(getBaseContext(),
NotificationListener.class));
                        } else {
                            Toast.makeText(MainActivity.this, "Choose a different
email", Toast.LENGTH_SHORT).show();
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                    }
                }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                //adding parameters to post request as we need to send firebase id
and email
                params.put("firebaseid", uniqueId);
```

```
                params.put("email", email);
                return params;
            }
        };
        //Adding the request to the queue
        RequestQueue requestQueue = Volley.newRequestQueue(this);
        requestQueue.add(req);
    }
}
```

Create MyApplication.java

```
public class MyApplication extends Application{

    @Override
    public void onCreate() {
        super.onCreate();
        //Initializing firebase
        Firebase.setAndroidContext(getApplicationContext());
    }
}
```

Create NotificationListener .java:

```
public class NotificationListener extends Service {


    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    //When the service is started
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        //Opening sharedpreferences
        SharedPreferences sharedPreferences =
getSharedPreferences(Constants.SHARED_PREF, MODE_PRIVATE);

        //Getting the firebase id from sharedpreferences
        String id = sharedPreferences.getString(Constants.UNIQUE_ID, null);

        //Creating a firebase object
        Firebase firebase = new Firebase(Constants.FIREBASE_APP + id);

        //Adding a valueevent listener to firebase
        //this will help us to  track the value changes on firebase
        firebase.addValueEventListener(new ValueEventListener() {

            //This method is called whenever we change the value in firebase
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                //Getting the value from firebase
                //We stored none as a initial value
                String msg = snapshot.child("msg").getValue().toString();

                //So if the value is none we will not create any notification
                if (msg.equals("none"))
                    return;

                showNotification(msg);
            }

            @Override
            public void onCancelled(FirebaseError firebaseError) {
                Log.e("The read failed: ", firebaseError.getMessage());
            }
        });

        return START_STICKY;
```

```
    }


    private void showNotification(String msg){
        //Creating a notification
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
        builder.setSmallIcon(R.mipmap.ic_launcher);
        Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.simplifiedcoding.net"));
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
0);
        builder.setContentIntent(pendingIntent);

        builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.mipmap.ic_launcher));
        builder.setContentTitle("Firebase Push Notification");
        builder.setContentText(msg);
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify(1, builder.build());
    }
}
```

Out put:

install apk

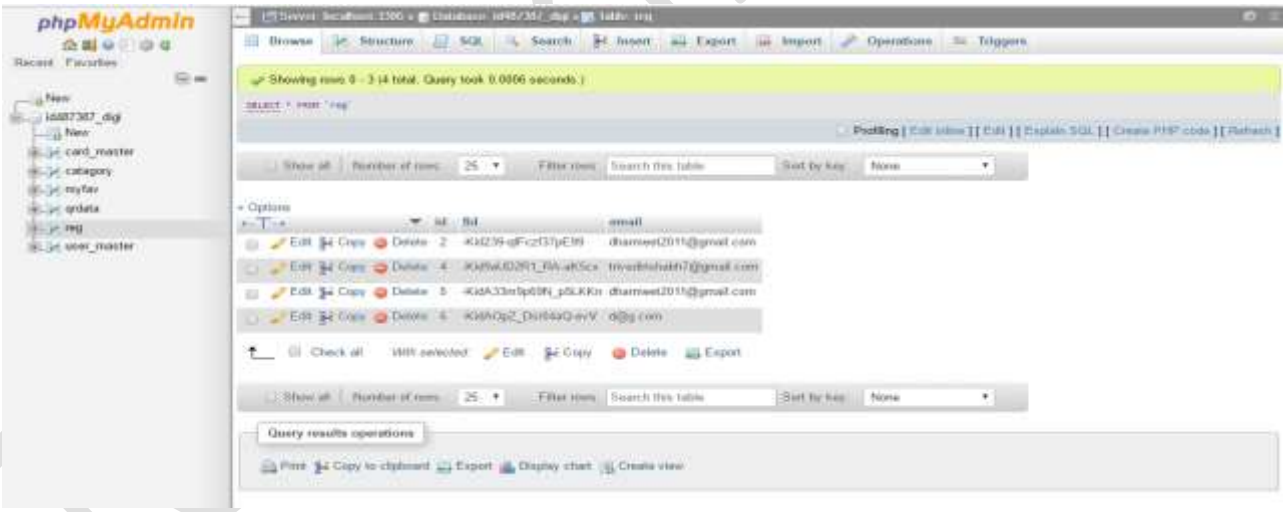
Register with any email id

open link below in browser and Send notification

https://demoweapp.000webhostapp.com/firebasedata/sendPushNotification.php

send Notification.

This is database which used



This is firebase database where our device is registered.

Social Media Integration :

- Android Login With Facebook Account

Create New Android Project :

- Open Android Studio and Click on File-> New ->New Project
- Give your application a name and a package.

Configure Facebook SDK to android studio

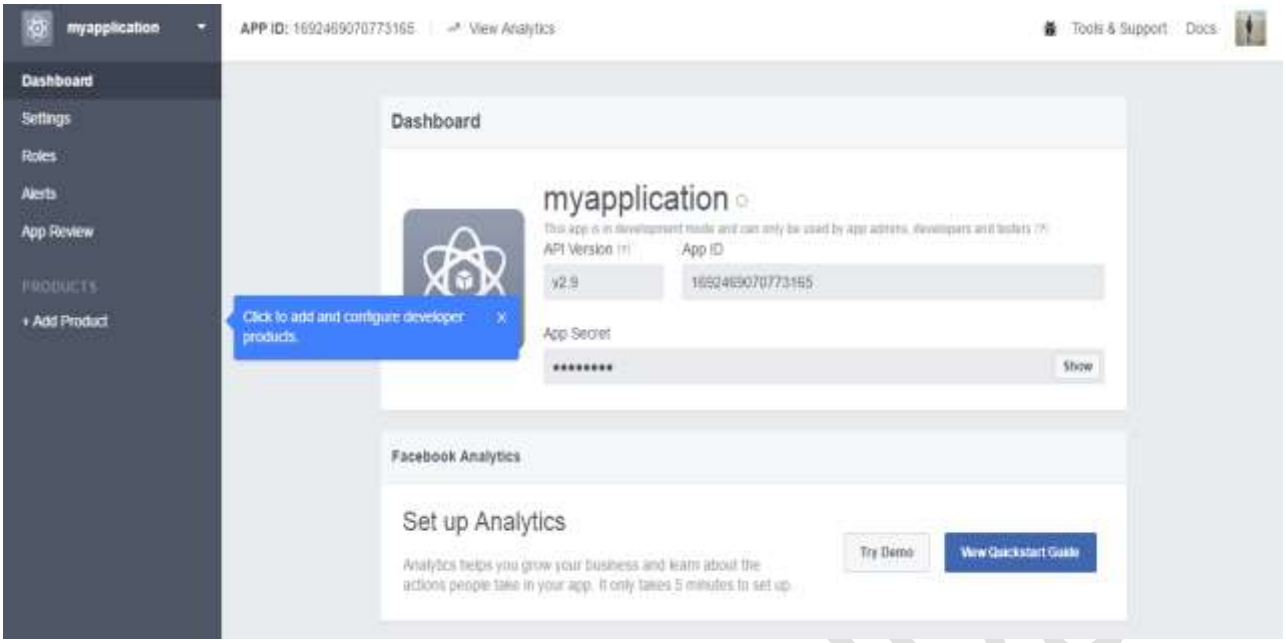
- Go to your gradle scripts -> build.gradle(Module:app)
- add a new dependency for Facebook SDK
  compile 'com.facebook.android:facebook-android-sdk:4.0.0'

Creating Facebook App

Go to https://developers.facebook.com/. If you have not registered yourself as a developer yet then facebook will ask you to register as a developer.

Create new App Id.

- You will be redirected to your apps dashboard.



- Here you can get your app id. Copy the app id it will be used further.
- Now from the left click on settings.
- Click on add a platform and select android.
- Enter your package name and class name of your main activity and click on save changes
- The last thing you need is your App Key Hashes.

Generating Key Hashes for your Login With Facebook Android App

- Go to android studio.
- On your project open your strings.xml file (res->values->strings.xml)
- Add the below code

```
<string name="app_id">YOUR APP ID</string>
```

- Replace your actual app id with your app id.
- Now go to your AndroidManifest.xml
- First add internet permission using the following code

```
<uses-permission android:name="android.permission.INTERNET"/>
<meta-data android:name="com.facebook.sdk.ApplicationId"
android:value="@string/app_id"/>
```

Generate HashKey using this code :

```
public class MyApplication extends Application {
  public void onCreate() {
    super.onCreate();
    printHashKey();
  }
  public void printHashKey()
        {
         // Add code to print out the key hash
    try {
                PackageInfo info = getPackageManager().getPackageInfo(
        "com.admin.example.facebookdemo",PackageManager.GET_SIGNATURES);
        for (Signature signature : info.signatures) {
        MessageDigest md = MessageDigest.getInstance("SHA");
```

```
            md.update(signature.toByteArray());
            Log.d("KeyHash:", Base64.encodeToString(md.digest(), Base64.DEFAULT));
             }
                    } catch (PackageManager.NameNotFoundException e)
            } catch (NoSuchAlgorithmException e) {
              }
            }
}
```

NOW ADD THIS TO ACTIVITY_MAIN.XML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <com.facebook.login.widget.LoginButton
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

Now Add this code to your MainActivity.java :

```
public class MainActivity extends ActionBarActivity {

    private CallbackManager callbackManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        FacebookSdk.sdkInitialize(getApplicationContext());
        setContentView(R.layout.activity_main);

        final LoginButton loginButton = (LoginButton) findViewById(R.id.login_button);

        callbackManager = CallbackManager.Factory.create();

        loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                Toast.makeText(MainActivity.this, "Login Success", Toast.LENGTH_LONG).show();

            }

            @Override
            public void onCancel() {
                Toast.makeText(MainActivity.this, "Login Canceled", Toast.LENGTH_LONG).show();
            }

            @Override
            public void onError(FacebookException e) {
                Toast.makeText(MainActivity.this, e.getLocalizedMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        callbackManager.onActivityResult(requestCode, resultCode, data);
    }
}
```

- Android Login With Google Plus Account :

Get a configuration file

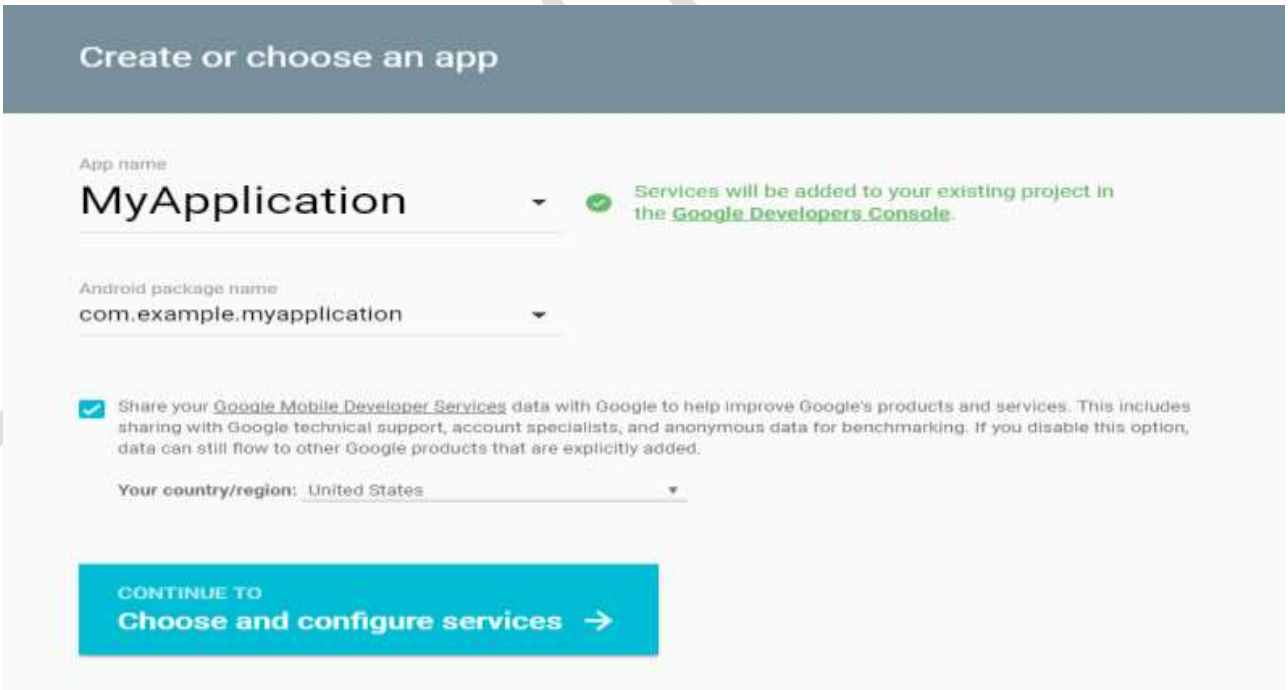
- Google plus sign-in lets users sign in to your Android app with their existing Google account and get their profile information like name, email, profile pic and other details.
- Go to android quick start guide and click on Get A Configuration File button. This will redirect you to a page where you can choose the project and package name.
- Now all the android projects which uses google apis, requires **google-services.json** file to be placed in project's app folder. Follow the below steps to get your google-services.json file.
  Java **keytool** can be used to generate SHA-1 fingerprint. Open your terminal and execute the following command to generate SHA-1 fingerprint. If it ask for password, type **android** and press enter.
- **On windows**
  keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
- **On Linux or Mac OS**
  keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

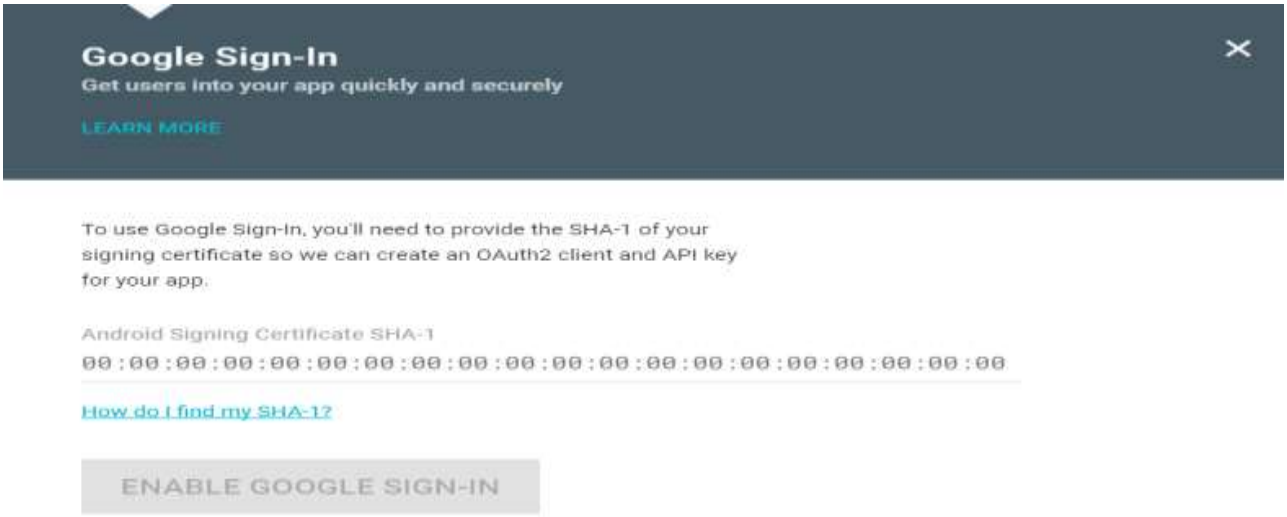
```
C:\Users\Abraham>keytool -exportcert -alias androiddebugkey -keystore "c:\Users\Abraham\.android\deb
ug.keystore" -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: May 13, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 6f3eb719
Valid from: Mon May 13 02:13:08 IST 2013 until: Wed May 06 02:13:08 IST 2043
Certificate fingerprints:
         MD5:  7F:21:AF:F4:0B:67:4F:88:12:90:54:69:5E:6D:BE:DE
         SHA1: E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2
         SHA256: 3E:23:20:84:9A:77:66:52:1E:1B:E2:D5:E8:13:BE:35:FA:A8:9F:B3:86:97:F6:15:EF:15:11:62
:13:EE:AD:F9
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:
```

Goto android quick start guide and click on Get A Configuration File button. This will redirect you to a page where you can choose the project and package name.

Create / choose an app and give your current app **package** name. I gave my package name as **com.example.myapplication**.

Paste the **SHA-1** fingerprint and click on **Enable Google Sign-In**. Finally click on **Generate Configuration File** to download your google-services.json





2. Create New Project:

Add Dependency to App Level Build.gradle and apply Plugin at the bottom of App level Build.Gradle ,

```
dependencies {
        compile 'com.google.android.gms:play-services-auth:9.2.1'
        compile 'com.github.bumptech.glide:glide:3.7.0'
}
apply plugin: 'com.google.gms.google-services'
```

Put Below code in your MainActivity.xml file ;

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical"
    android:paddingTop="16dp"
    tools:context="com.example.admin.googlepluslogin.MainActivity">

    <LinearLayout
        android:id="@+id/llProfile"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:orientation="horizontal"
        android:weightSum="3"
        android:visibility="gone">

        <ImageView
            android:id="@+id/imgProfilePic"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"/>
```

```
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:orientation="vertical"
        android:layout_weight="2" >

        <TextView
            android:id="@+id/txtName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:textSize="20dp" />

        <TextView
            android:id="@+id/txtEmail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:textSize="18dp" />
    </LinearLayout>
</LinearLayout>

<com.google.android.gms.common.SignInButton
    android:id="@+id/btn_sign_in"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/btn_sign_out"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Logout from Google"
        android:visibility="gone"
        android:layout_marginBottom="10dp"/>

    <Button
        android:id="@+id/btn_revoke_access"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Revoke Access"
        android:visibility="gone" />
</LinearLayout>
```

Now open **MainActivity.java** and do the below modifications. The code is self explanatory and very easy understand.

- implement the activity from **GoogleApiClient.OnConnectionFailedListener**
- Create the **GoogleApiClient** instance in **onCrate()** method.
- **signIn()** performs google plus sign in, **signOut()** logs out user from google account and **revokeAccess()** completely revokes the access from google plus.
- **onActivityResult()** is called whenever user returns from Google Login UI.
- In **onStart()** method, checked for cached google sign in session and appropriate UI is displayed.
- **handleSignInResult()** handles the google plus profile information upon successful login.
- **updateUI()** toggles the UI by showing / hiding the appropriate buttons and text views.

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener,GoogleApiClient.OnConnectionFailedListener  {
    private static final String TAG = MainActivity.class.getSimpleName();
    private static final int RC_SIGN_IN = 007;

    private GoogleApiClient mGoogleApiClient;
    private ProgressDialog mProgressDialog;

    private SignInButton btnSignIn;
    private Button btnSignOut, btnRevokeAccess;
    private LinearLayout llProfileLayout;
    private ImageView imgProfilePic;
    private TextView txtName, txtEmail;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnSignIn = (SignInButton) findViewById(R.id.btn_sign_in);
        btnSignOut = (Button) findViewById(R.id.btn_sign_out);
        btnRevokeAccess = (Button) findViewById(R.id.btn_revoke_access);
        llProfileLayout = (LinearLayout) findViewById(R.id.llProfile);
        imgProfilePic = (ImageView) findViewById(R.id.imgProfilePic);
        txtName = (TextView) findViewById(R.id.txtName);
        txtEmail = (TextView) findViewById(R.id.txtEmail);

        btnSignIn.setOnClickListener(this);
        btnSignOut.setOnClickListener(this);
        btnRevokeAccess.setOnClickListener(this);


    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestEmail()
            .build();

    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this, this)
            .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
            .build();

    // Customizing G+ button
    btnSignIn.setSize(SignInButton.SIZE_STANDARD);
    btnSignIn.setScopes(gso.getScopeArray());
}


private void signIn() {
    Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
    startActivityForResult(signInIntent, RC_SIGN_IN);
}


private void signOut() {
    Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
            new ResultCallback<Status>() {
                @Override
                public void onResult(Status status) {
                    updateUI(false);
                }
            });
}
```

```
private void revokeAccess() {
    Auth.GoogleSignInApi.revokeAccess(mGoogleApiClient).setResultCallback(
            (ResultCallback) (status) -> { updateUI(false); });
}

private void handleSignInResult(GoogleSignInResult result) {
    Log.d(TAG, "handleSignInResult:" + result.isSuccess());
    if (result.isSuccess()) {
        // Signed in successfully, show authenticated UI.
        GoogleSignInAccount acct = result.getSignInAccount();

        Log.e(TAG, "display name: " + acct.getDisplayName());

        String personName = acct.getDisplayName();
        String personPhotoUrl = acct.getPhotoUrl().toString();
        String email = acct.getEmail();

        Log.e(TAG, "Name: " + personName + ", email: " + email
                /* + ", Image: " + personPhotoUrl*/  );

        txtName.setText(personName);
        txtEmail.setText(email);
        Glide.with(getApplicationContext()).load(personPhotoUrl)
                .thumbnail(0.5f)
                .crossFade()
                .diskCacheStrategy(DiskCacheStrategy.ALL)
                .into(imgProfilePic);
        updateUI(true);
    } else {
        // Signed out, show unauthenticated UI.
        updateUI(false);
    }}
```

```
@Override
public void onClick(View v) {
    int id = v.getId();

    switch (id) {
        case R.id.btn_sign_in:
            signIn();
            break;

        case R.id.btn_sign_out:
            signOut();
            break;

        case R.id.btn_revoke_access:
            revokeAccess();
            break;
    }
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

```
@Override
public void onClick(View v) {
    int id = v.getId();

    switch (id) {
        case R.id.btn_sign_in:
            signIn();
            break;

        case R.id.btn_sign_out:
            signOut();
            break;

        case R.id.btn_revoke_access:
            revokeAccess();
            break;
    }
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

```
private void showProgressDialog() {
    if (mProgressDialog == null) {
        mProgressDialog = new ProgressDialog(this);
        mProgressDialog.setMessage("Loading....");
        mProgressDialog.setIndeterminate(true);
    }

    mProgressDialog.show();
}

private void hideProgressDialog() {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.hide();
    }
}

private void updateUI(boolean isSignedIn) {
    if (isSignedIn) {
        btnSignIn.setVisibility(View.GONE);
        btnSignOut.setVisibility(View.VISIBLE);
        btnRevokeAccess.setVisibility(View.VISIBLE);
        llProfileLayout.setVisibility(View.VISIBLE);
    } else {
        btnSignIn.setVisibility(View.VISIBLE);
        btnSignOut.setVisibility(View.GONE);
        btnRevokeAccess.setVisibility(View.GONE);
        llProfileLayout.setVisibility(View.GONE);
    }

}
```

Work with Android System

Wake Locks

- **WakeLocks** are a Power Manager system Service feature, available to your applications to control the power state of the host device.
- Wake Locks can be used to keep the CPU running, prevent the screen from dimming, prevent the screen from turning off, and prevent the keyboard backlight from turning off.
- **Creating and holding Wake Locks can have a dramatic influence on the battery drain associated with your application. It's good practice to use Wake Locks only when strictly necessary, for as short a time as needed, and to release them as soon as possible.**
- Screen Wake Locks are typically used to prevent the screen from dimming during applications that are likely to involve little user interaction while users observe the screen
- To create a Wake Lock, call newWakeLock on the Power Manager, specifying one of the following Wake Lock types:
    - **FULL_WAKE_LOCK** Keeps the screen at full brightness, the keyboard backlight illuminated, and the CPU running.
    - **SCREEN_BRIGHT_WAKE_LOCK** Keeps the screen at full brightness, and the CPU running.
    - **SCREEN_DIM_WAKE_LOCK** Keeps the screen on (but lets it dim) and the CPU running.
    - **PARTIAL_WAKE_LOCK** Keeps the CPU running.
    - **PowerManager pm = (PowerManager)getSystemService(Context.POWER_SERVICE);**
    - **WakeLock wakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "MyWakeLock");**
- You can optionally specify a timeout to ensure the maximum duration the Wake Lock will be held for. When the action for which you're holding the Wake Lock completes, call release to let the system manage the power state.
- ***NOTE: it is not work on emulator so please try this example on your ANDROID phone***

WakeLock.java

```
public class WakeLockDemo extends Activity {
	private Button btn;
	private WakeLock lock;
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_wake_lock_demo);
//Set power manager
PowerManager power = (PowerManager)getSystemService(Context.POWER_SERVICE);
// Initialize WakeLock
	lock = power.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, "MyLock");
		btn = (Button) findViewById(R.id.button1);
		btn.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				// Relese wake lock
				lock.release();
			}
		});
	}
}
```

Text to Speech

- Android 1.6 (SDK API level 4) introduced the text to speech (TTS) engine. You can use this API to produce speech synthesis from within your applications, allowing them to ‘‘talk’’ to your users.
- *Due to storage space constraints on some Android devices, the language packs are not always preinstalled on each device. Before using the TTS engine, it’s good practice to confirm the language packs are installed.*
- Start a new Activity for a result using the ACTION_CHECK_TTS_DATA action from the TextToSpeech. Engine class to check for the TTS libraries.
- The onActivityResult handler will receive CHECK_VOICE_DATA_PASS if the voice data has been installed successfully.
- If the voice data is not currently available, start a new Activity using the ACTION_INSTALL_TTS_DATA action from the TTS Engine class to initiate its installation.
- When Text To Speech has been initialized you can use the speak method to synthesize voice using the default device audio output.

```
tts.speak("Hello, Android", TextToSpeech.QUEUE_ADD, null);
```

- The speak method lets you specify a parameter to either add the new voice output to the existing queue, or flush the queue and start speaking straight away.

XML File

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:background="@null"
        android:src="@drawable/ic_launcher" />
<EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:inputType="textNoSuggestions" >
<requestFocus />
</EditText>
</RelativeLayout>
```

Java File

```
publicclass TextToSpeechDemo extends Activity {
      private EditText txtSpeak;
      private ImageButton imgSpeak;
      private TextToSpeech tts = null;
      privateboolean isInitTTS;
      privatestaticintTTS_DATA_CHECK = 1;
      @Override
      publicvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_text_to_speech_demo);
```

```
            txtSpeak = (EditText) findViewById(R.id.editText1);
            imgSpeak = (ImageButton) findViewById(R.id.imageButton1);
            imgSpeak.setOnClickListener(new OnClickListener() {
            @Override
            publicvoid onClick(View arg0) {
                  // Intent to start TTS Engine
                  Intent intent = new Intent(Engine.ACTION_CHECK_TTS_DATA);
                  startActivityForResult(intent, TTS_DATA_CHECK);
                  }
            });
      }
protectedvoid onActivityResult(int RequestCode, int ResultCode, Intent Data) {
if (RequestCode == TTS_DATA_CHECK) {
      //Check for Success on activity result
      if (ResultCode == Engine.CHECK_VOICE_DATA_PASS) {
      tts = new TextToSpeech(this, new OnInitListener() {
            @Override
            publicvoid onInit(int status) {
            // TODO Auto-generated method stub
            if (status == TextToSpeech.SUCCESS) {
                  isInitTTS = true;
                        if (tts.isLanguageAvailable(Locale.US) >= 0) {
                              tts.setLanguage(Locale.US); // Set Language
                              tts.setPitch(0.8f); // Set Pitch
                              tts.setSpeechRate(1.1f); // Set Speech Rate
                              if (tts != null&& isInitTTS) {
                                    tts.speak(txtSpeak.getText().toString(),
            TextToSpeech.QUEUE_ADD, null); // Speak and add in queue
            Toast.makeText(TextToSpeechDemo.this,
                              “Speaking", Toast.LENGTH_LONG).show();
                                          }
                                    } else {
Intent intn = new Intent(Engine.ACTION_INSTALL_TTS_DATA);
                        startActivity(intn);
                                          }
                                    }
                              }
                        });
                  }
            }
      }
```

Using Camera

Most Android devices have at least one camera. Some devices have a front and a back facing camera. Using the camera on the Android device can be done via the integration of existing camera application. In this case you would start the existing Camera application via an *intent* and use the return data of the application to access the result.

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
        android:id="@+id/btn_tc_pic"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="Take_Photo" />
<ImageView
        android:id="@+id/image_v_cam"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn_tc_pic"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true" />
</RelativeLayout>
```

Activity Class

```
public class MainActivity extends Activity {
      private Button btn_tcP;
      private ImageView show_p;
      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            btn_tcP = (Button) findViewById(R.id.btn_tc_pic);
            show_p = (ImageView) findViewById(R.id.image_v_cam);
            btn_tcP.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {

      // Create intent to open camera activity
            Intent i = new Intent(
      android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
      startActivityForResult(i, 1);
                  }
            });
      }
      @Override
      protected void onActivityResult(int requestCode, int resultCode, Intent
data)       {
            // TODO Auto-generated method stub
            super.onActivityResult(requestCode, resultCode, data);
      // Get and Set Image from Activity Result
            Bitmap imageurl = (Bitmap) data.getExtras().get("data");
            show_p.setImageBitmap(imageurl);
      }
}
```

Android Bluetooth

Bluetooth is a way to exchange data with other devices wirelessly. Android provides Bluetooth API to perform several tasks such as:

BluetoothAdapter class

By the help of BluetoothAdapter class, we can perform fundamental tasks such as initiate device discovery, query a list of paired (bonded) devices, create a BluetoothServerSocket instance to listen for connection requests etc.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<TextView
        android:id="@+id/out"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" ></TextView>
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="49dp"
        android:text="TURN_ON" />
<Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/button1"
        android:layout_marginTop="27dp"
        android:text="DISCOVERABLE" />
<Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button2"
        android:layout_below="@+id/button2"
        android:layout_marginTop="28dp"
        android:text="TURN_OFF" />
</RelativeLayout>
```

```
publicclass MainActivity extends Activity {
      privatestaticfinalint REQUEST_ENABLE_BT = 0;
      privatestaticfinalint REQUEST_DISCOVERABLE_BT = 0;
      @Override
      protectedvoid onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            final TextView out = (TextView) findViewById(R.id.out);
            final Button button1 = (Button) findViewById(R.id.button1);
            final Button button2 = (Button) findViewById(R.id.button2);
            final Button button3 = (Button) findViewById(R.id.button3);
            final BluetoothAdapter mBluetoothAdapter = BluetoothAdapter
                        .getDefaultAdapter();
            if (mBluetoothAdapter == null) { //Check Bluetooth not available
                  out.append("device not supported");
            }
            button1.setOnClickListener(new View.OnClickListener() {
                  publicvoid onClick(View v) {
      // Check if Bluetooth alredy on
```

```
                        if (!mBluetoothAdapter.isEnabled()) {
                              Intent enableBtIntent = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
                    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
                              }
                        }
              });
              button2.setOnClickListener(new View.OnClickListener() {
                    @Override
                    publicvoid onClick(View arg0) {
                        if (!mBluetoothAdapter.isDiscovering()) {
              // Make Device Discoverable
        Toast.makeText(getApplicationContext(),
                                          "MAKING YOUR DEVICE DISCOVERABLE",
                                          Toast.LENGTH_LONG);
        Intent enableBtIntent = new Intent(
        BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
                                  startActivityForResult(enableBtIntent,
                                          REQUEST_DISCOVERABLE_BT);
                              }
                        }
              });
              button3.setOnClickListener(new View.OnClickListener() {
                    @Override
                    publicvoid onClick(View arg0) {
                        mBluetoothAdapter.disable();
              Toast.makeText(getApplicationContext(),
        "TURNING_OFF BLUETOOTH", Toast.LENGTH_LONG);
                    }
              });
        }
}
```

You need to provide following permissions in AndroidManifest.xml file.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Android WiFi

The **android.net.wifi.WifiManager** class can be used to manage the wifi connectivity. It can be used to add network, disable network, scan for access points, disconnect etc.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="76dp"
        android:layout_marginTop="67dp"
        android:text="Enable Wifi" />
<Button
        android:id="@+id/button2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/button1"
        android:layout_marginTop="44dp"
        android:text="Disable Wifi" />
</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends Activity {
    Button enableButton,disableButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        enableButton=(Button)findViewById(R.id.button1);
        disableButton=(Button)findViewById(R.id.button2);
        enableButton.setOnClickListener(new OnClickListener(){
            public void onClick(View v){
      // Initialize wifi manager
                WifiManager wifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(true);
            }
        });
        disableButton.setOnClickListener(new OnClickListener(){
            public void onClick(View v){
                WifiManager wifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(false);
            }
        });
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

Android Sensor

Sensors can be used to monitor the three-dimensional device movement or change in the environment of the device.

Android provides sensor API to work with different types of sensors.

Android simple sensor app example

Let's see the two sensor examples.

1. A sensor example that prints x, y and z axis values. Here, we are going to see that.
2. A sensor example that changes the background color when device is shuffled. Click for **changing background color of activity sensor example**

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="92dp"
        android:layout_marginTop="114dp"
        android:text="TextView" />
</RelativeLayout>
```

Java File

```
public class MainActivity extends Activity {
    SensorManager sm = null;
    TextView textView1 = null;
    List list;
    SensorEventListener sel = new SensorEventListener(){
        public void onAccuracyChanged(Sensor sensor, int accuracy) {}
        public void onSensorChanged(SensorEvent event) {
            float[] values = event.values;
            textView1.setText("x: "+values[0]+"\ny: "+values[1]+"\nz:
"+values[2]);
        }
    };
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
// Initialize Sensor Manager
      sm = (SensorManager)getSystemService(SENSOR_SERVICE);
        textView1 = (TextView)findViewById(R.id.textView1);
// Get list of sensors
        list = sm.getSensorList(Sensor.TYPE_ACCELEROMETER);
        if(list.size()>0){
sm.registerListener(sel, (Sensor) list.get(0),
SensorManager.SENSOR_DELAY_NORMAL);
        }else{
     Toast.makeText(getBaseContext(), "Error: No Accelerometer.",
Toast.LENGTH_LONG).show();
        }
    }
    protected void onStop() {
        if(list.size()>0){
      // Unregister sensor
          sm.unregisterListener(sel);
        }
        super.onStop();
    }  }
```

Making Phone Call

Make a phone call in android

We are able to make a phone call in android via intent. You need to write only three lines of code to make a phone call.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="118dp"
        android:text="Call" />
<EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="25dp"
        android:ems="10" />
</RelativeLayout>
```

Permission in Manifist

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

Java File

```
public class MainActivity extends Activity {
    EditText edittext1;
    Button button1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Getting the edittext and button instance
        edittext1=(EditText)findViewById(R.id.editText1);
        button1=(Button)findViewById(R.id.button1);

        //Performing action on button click
        button1.setOnClickListener(new OnClickListener(){

            @Override
            public void onClick(View arg0) {
                String number=edittext1.getText().toString();
                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:"+number));
                startActivity(callIntent);
            }
        });
    }
}
```

Send SMS

Send sms in android

We can send sms in android via intent. You need to write only 4 lines of code the send sms in android.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"   >
<EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="20dp"
        android:ems="10" />

<EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText1"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="26dp"
        android:ems="10"
        android:inputType="textMultiLine" />

<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText1"
        android:layout_alignBottom="@+id/editText1"
        android:layout_toLeftOf="@+id/editText1"
        android:text="Mobile No:" />
<TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText2"
        android:layout_alignBottom="@+id/editText2"
        android:layout_alignLeft="@+id/textView1"
        android:text="Message:" />

<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText2"
        android:layout_below="@+id/editText2"
        android:layout_marginLeft="34dp"
        android:layout_marginTop="48dp"
        android:text="Send SMS" />
</RelativeLayout>
public class MainActivity extends Activity {
    EditText mobileno,message;
    Button sendsms;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mobileno=(EditText)findViewById(R.id.editText1);
        message=(EditText)findViewById(R.id.editText2);
```

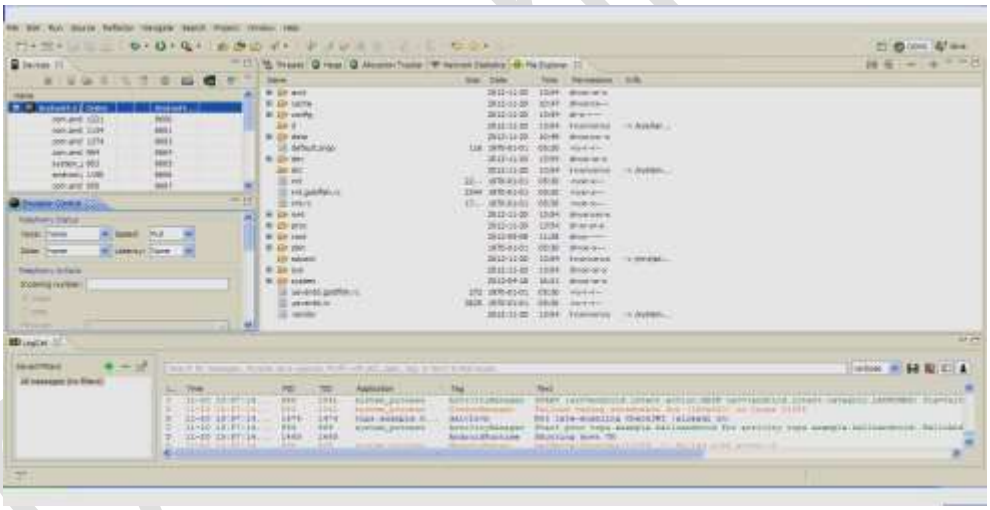
```
        sendsms=(Button)findViewById(R.id.button1);
sendsms.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                String no=mobileno.getText().toString();
                String msg=message.getText().toString();
                Intent intent=new
Intent(getApplicationContext(),MainActivity.class);
                PendingIntent
pi=PendingIntent.getActivity(getApplicationContext(), 0, intent,0);
                SmsManager sms=SmsManager.getDefault();
                sms.sendTextMessage(no, null, msg, pi,null);
                Toast.makeText(getApplicationContext(), "Message Sent
successfully!",
                    Toast.LENGTH_LONG).show();
            }
        });      }  }
```

SMS Send Permission

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Dalvik Debug Monitoring Service (DDMS)

- Use the DDMS perspective to monitor and control the Dalvik virtual machines on which you're debugging your applications.
- Android ships with a debugging tool called the Dalvik Debug Monitor Server (DDMS), which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location
- data spoofing, and mo



Android Asset Packaging Tool (AAPT)

Constructs the distributable Android package files (.apk).

The following diagram depicts the components involved in building and running an application:

Android Debug Bridge (ADB)

The *ADB* is a client-server application that provides a link to a running emulator. It lets you copy files, install compiled application packages (.apk), and run shell commands.

Test Application on Real Device

- Plug in your device to your development machine with a USB cable. If you're developing on Windows, you might need to install the appropriate USB driver for your device. For help installing drivers, see the OEM USB Drivers document.
- Enable **USB debugging** on your device.
    - On most devices running Android 3.2 or older, you can find the option under **Settings > Applications > Development**.
    - On Android 4.0 and newer, it's in **Settings > Developer options**.

Note: On Android 4.2 and newer, Developer options is hidden by default. To make it available, go toSettings > About phone and tap Build number seven times. Return to the previous screen to findDeveloper options.

To run the app from Eclipse:

- Open one of your project's files and click **Run** from the toolbar.
- In the **Run as** window that appears, select **Android Application** and click **OK**.

Eclipse installs the app on your connected device and starts it.

Or to run your app from a command line:

Publish Application

- If you're creating an application that you want to distribute to the public, the best way to do so is to upload your application to Google Play.

Register for Google Play

- The first step in publishing is to register for Google Play at the Google Play publisher site. The process is relatively straightforward and should only take you a few minutes.

Note: There is a $25 USD charge to register.

Prepare to Publish

The publishing process in Google Play can be a lot quicker if you have all of the store listing content complete and available before you start. So when you're ready to publish your app, here are the items you should have available:

- Your final Android application file (APK) must be under 50MB in size
- The title for your app (max 30 characters).
- A description of your app (max 4000 characters).
- High-res icon, 512 x 512 32-bit PNG (with alpha).
- At least 2 screenshots are required overall (Max 8 screenshots per type), JPEG or 24-bit PNG (no alpha). Min length for any side: 320px. Max length for any side: 3840px. (You can upload a maximum of 8 screenshots per type. Types include "Phone", "7-inch tablet" and "10-inch tablet".)

Optional items that can be updated later include:

- Optional promo text (80 characters) and promo graphic. Promo graphic needs to be 180w x 120 h JPG or 24-bit PNG (no alpha).
- Screenshots: Only 2 are required, but you can upload a maximum of 8 screenshots per type. Types include "Phone", "7-inch tablet" and "10-inch tablet".
- Feature graphic: 1024w x 500h JPG or 24-bit PNG (no alpha).

Add New Application

- Once you've registered, you can log in to your Google Play developer account and begin publishing your application via the "+ Add new application" button. The first thing you'll see is a dialog asking for the default language and the title of your app. You're then presented two options "Upload APK"

and "Prepare Store Listing". The one you start with is up to you, but for this article, we'll assume you'll choose the upload first.

Upload APK

- This step is the process of uploading your application's Android Package File (APK). A link to this file is emailed to you whenever you successfully build your application, and can be downloaded at any time from your projects section on Andromo. Click the "Upload APK" button, and browse for your application file to upload. You'll see a "Save Draft" button on the dialog box after you upload the APK. Once you complete this step, you'll see a checkmark listed beside "APK" along the left hand side of the screen. The next step is to prepare your store listing.

Pricing & Distribution

- In this section, you define the pricing details for your app (free or paid), where you want it available, and consent options. When you're finished, save your changes and if you've entered all of the required information, it will show a checkmark next to "Pricing & Distribution" along the left-hand side.

Publishing your App

- If you've submitted all of the required information, a "Publish this app" button will be enabled on the page. Your application should appear in the store usually after a few hours (but could take longer). If you've included AdMob in your app, don't forget to "link your app" which you can normally do about 24 hours after it's been published.

ANDROID INTERVIEW QUESTION

Core java

1.What is the difference between static and non static variables ?
2. What are pass by value and pass by reference?
3.What is synchronization
4.What are the differences between an abstract class and an interface?
5.What are different type of exceptions in Java?
6.What is an abstract class and Abstract Method?
7.What is an interface?
8. What is final ?
10. What is difference between Constuctor and Method ?
11.What is Collection API ?
12.Explain the user defined Exceptions?
13.Explain garbage collection?
14.What is OOPS?
15.Explain the Encapsulation principle.
16.Explain the Polymorphism principle.
17.Explain the Inheritance principle.
18.Explain the different forms of Polymorphism.
19.What are Access Specifiers available in Java?
20.Describe the wrapper classes in Java.
21.What is method overriding?
22.What is method overloading?
23.Difference between Exception and Error.
24.Use of Observer and Oberservable.
25.What is Constructor ?
26.Explain parameterised Constructor.
27.**Destructor** in java.

Android

1. What is an Intent?
2. What is a Sticky Intent?
3. What is a resource?
4. Describe Briefly the Android Application Architecture
5. Which are the formats in which we get the response through the web service.
6. Can an application be started on powerup?
7. Life Cycle of android activity ?
9. What is needed to make a multiple choice list with a custom view for each row?
10. What dialog boxes are supported in android?Android supports 4 dialog boxes:
11. How is nine-patch image different from a regular bitmap?
12. What's the difference between file, class and activity in android?
13. Explain about the exceptions of Android?
14. What is an adb ?
15. Describe the APK format.
16. Explain Service in android.
17. Give brief introduction about Broadcast Receiver.

18. What is android manifest file is for?
19. What is .dex extension
20. What is an Intent Receiver?
21. What is fragmentation in android?
22. Explain about the R.java file.
23. What is shared preference ?
24. What is SQLite?
25.What is Android Runtime?
26. How to connect SQLite database with android application
27.What is use of SQLiteHelper class?
29. What are the different ways of storing the data in the phone?
30. Steps to start use of google map api in android.
31. Different kinds of Intents.
32. How to ensure that the app design will be consistent across the different screen resolutions.
33. How to pass the data across the different activities?
34. Service Life Cycle
36. What is parsing?Types of Parsing?Types of parsing?
37. Why cannot you run standard Java bytecode on Android?
38. When does Android start and end an application process?
39. How does Android system track the applications?
40.Android application can be only programmed in java?
41. What is Toast in Android?
42. What is Notification Manager in Android?
45. Android Implicit Intent and late time binding?
46. How many ways to store data in Android?