# Android Development

Summer Training Report

Submitted in the partial fulfillment for the degree of

Bachelor of Technology

In

Information Technology & Engineering

by

**Palak Singhal – 02196303116**


Maharaja Surajmal Institute of Technology

(Affiliated to Guru Gobind Singh Indraprastha University)

Janakpuri, New Delhi-58

September-2018

# Certificate

This is to certify that the work, presented is a bona fide work carried out by **Palak Singhal**, Enrollment No. - 02196303116, in the partial fulfilment of the degree Btech in Information Technology submitted to esteemed institute, Maharaja Surajmal Institute of Technology, Janakpuri, is an authentic record of my work carried out from 18th June to 31st July, 2018 under the guidance of Mr. Yogender (Android Instructor) from Kyrion, WAC and full filed all the requirements for submission of the project and report and has successfully completed training in Android with sincerity and punctuality Conducted by WAC (What After College) in association with Kyrion.

I got sufficient knowledge in the field of Android from my mentors and understood the importance of self-work and enthusiasm to complete it.

**Mr. Sitender Malik**
**(Assistant Professor)**

**Mr. Manoj Malik**
**(H.O.D. IT, MSIT)**

# Acknowledgement

I want to express my profound gratitude to our college Maharaja Surajmal Institute of Technology, Janakpuri for providing us an opportunity to undertake and complete such an interesting project report.

I am very thankful to our respected *Director,* Prof. K.P. Chaudhary sir for the confidence he had on us regarding this project. We are very much obliged to our respected *Head of Department (H.O.D.),* **Mr. Manoj Malik Sir** for inspiring and motivating us to bring out a successful project.

I am very grateful to our respected *faculty,* **Mr. Sitender sir** (Assistant Professor, MSIT) who was our backbone and our guide throughout this project. Without his assistance, constant guidance, cordial support and valuable information we wouldn't have completed this project within a short period of time.

It would be really very unfair without mentioning my friends and family. The immense love, moral support and blessings they have given is truly unmeasurable.

# Contents

# Images

# Abstract

The era of mobile technology opens the windows to the android apps. The websites are vanishing and the mobile phones are emerging. It's the time to change from conventional websites to mobile apps, which has already became the part of our daily routine. From waking up early in morning to late night sleep time we are more or less dependent on apps to make our life easy. Android games are becoming very popular. They are mind refreshing and stress relieving. They provide us a good source of passing free time. Many of these games are so addicting that one can play them for hours without break. **Balloon Popper** is also a gaming App. It improve our reflexes to act immediately and also makes us stress free. In this game we have to pop all balloons appearing on our screen by touching on them as our finger acts as a pin for those balloons. Background of the app is an image which has many buildings. We are given five chances, if we miss a balloon one pin or chance gets reduced. It has ten levels with ten balloons to pop in each level. With the increase in level, the speed of emerging balloons also increase. After every level completed, we are allowed to take pause and start the game again when we are ready. We are notified if we reach highest reached level and score the highest. It has a balloon popper sound i.e. we will hear a popping sound whenever we pop any balloon. It also has a background music to make us feel more relaxed.

# CHAPTER-1

# COMPANY PROFILE

# 1. ABOUT COMPANY

The summer training was organized by Kyrion and WAC collectively in association with Hewlett Packard for the duration of six weeks i.e. 48 hours of teaching. A lot of courses were offered by the organization with suitable venue. The teaching was really effective and training seems to be successful.

## 1.1 WHAT AFTER COLLEGE (WAC)

### 1.1.1 INTRODUCTION

What After College curates career-enhancing activities for the students and allows experts, institutions and brands to reach out to thousands of students easily. It provides a platform to thousands of students to develop and enhance their skills. It provides a source of quality knowledge to its students.

For students, What After College is the one stop-solution enhance their skills, connect with experts and share their stories.

With over a decade long experience in the education industry, topped with the massive reach to popular colleges all over India and tens of thousands students, What After College is growing as one of the most well known and loved brand in the colleges.

### 1.1.2 AIM

What After College is ambitiously working towards its aim of creating an ecosystem where all the stakeholders in the higher educational industry can mutually and fully benefit from each through a streamlined set of services and products.

They curates career-enhancing activities for the students and allows experts, institutions and brands to reach out to thousands of students easily. It provides a platform to thousands of students to develop and enhance their skills. It provides a source of quality knowledge to its students.

### 1.1.3 VISION FOR BUSINESS

With the aim of creating an education ecosystem that contributes towards the holistic development of students and delivers the meaningful growth to the educators, institutions, businesses and service providers. What After College curates and empowers impactful contents (articles, videos, reports) making sure the content reaches the target group to whom it matters the most.

What After College enables the stakeholders in the higher education segment to make informed decisions, connect with each other and provides a bouquet of services to help them flourish.

### 1.1.4 VISION FOR STUDENTS

They aim to empower students by helping them understand the different career options and attaining their dream career by equipping them with the right set of skills. They do this through offline events, workshops, seminars, talks, expert interviews, informative content and building a community. They act as a one stop solution for career paths, guidance and execution to students who are confused about their careers in this world of ever increasing competition.

### 1.1.5 SERVICES

What After College has structured and conceptualized a series of key activities and products to bring effective and industry based solutions to the market that allow us to fully explore and tap into the opportunities that this industry provides.

Some of them are:-

- Quality Admission Solutions:-
    I. Student Engagement through Experiential Marketing:
    II. Lead generation through Digital Marketing.
- Revenue Generation for Colleges:-

I. Skill Development Academy

- Brand Focused Activities

### 1.1.6 STRONG IP'S

What After College has built a strong I.P. in the form of branded skill development workshop, talk series and awareness drive.

## 1.2 KYRION

### 1.2.1 VISION

Team at Kyrion believes that opportunities are borne out of imperfections. Their Education Entrepreneurship Firm, aims to deliver the high-end security solutions and training to the students and corporates to prepare them to meet the challenges of the IT security industry.

### 1.2.2 INOVATORS

Since 1983, chcharity has driven evolution in charity websites: they pioneered real-time news distribution to financial terminals in 1990's, they were first to provide their customers with the ability to post news to their social media channel at the same time as their traditional channels, and they continue to lead the industry with news release reporting and analytics that is critical to our customers' success.

They take pride in being responsive, making a difference at a critical point in peoples' lives. They have been doing this since 1944, working with veterans of every conflict, and we envisage continuing doing so for the 'long-haul' – supporting all future generations of our soldiers and their dependents.

# CHAPTER – 2

# TECHNOLOGY & TOOLS USED DURING INTERNSHIP

# 2. TECHNOLOGY & TOOLS

The era of mobile technology opens the windows to the android apps. The websites are vanishing and the mobile phones are emerging. It's the time to change from conventional websites to mobile apps, which has already became the part of our daily routine

IT Sector is flooded with various technologies for developing mobile apps but using Java for developing android apps is quite popular and the most used technique. So I learned Java basics, Material Designing, Event handling, Intents, Activity & Fragments, and xml in the summer training.

The most popular tool used for building android applications is Android Studio. **Android Studio** is the official IDE for Android application development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, **Android Studio** offers even more features that enhance your productivity when building **Android** apps,

## 2.1 JAVA BASICS

### 2.1.1 WHAT IS JAVA?

Java is a **programming language** and a **platform**. Java is a high level, robust and secured and object oriented language.

**Platform:** Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

### 2.1.2. USES OF JAVA

According to Sun, 3 billion devices run java. There are many devices where Java is currently used. Some of them are as follows:

- Desktop Applications such as acrobat reader, media player, antivirus etc.
- Web Applications such as javatpoint.com etc.
- Enterprise Application such as banking application

- Mobile
- Embedded system
- Smart Cards
- Robotics
- Games etc.

## 2.1.3. TYPES OF JAVA APPLICATIONS

There are mainly 4 types of applications that can be created using java programming:

- **Standalone Application**: Ex Media player, Antivirus etc.
- **Web Application**: An application that runs on server side and creates dynamic pages is called web application. Currently JSP, Serverlet, JSF etc. are used for web application.
- **Enterprise Application**: An application that is distributed in nature, such as Banking Applications etc. is called enterprise application. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise application.
- **Mobile Application:** An application that is created for mobile devices. Currently Android and Java ME are used for mobile applications.

## 2.1.4 JAVA IDE

A Java IDE (for *Integrated Development Environment*) is a software application which enables users to more easily write and debug Java programs. Many IDEs provide features like syntax highlighting and code completion, which help the user to code more easily.

Some of famous java IDEs are:

- **Eclipse:** Eclipse is a Free and Open Source IDE, plus a developer tool framework that can be extended for a particular development need. IBM was behind its

development, and it replaced IBM Visual Age tool. The idea was to create a standard look and feel that can be extended via plugins. The extensibility eclipse from other IDEs. The goals for Eclipse are to give a standard IDE for developing code and to give a starting point and the same look and feel for all other sophisticated tools built on eclipse.

- **NetBeans:** The NetBeans IDE is a free and Open Source IDE for software developers. The IDE runs on many platforms including Windows, GNU/Linux, Solaris and Mac OS X. It is easy to install and use straight out of the box. You can easily create Java applications for mobile devices using Mobility Pack in NetBeans. With NetBeans 6.0, the IDE has become one of the most preferred development tools, whether it be designing a Swing UI, building a mobile application, an enterprise application or using it as a platform for creating your own IDE.
- **BlueJ:** BlueJ is an IDE that includes templates and will compile and run the applications for you. BlueJ is often used by classes because it is not necessary to set classpaths. BlueJ has its own sets of libraries and you can add your own under preferences. That sets the classpath for all compilations that come out of it to include those you have added and the BlueJ libraries.
- **Other IDEs: -** Some other IDEs are Geany, JDeveloper, Visual Café, Zeus, Gel etc.

### 2.1.5 HISTORY OF JAVA

**The history of Java** is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated this project to develop a language for digital devices such as set-top boxes, televisions etc. But, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted and Dynamic".

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

1) **James Gosling**, **Mike Sheridan**, & **Patrick Naughton** initiated Java language project in June 1991. The small team of sun engineers called **Green Team**.
2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.
3) Firstly, it was called **"Greentalk"** by James Gosling and file extension was .gt.
4) After that, it was called Oak and was developed as a part of the Green project.
5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.
6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.
7) **Why had they chosen java name for java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, cool, unique, and easy to spell and fun to say. Java is an island of Indonesia where first coffee was produced (called java coffee).
8) Notice that Java is just a name not an acronym.
9) Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.
10) In 1995, Time magazine called **Java one of the Ten Best Products of 1995.**
11) JDK 1.0 released in (January 23, 1996).

## 2.1.6 FEATURES OF JAVA

The main objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some awesome features which play important role in the popularity of this language. The features of Java are also known as java *buzzwords*.

A list of most important features of Java language are given below.

- **Simple:** Java is very easy to learn and its syntax is simple, clean and easy to understand. According to Sun, Java is a simple programming language because Java syntax is based on C++. Java has removed many confusing and rarely-used features e.g. explicit pointers, operator overloading etc. There is no need to remove unreferenced objects because there is Automatic Garbage Collection in java.
- **Object-Oriented:** Java is object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior. Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
- **Platform Independent:** Java is platform independent because it is different from other languages like C, C++ etc. which are compiled into platform specific machines while Java is a write once, run anywhere language.

  A platform is the hardware or software environment in which a program runs.
- **Secured:** Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because there is no explicit pointer and Java Programs run inside virtual Machine Sandbox.
- **Robust:** Robust means strong. It uses strong memory management. There are lack of pointers that avoids security problem. There is exception handling and type checking mechanism in java. All these points makes java robust.

- **Multi-threaded:** A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares common memory. Threads are important for multi-media, web apps etc.

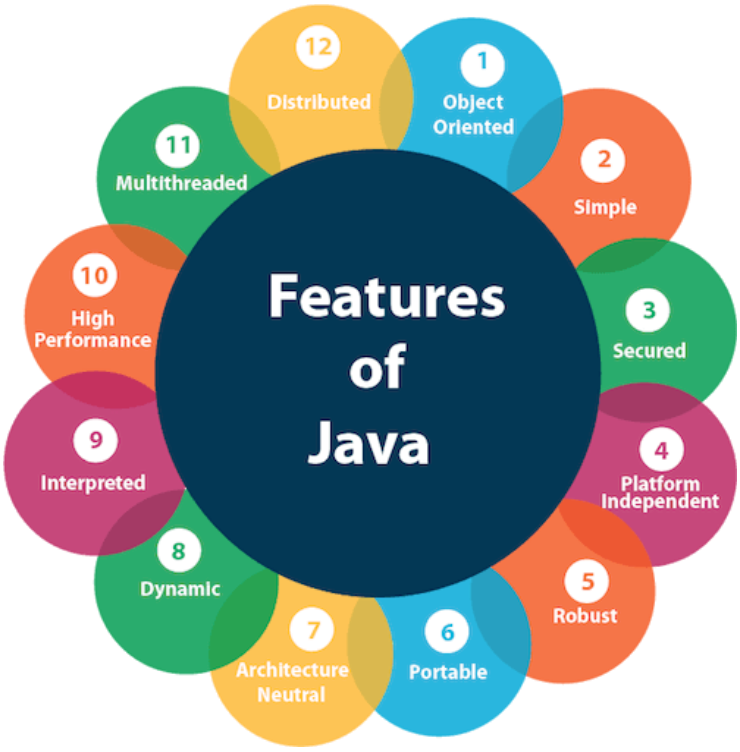


**Fig. 2.1 Features of java**

- **High-performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g. C++).
- **Portable:** Java is portable because it facilitates you to carry the java bytecode to any platform. It doesn't require any type of implementation.
- **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages i.e. C, C++.

## 2.2 XML LAYOUTS

### 2.2.1 LAYOUT

A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects,

The View objects are usually called "widgets". The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout.

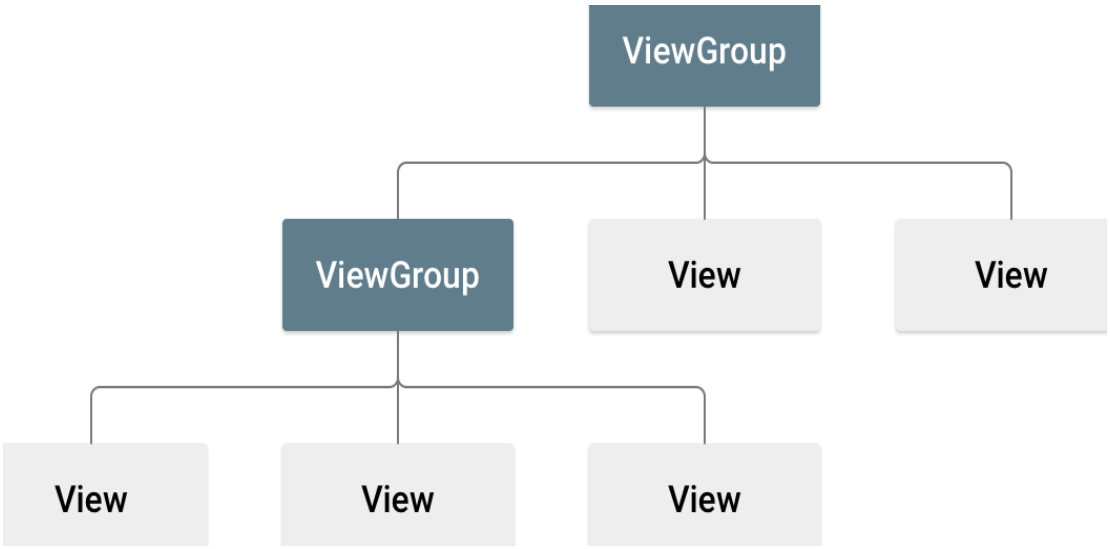


**Fig.2.2 Illustration of a view hierarchy, which defines a UI layout**

## 2.2.2 TYPES OF LAYOUTS

We have different types of layouts:

- **Linear Layout**: It is a view group that aligns all children in a single direction, vertically or horizontally.
- **Relative Layout: It** is a view group that displays child views in relative positions.
- **Table Layout:** It is a view that groups views into rows and columns.
- **Absolute Layout:** It enables you to specify the exact location of its children.
- **Frame Layout: It** is a placeholder on screen that you can use to display a single view.
- **Liat View:** It is a view group that displays a list of scrollable items.
- **Grid View:** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

The Android framework gives us the flexibility to use either or both of these methods to build your app's UI. For example, us can declare your app's default layouts in XML, and then modify the layout at runtime.

## 2.2.3 WRITE XML

Using Android's XML vocabulary, we can quickly design UI layouts and the screen elements they contain, in the same way we create web pages in HTML — with a series of nested elements.

Each layout file must contain exactly one root element, which must be a View or ViewGroup object. Once you've defined the root element, you can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout. For example, here's an XML layout that uses a vertical LinearLayout to hold a TextView and a Button:

After declaring your layout in XML, save the file with the .xml extension, in your Android project's res/layout/ directory, so it will properly compile.

### 2.2.4 LOAD XML RESOURCES

When you compile your app, each XML layout file is compiled into a View resource. You should load the layout resource from your app code, in your Activity.onCreate() callback implementation. Do so by calling setContentView(), passing it the reference to your layout resource in the form of:R.layout.*layout_file_name.* The onCreate() callback method in your Activity is called by the Android framework when your Activity is launched.

### 2.2.5 VIEWS

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, imageview etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup**objects or you can declare your layout using simple XML file **main_layout.xml** which is located in the res/layout folder of your project.

### 2.2.6 ATTRIBUTES

Every View and ViewGroup object supports their own variety of XML attributes. Some attributes are specific to a View object (for example, TextView supports the textSize attribute), but these attributes are also inherited by any View objects that may

extend this class. Some are common to all View objects, because they are inherited from the root View class (like the id attribute). And, other attributes are considered "layout parameters," which are attributes that describe certain layout orientations of the View object, as defined by that object's parent ViewGroup object.

### 2.2.7 ID

Any View object may have an integer ID associated with it, to uniquely identify the View within the tree. When the app is compiled, this ID is referenced as an integer, but the ID is typically assigned in the layout XML file as a string, in the id attribute. This is an XML attribute common to all View objects and it is used very often.

The at-symbol (@) at the beginning of the string indicates that the XML parser should parse and expand the rest of the ID string and identify it as an ID resource. The plus-symbol (+) means that this is a new resource name that must be created and added to our resources (in the R.java file). There are a number of other ID resources that are offered by the Android framework. When referencing an Android resource ID, you do not need the plus-symbol, but must add the android package namespace.

With the android package namespace in place, we're now referencing an ID from the android.R resources class, rather than the local resources class.

### 2.2.8 LAYOUT PARAMETERS

XML layout attributes named layout_*something* define layout parameters for the View that are appropriate for the ViewGroup in which it resides.

Every ViewGroup class implements a nested class that extends ViewGroup.LayoutParams. This subclass contains property types that define the size and position for each child view, as appropriate for the view group.

every LayoutParams subclass has its own syntax for setting values. Each child element must define LayoutParams that are appropriate for its parent, though it may also define different LayoutParams for its own children.

All view groups include a width and height (layout_width and layout_height), and each view is required to define them. Many LayoutParams also include optional margins and borders..

### 2.2.9 LAYOUT POSITION

The geometry of a view is that of a rectangle. A view has a location, expressed as a pair of *left* and *top*coordinates, and two dimensions, expressed as a width and a height. The unit for location and dimensions is the pixel.

It is possible to retrieve location of a view by invoking the methods getLeft() and getTop(). The former returns the left, or X, coordinate of the rectangle representing the view. The latter returns the top, or Y, coordinate of the rectangle representing the view. These methods both return the location of the view relative to its parent.

In addition, several convenience methods are offered to avoid unnecessary computations, namely getRight()and getBottom(). These methods return the coordinates of the right and bottom edges of the rectangle representing the view.

**2.3 Linear Layout**

**2.4 Relative Layout**


**2.5 Web View**

## 2.3 ANDROID - EVENT HANDLING

### 2.3.1 EVENTS

**Events are a useful way to collect data about a user's interaction with interactive components of Applications.** Like button presses or screen touch etc. The Android framework maintains an event queue as first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management –

- **Event Listeners** − An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- **Event Listeners Registration** − Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- **Event Handlers** − When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

### 2.3.2. EVENT LISTENERS

Some Event listeners are:

- **OnClickListener( ):** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick( ) event handler to handle such event.
- **OnLongClickListener( )**: This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick( ) event handler to handle such event.
- **OnFocusChangeListener( )**: This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange( ) event handler to handle such event.
- **OnTouchListener( )**: This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch( ) event handler to handle such event.
- **OnMenuItemClickListener( )**: This is called when the user selects a menu item. You will use onMenuItemClick( ) event handler to handle such event.
- **onCreateContextMenuItemListener( )**: This is called when the context menu is being built(as the result of a sustained "long click).

## 2.4 ANDROID STUDIO

### 2.4.1 INTRODUCTION

**Android Studio** is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based OS. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage

starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.2, which was released in September 2018.

### 2.4.2 FEATURES

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

### 2.4.3 SYSTEM REQUIREMENTS

- **OS version:** Microsoft Windows 7/8/10 (32-bit or 64-bit), 64-bit required for native debugging, Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra), GNOME or KDE desktop Linux (64 bit capable of running 32-bit applications)(GNU C Library (glibc) 2.19+).
- **Ram:** 3 GB RAM min, 8 GB RAM recommended; plus 1 GB for Android Emulator.
- **Disc Space**: 2 GB of available disc space minimum, 4 GB recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image).
- **Java version:** Java Development Kit (JDK) 8, use of bundled Open JDK is recommended.

- **Screen resolution:** 1280×800 minimum screen resolution.

### 2.4.4 PROJUCT STRUCTURE

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

Each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings etc.

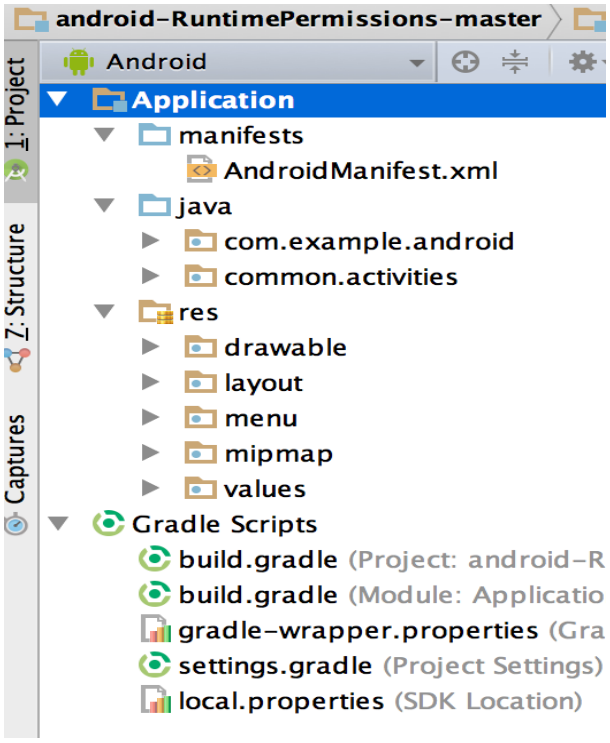


**Fig. 2.6 The Project Files in Android View**

## 2.4.5 THE USER INTERFACE

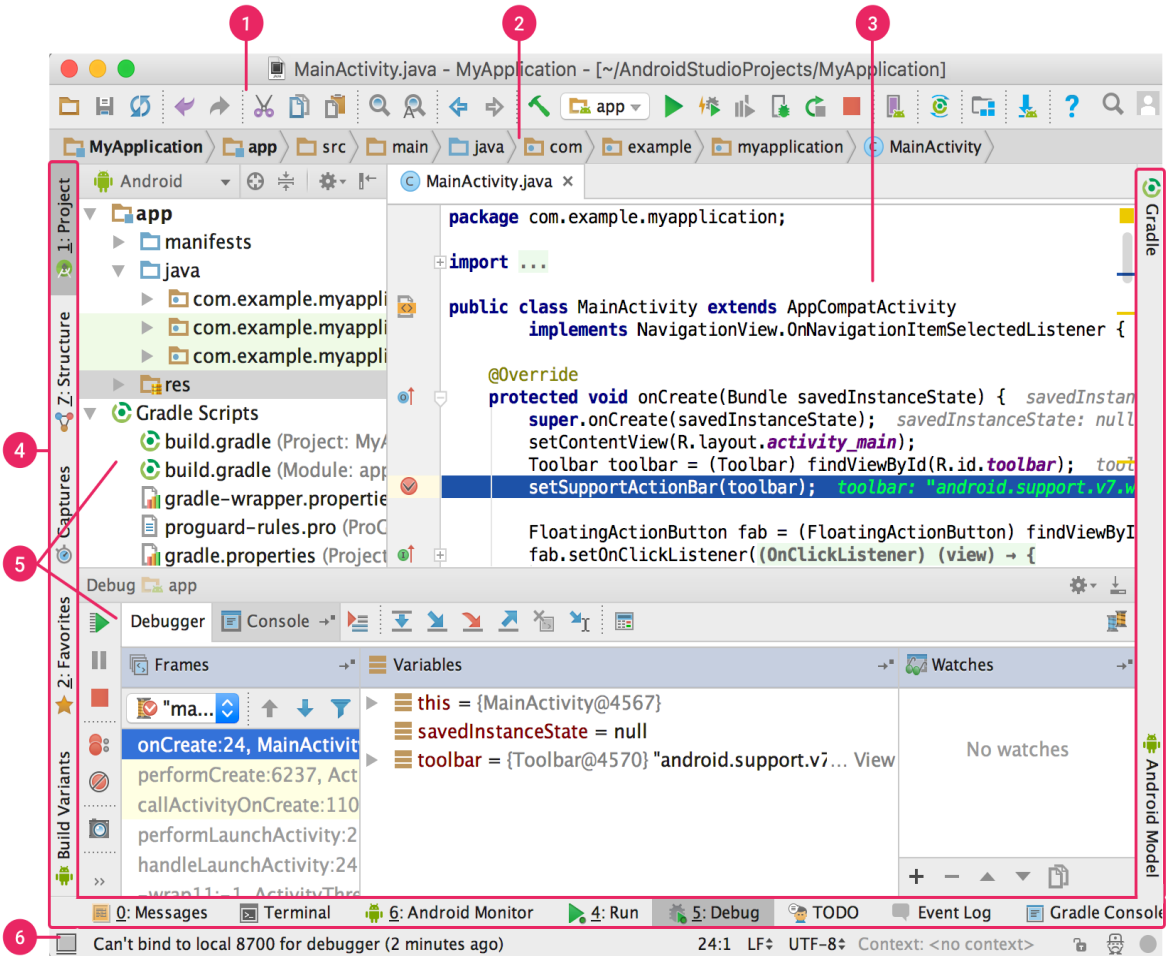


**Fig 2.7 Android Studio Main Window**

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

# CHAPTER- 3

# PROJECT

# 3. PROJECT

## 3.1 INTRODUCTION

In past few years technology has gone through rapid evolution. Smart phones have first replaced cell phones which gave a boost to the Android Market in IT Sector. The era of mobile technology opened the windows to the android apps. The websites are vanishing and the mobile phones are emerging. It's the time to change from conventional websites to mobile apps, which has already became the part of our daily routine. From waking up early in morning to late night sleep time we are more or less dependent on apps to make our life easy. Mobile apps are in great demand. Market is flooded with millions of apps. Today we have atleast one app for any work. Smart phone have become mini computers due to mobile apps. Mobile applications often stand in contrast to desktop applications which run on desktop computers, and with web applications which run in mobile web browsers rather than directly on the mobile device.

Mobile application development requires the use of specialized integrated development environments. Mobile apps are first tested within the development environment using emulators and later subjected to field testing. Emulators provide an inexpensive way to test applications on mobile phones to which developers may not have physical access.

Balloon Popper App is a basic Android Game App. It improve our reflexes to act immediately and also makes us stress free. In this game we have to pop all balloons appearing on our screen by touching on them as our finger acts as a pin for those balloons. Background of the app is an image which has many buildings. We are given five chances, if we miss a balloon one pin or chance gets reduced. It has ten levels with ten balloons to pop in each level. With the increase in level, the speed of emerging balloons also increase. After every level completed, we are allowed to take pause and start the game again when we are ready. We are notified if we reach highest reached level and score the highest. It has a balloon popper sound i.e. we will hear a popping sound whenever we pop any balloon. It also has a background music to make us feel more relaxed.

## 3.2 PROJECT DESIGN

### 3.2.1 APP LAUNCH ICON & OTHER RESOURCES

Android Studio provides a default icon to app which is the Android symbol itself. The icon can be found in mipmap folder in res directory of 5 sizes for different sizes of devices. To set a different icon just add the new icon in mimap folders and change the icon name in AndroidManifest.xml in manifests folder. Also place the other resource in required directory i.e. audio files in raw folder etc.

**Fig 3.1 Default Icon**

**Fig 3.2 Changed Icon**

### 3.2.2 DECIDING THE UI LAYOUT

Good **app UI design** can play important role in the success of our android app. Interface design works on the layout of functionality of interfaces so selecting the layout is the first step for a successful UI implementation.

Balloon Popper has a **Relative Layout**, is a view group that displays child views in relative positions. Since the app has a lot of views that are needed to be placed relatively. It may have further relative and relative layouts along with various viewgroups.

### 3.2.3 VIEWS USED

UI of the apps needs to display buttons, text, icons and images. This can be done by using various views. Following are the views used in Balloon Popper:

- **ImageView** : Display image. It has been used to display the pin & pin_off icon on screen from the drawable folder.
- **TextView**: Displays text. It has been used to display the text “Score”, “Level”, scores and levels.
- **Button**: Creates button with text. It has been used to create a button to start and stop the game.

By using proper and necessary attributes position all the views in their correctly. ID attribute is used to provide a unique identity to views so that they can be referred and altered.

Here, with the finishing of all these, the designing part of our app is completed. But it will give us only a Image like screen with a desired view. There wull be no functioning at all.

## 3.3 IMPLEMENTING JAVA

### 3.3.1 BALLOONS

As the name Balloon Popper suggests, we obviously need balloons to pop. So a separate class file has been created for animating colorful balloons i.e. Balloon.java. In this file create a class Balloon. There are following functions of this class:

- Constructor of this class sets image of balloon from drawable folder to animatorlistener and the color of that particular balloon and converts the height and width from pixel to dp using a separate class file which is PixelHelper.java.
- To handle the balloon animation i.e. handles the duration, balloons appearing on screen etc.
- To allow the balloon to float till not popped and return false if remain unpopped.
- To pop the balloon ontouch and set variable popped.

### 3.3.2 SHARED PREFRENCES

Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow us to save and retrieve data in the form of key, value pair.

In order to use shared preferences, we need to import it and have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences. The first parameter is the key and the second parameter is the MODE. Apart from private there are other modes available.

In this project SharedPreferences is used to store the score and levels and also to update the high score and max reached levels by making a separate java class file as PreferencesHelper.java. Modes used in project:

- **apply()**: It is an abstract method. It will commit your changes back from editor to the sharedPreference object you are calling
- **putInt(String key, int value)**: It will save a integer value in a preference editor

### 3.3.3 DIALOG FRAGMENT

A fragment that displays a dialog window, floating on top of its activity's window. This fragment contains a Dialog object, which it displays as appropriate based on the fragment's state. Control of the dialog (deciding when to show, hide, dismiss it) should be done through the API here, not with direct calls on the dialog.

Implemen Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle) to supply the content of the dialog. Alternatively, they can overrideonCreateDialog(Bundle) to create an entirely custom dialog, such as an AlertDialog, with its own content.

In this project, dialogFragment has been used to display high score and highest reached levels by declaring a class MyAlertiDalog in separate java class file.

### 3.3.4 PLAYING AUDIO

AudioManager provides access to volume and ringer mode control.

The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw

resources), from standalone files in the filesystem, or from a data stream arriving over a network connection, all using MediaPlayer APIs.

One of the most important components of the media framework is the MediaPlayer class. An object of this class can fetch, decode, and play both audio and video with minimal setup. It supports several different media sources such as:

- Local resources
- Internal URIs, such as one you might obtain from a Content Resolver
- External URLs (streaming)

Both audiomanagr and multimedia are used to play background audio and popping sound in app in a separate java class life i.e. SoundHelper.

### 3.3.5 Main Class

All the above mentioned class files are used in main to perform their role along with the functioning of main itself. Main class file, as the name suggests is a remote control for all other java classes. It decides which function the other class files perform and when.

# CHAPTER – 4

# SCREENSHOTS OF PROJECT


**Fig. 4.1 App Icon**

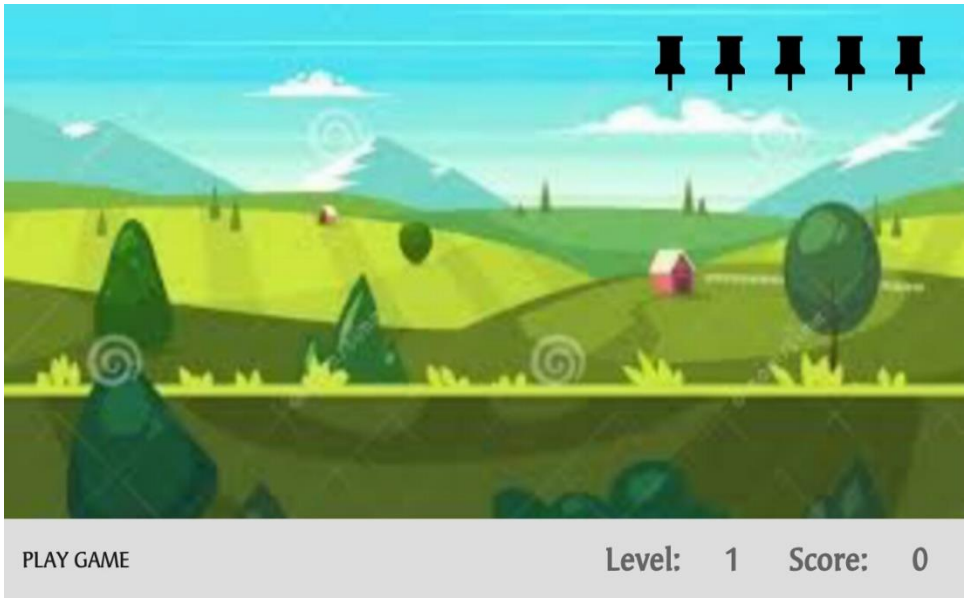


**Fig. 4.2 Launching screen of App**

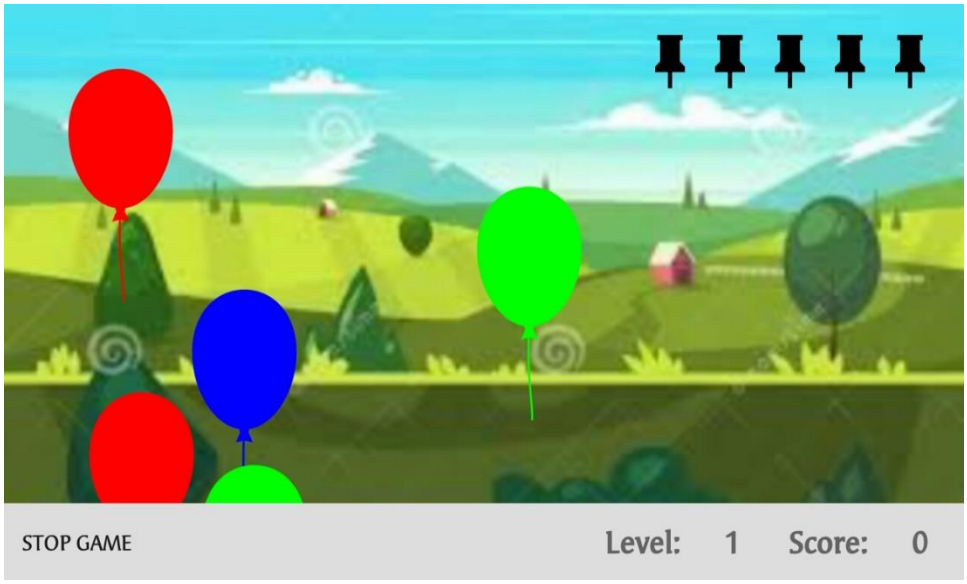


**Fig. 4.3 After Starting Game**

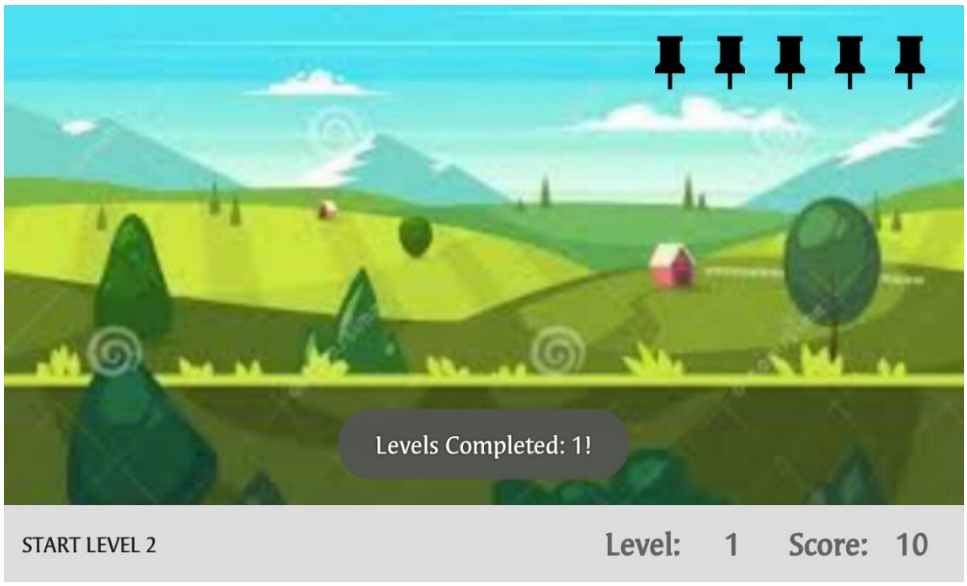


**Fig. 4.4 Finishing Level**

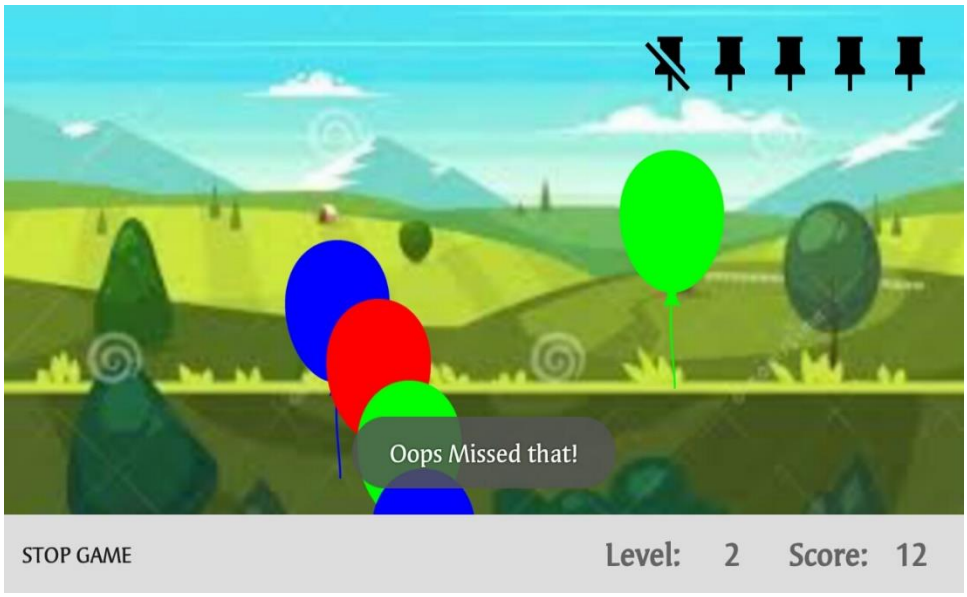


**Fig. 4.5 Missing a ballon**

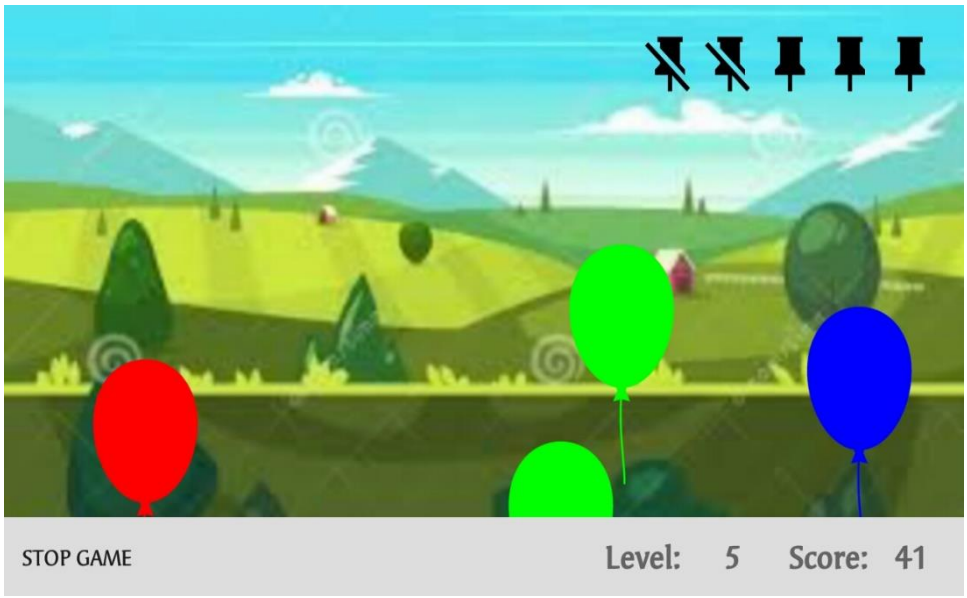


**Fig. 4.6 After missing 2 balloons**

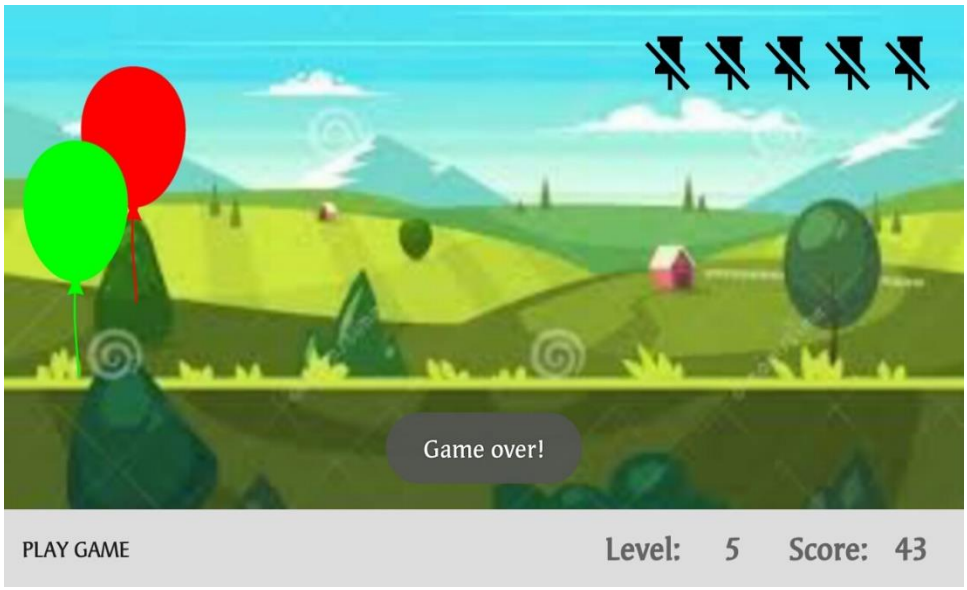


**Fig. 4.7 Missing 5 Balloons**

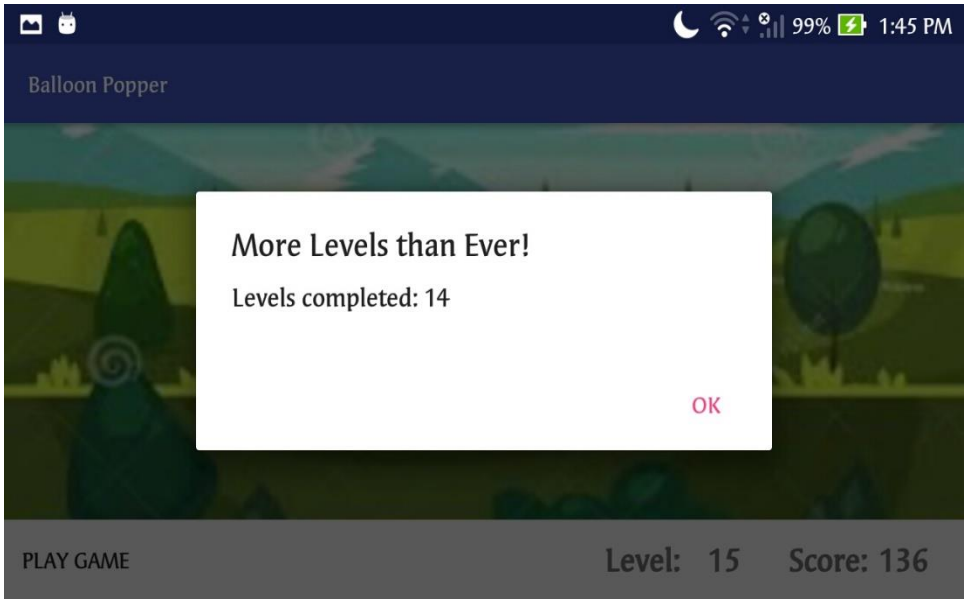


**Fig. 4.8 For highest level reached**

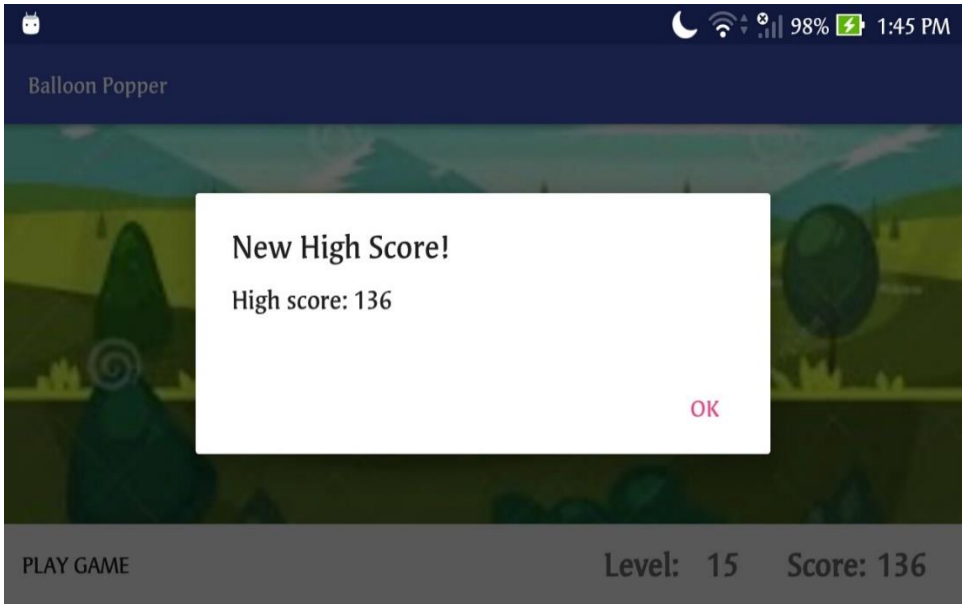


**Fig. 4.9 High Score**

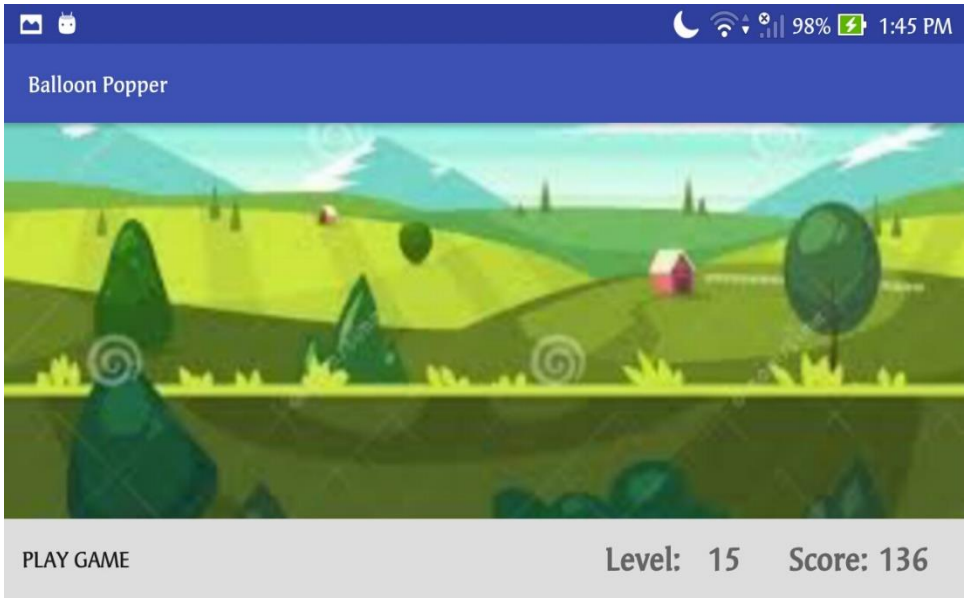


**Fig. 4.10 For starting new game**

# CHAPTER – 5

# CONCLUSION

The training proved pretty beneficial for me as I developed new skills for developing android apps. During this training I learned a lot:

- Java Basics
- Android Studio Basics
- XML Layouts
- Material Designing and Advanced Layouts
- Java Event Handling
- Shared Preferences
- Fragments

I learned all the above things. Before this training, I had no idea about core java and Android for mobile apps. Although I had a little bit knowledge about java and its syntax. Learning android was a new and good experience. Now I know more than before and can develop mobile applications. Balloon Popper App that I made has used all of these concepts and some more. It is a game app in which we have to pop balloons and score best. There were many obstacles in the way of developing this app but there is always a way. Although it is a very basic game app but is really engaging. In this mobile era where websites and online games are vanishing and mobile apps and games are emerging, Android Development can give a good start to career.

# BIBLIOGRAPHY


1. https://lynda.com
2. https://google.com
3. https://udacity.com
4. https://javatpoint.com
5. https://androidstudio.org/documentation
6. https://github.com
7. https://developer.android.com/guide
8. https://tutorialspoint.com

# APPENDICES

- **Animation:** Animation is the rapid display of a sequence of images to create an illusion of movement.
- **Android:** Android is a Linux-based operating system.
- **Class: Class** is a user **defined** data type, which holds its own data members and member functions, which can be accessed and used by creating instance of that **class**.
- **Embedded System:** An **embedded system** is a programmed controlling and operating **system** with a dedicated function within a larger mechanical or electrical **system**, often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts.
- **System:** A system is a set of interacting or interdependent components forming an integrated whole or a set of elements and relationships which are different from relationships of the set or its elements to other seta or elements.
- **Operating System:** An **operating system** (OS) is **system** software that manages computer hardware and software resources and provides common services for computer programs.
- **UI:** User Interface.
- **URL:** A **Uniform Resource Locator** (**URL**), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network
- **UX:** User Experience (UX) involves a person's emotions about using a particular product, system or service.