IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. I (Nov – Dec. 2015), PP 30-40
www.iosrjournals.org

The Importance of Object-Oriented Programming in This Era of Mobile Application Development

Onu, F.U*, Ikedilo, O.E*, Nwoke, B.O* and Okafor, P*
**Department of Computer Science, Ebonyi State University, Abakaliki*

Abstract: *In the past two decades object oriented programming has become the dominant programming OOP paradigm used by application developers. Object oriented programming scales very well, from the most trivial of problems to the most complex tasks. It provides a form of abstraction that resonates with techniques people use to solve problems in their everyday life. Although the fundamental features of what we now call object-oriented programming were invented in the 1960's, object oriented languages really came to the attention of the computing public in the 1980's. Software producers rush to release object oriented versions of their products. Countless books and special issues of academic and trade journals have appeared on the subject. Object-oriented programming is being greeted with even more enthusiasm than we saw heralding earlier revolutionary ideas, such as structured programming". The intent in this publication is to emphasize and explain the importance of object-oriented programming in this era of mobile development, and in doing so to illustrate the following two propositions: OOP is a revolutionary idea, totally unlike anything that has come before in programming, OOP is an evolutionary step, following naturally on the heels of earlier programming techniques.*
Keywords: *Android, Object-oriented Programming, Mobile app, App store, SDK, Smartphones*

I. Introduction

The future of computing is moving towards mobile devices. As such, they are gradually replacing PC's as the most common method for accessing the Internet [Stephanides. 2002]. A mobile device is basically any handheld computer. It is designed to be extremely portable, often fitting in the palm of your hand or in your pocket. Some mobile devices are more powerful, and they allow you to do many of the same things you can do with a desktop or laptop computer. Currently, the two most popular platforms are Google's Android and Apple's IOS. There are, however, several other platforms, which add to the market fragmentation. Among those, Windows Phone is considered to be the most promising. Besides, mobile application development has also become an important area of scientific studies.

1.2 Motivation Of The Study

In Mobile App Development and Software Development in general, it has been observed that developers motivations, challenges, and future plans for app development are surrounded by the following prime factors; a sense of achievement and not money is the main motivating factor in software development (of course, a paycheck is always appreciated). This can be viewed under these group of persons: Hobbyists, the Explorers, the Hunters, the Guns for Hire, the Product Extenders, the Digital Content Publishers, the Gold Seekers and the enterprise IT developers. Each had their own unique motivations, platforms, and modes of workability.

"We find that Explorers and Hobbyists, those seeking to learn, have fun and self-improve, make up 33% of the mobile developer population but only 13% of the app economy revenues. These segments prefer – more than average – BlackBerry 10, Windows Phone as a platform, as these are more often associated with experimentation and learning. [W. Boswell 2013]

Alexander Sergeev outlined 10 undeniable facts to be a mobile app developer;

- There's plenty of opportunity. 5.2 billion People use mobile devices. 30% (1.6 billion) use smartphones.
- It's expected that mobile app revenues from app stores around the world will reach $45.4 billion in 2015 and will grow to $76.52 billion in 2017 according to forecasts from Statista.
- The GSMA's latest study "Smartphone forecasts and assumptions, 2007-2020" suggests that by 2020 there will be 6 billion smartphone connections. Just think of it. It's only 5 years from now.
- There will be 167.05 billion of free mobile app downloads throughout the world in 2015. The number will increase to 211.31 billion free downloaded apps in 2016 as predicted by Statista.
- Emerging markets like Indonesia, China, Brazil and India will outdo developed nations in smartphone and wider technology growth in 2015. It represents a severe growth potential beyond developed markets.

- Owners of tablets tend to spend 13% more time in apps and open them 11% more frequently on those bigger devices. What's more, phablet users come back to mobile apps 38% more often and stay in them 10% longer than users of regular smartphones screens. This fact is provided by analytics firm Localytics.
- Among all the digital advertising formats in the U.S, mobile format is growing the fastest according to new data from BI Intelligence.
- As reported by Comscore, of all the time consumers spend online 52% do so using smartphone and tablet apps.
- In 2018 mobile ad spend will top nearly $42 billion as stated in BI Intelligence. It will also increase by a five-year compound annual growth rate (CAGR) of 43 percent from 2013.
- It also should be mentioned that in-app ads are more efficient than mobile web ads. Measurement company Medialets has found out that in-app click-through rates averaged .56 percent globally in comparison with .23 percent for mobile web ads in the first half of the year.

1.3 Objectives Of The Study

The objective of this publication is to emphasize and explain the importance of object-oriented programming in this era of mobile development. This will help developers in choosing carriers in mobile application development and software development in general.

1.4 Scope Of The Study

This research does not show details on logics involves in mobile app development. It did not also include any code or algorithm on how a particular application is developed. It only explicitly highlights on the importance of Objected oriented Programming in this Era of Mobile Application over any other development platform.

II. Review Of Related Literature

2.1 History Of Mobile Applications

Mobile communication has become so integrated into our everyday lives that many people feel uncomfortable without a cell/mobile phone. In their earlier stage, the most popular functions of mobile phones were to make and receive calls and to send text messages. Smart phone does more than these. It is a multifunctional device that not only communicates, but helps its users to learn, earn, and have fun. This is made possible by the development of mobile applications. Mobile applications date back to the end of the twentieth century. Typically, they were small arcade games, ring tone editors, calculators, calendars, and so forth. The beginning of the new millennium saw a rapid market evolution of mobile content and applications. Operating systems for smart phones (Windows Mobile, Symbian, RIM, Android, Mac iOS), are open to the development of third-party software, unlike the conventional programming environment of standard cell phones.

Manufacturers tried to make their products more attractive for customers by introducing more and more applications. But quality matters as well. Cell phone development needs to be easy and intuitive. Every company tries to facilitate the process of development so that users are able to customize their devices. Motivation: Juniper Research estimates in 2014 the direct and indirect revenues from sales of mobile applications will total 25 billion dollars. Mobile users demand more choice, more opportunities to customize their phones, and more functionality. Mobile operators want to provide value-added content to their subscribers in a manageable and lucrative way. Mobile developers want the freedom to develop the powerful mobile applications users demand without restrictions. Finally, handset manufacturers want a stable, secure, and affordable platform to power their devices.

2.2 The First Apps

First-generation mobile phones were designed and developed by the handset manufacturers. Competition was fierce and trade secrets were closely guarded. They didn't want to expose the secrets of their handsets, so they developed the phone software in-house. Developers that weren't part of this inner circle had no opportunity to write applications for the phones. It was during this period the first "time-waster" games begin to appear. Nokia was famous for putting the 1970s video game Snake on some of its earliest phones. Other followed, adding games like Pong, Tetris, and Tic-Tac-Toe. These early phones changed the way people thought about communication. As mobile phone prices dropped, batteries improved, reception areas grew, and more and more people began carrying these handy devices. Soon mobile phones were more than just a novelty.

Customers began pushing for more features and more games. But handset manufacturers didn't have the motivation or the resources to build every application users wanted. They needed some way to provide a portal for entertainment and information services without allowing direct access to the handset. What better way to provide these services than the Internet?

2.3 Wireless Application Protocol (Wap)

It turned out direct phone access to the Internet didn't scale well for mobile. By the late 90s, professional Web sites were full color and loaded with text, images, and other types of media. They relied on JavaScript and Flash to enhance the user experience and were often designed at 800×600 pixels. Early phones had very small monochrome low-res screens and limited storage and processing power. They couldn't handle the data-intensive operations required by traditional Web browsers. The bandwidth requirements for data transmission were also costly to the user. The Wireless Application Protocol (WAP) standard was developed to address these concerns. WAP was a stripped-down version of HTTP, which is the basic protocol of the World Wide Web [K. Naik 2001]. WAP browsers were designed to run within the memory and bandwidth constraints of the phone. Third-party WAP sites served up pages written in a markup language called Wireless Markup Language (WML). The WAP solution was great for handset manufacturers. They could write one WAP browser to ship with the handset and rely on developers to come up with the content users wanted. The WAP solution was great for mobile operators.

They could provide a custom WAP portal directing their subscribers to the content they wanted to provide, and wallow in the high data charges associated with browsing. But there was one problem – Developers and content providers didn't deliver, except in a limited way. Users accessed the news, stock market quotes, and sports scores on their phones. Some of the most popular commercial WAP applications that emerged during this time were simple wallpaper and ring tone catalogues, allowing users to personalize their phones for the first time. Commercializing WAP applications was difficult, and there was no built-in billing mechanism. Users browsed a WAP site and requested a specific item, then filled out a simple order form with their phone number and the handset model of their phone. It was up to the content provider to deliver an image or audio file compatible with the phone. Payment was handled through various premium-priced delivery mechanisms such as Short Message Service (SMS), Enhanced Messaging Service (EMS), Multimedia Messaging Service (MMS), and WAP Push.

2.4 Mobile Platforms

WAP fell short of commercial expectations, except in Japan and a few other places. Small handset screens were too small for surfing. Reading a sentence fragment at a time and then waiting seconds for the next segment to download ruined the user experience, especially because every second of downloading was charged to the user. Critics began to call WAP "Wait and Pay." It came as no surprise when users wanted more and they will always want more [Peymandoust 2002]. Writing graphic-intensive video game applications with WAP was nearly impossible. The kids most likely to personalize their phones with wallpapers and ring tones looked at their portable gaming systems and asked for a device that was both a phone and a gaming device or a phone and a music player. If devices like Nintendo's Game Boy could provide hours of entertainment with only five buttons, why not just add phone capabilities? Memory was getting cheaper; batteries were getting better and PDAs and other embedded devices were beginning to run compact versions of common operating systems like Linux and Windows. The traditional desktop application developer was suddenly involved in the embedded device market, especially with Smartphone technologies like Windows Mobile, which they found familiar. Handset manufacturers realized that if they wanted to continue to sell their products, they needed to change their protectionist policies regarding handset design and expose their internal workings to some extent. A variety of different proprietary platforms emerged and developers are still actively creating applications for them. One of the first was the Palm OS (now Garnet OS) and RIM Blackberry OS. Sun Microsystems popular Java platform became Java Micro Edition (Java ME). Qualcomm developed its Binary Runtime Environment for Wireless (BREW). Symbian OS was developed by Nokia, Sony Ericsson, Motorola, and Samsung. The Apple iPhone iOS joined the ranks in 2007. Google's Android came along a year later.

Most platforms have associated developer programs that keep the developer communities small, vetted, and under contractual agreements on what they can and cannot do. These programs are often required and developers must pay for them. Each platform has benefits and drawbacks. The truth is no one platform has emerged victorious. Some platforms are best suited for commercializing games and making millions if your company has brand backing. Other platforms are more open and suitable for the hobbyist. No mobile platform is best suited for all possible applications. The mobile phone market has become increasingly fragmented, with all platforms sharing part of the pie. For manufacturers and mobile operators, handset product lines have become complicated fast. Platform market penetration varies greatly by region and user demographic. Instead of choosing just one platform, manufacturers and operators have been forced to sell phones for all the different platforms to compete. Mobile software developers work with different programming environments, different tools, and different programming languages. Porting among the platforms is often costly and not straightforward. Keeping track of handset configurations and testing requirements, signing and certification programs, carrier relationships, and application marketplaces have become complex spin-off businesses of their own.

2.5 Problem Domain And Solution Domain

A problem is a functional specification of desired activities to generate the intended output. A solution is the method of achieving the desired output [San Diego. 1994]. Every problem belongs to a domain of knowledge. The domain is the general field of business or technology in which the user will use the software. Hence, the term problem domain is used in problem solving. The domain or the sector to which the problem belongs defines the problem domain. The problem that specifies the requirement in a particular knowledge domain and the domain experts associated with the task of explaining the requirements belong to the problem domain. Similarly, the solution obtained belongs to the solution domain. The subject matter that is of concern to the computer and the persons associated with the task of devising solution defines the solution domain. The problem domain specifies the scope of the problem along with the functional requirements represented in a high level so that human beings can understand it. The solution domain contains the procedures or techniques used to generate the desired output by a computer. Thus, problem solving is a mapping of problem domain to solution domain. It is the act of finding a solution to a problem. The formulation of a solution for a simple problem is easy. The solution for simple problems may not require any systematic approach. But a complex problem requires logical thinking and careful planning. Generally, the problems to be solved using computers will be reasonably complex.

2.6 Principles Of Graphical User Interface Design

Visual programming can be a good solution to help non-programmers learn programming more easily. Visual programming tools enable people to see and test what they build immediately after putting together the pieces of different components. These tools also create a more enjoyable programming experience by reducing the frustration of getting lost in textual codes and debugging. Since Glinert's (1986) pioneering work on BLOX (a visual programming language consisting of puzzle-like pieces), there have been a few successful visual programming tools available. For example, Scratch is a free tool that makes it easy to create one's own interactive stories, animations, games, music, and art in two-dimension format (Lifelong Kindergarten Group, 2006). Another tool, Alice, is a three-dimensional (3D) programming environment for creating story-telling animations, playing interactive games, or sharing videos on the Web (Carnegie Mellon University, 2008). One great advantage of introducing programming to novices with visual programming languages is that it can help them avoid syntax errors commonly seen in working with textual programming languages. In addition, the drawer analogy used for arranging the puzzle pieces (called "blocks" in App Inventor) with similar function can reduce the need for novices to remember exact textual codes (Turbak et al., 2012), which can greatly reduce the potential cognitive load caused by programming with textual codes (Margulieux, Guzdial, & Catrambone, 2012).

App Inventor (AI) also features drag-and-drop visual programming, which lets designers see how different pieces come together, and how their programming relates to the behaviors of their artifacts/products—the mobile apps (Hsu, Rice, & Dawley, 2012). AI is a free web-based tool that consists of two major elements: Component Designer and Block Editor which together allow users to develop mobile apps running on Android devices. Component Designer lets one design the app's interface and integrate non-visible components (i.e., feature/function not visible to users on the mobile device interface) such as GPS (global positioning system) or sound. Block Editor allows one to program mobile apps' behaviors and to control how apps react under certain circumstances. This tool has great potential for enabling educators with limited programming knowledge and experiences to experiment and design mobile apps that suit their professional needs.

2.7 Designing A Mobile Application

However, developers face several challenges when developing mobile applications. First, the range of mobile platforms requires the knowledge and experience with the different programming languages as well as software development kits (hereinafter referred to as SDK). This consequentially results in a duplication of development effort and maintenance costs [G. Stephanides, 2002]. Second, developers need to consider the physical limitations of mobile devices. Third, due to the low selling price of the mobile applications, developers need to reduce the development costs. On the other hand, if we want to increase reusability, decompose problem into several easily understood segments or enable the future modifications, object-oriented programming (hereinafter referred to as OOP) need to be considered. They represent dependable guidelines that may help ensure good programming practices and write a reliable code.

However, in developing mobile applications, there are specific aspects that need to be considered. The number of users of mobile applications is usually dependent on both the quality of the application itself and the targeted platforms, for which the application was developed. Ideally, it would be best to develop each application for all platforms simultaneously without any adjustment of the source code. To this end, many tools for multi-platform development have been introduced, which support the "build once, deploy everywhere" ideology. Mobile applications, developed in native languages are still preferred among the end users. But, as

already stated, developing a mobile application for several platforms is a time and cost consuming process. The development of mobile applications for multiple platforms requires different environments and knowledge of specific programming languages, depending on the platform. Each platform has its own set of functionalities, rules and requirements, thereby limiting the interoperability of applications and their simultaneous development.
A **mobile app** is a computer program designed to run on mobile devices such as smartphones and tablet computers. Most such devices are sold with several apps included as pre-installed software, such as a web browser, email client, calendar, mapping program, and an app for buying music or other media or more apps. Some pre-installed apps can be removed by an ordinary uninstall process, thus leaving more storage space for desired ones. Where the software does not allow this, some devices can be rooted to eliminate the undesired apps [American Dialect Society. 2010]. pps that are not preinstalled are usually available through application distribution platforms, which began appearing in 2008 and are typically operated by the owner of the mobile operating system, such as the Apple App Store, Google Play, Windows Phone Store, and BlackBerry App World. Some apps are free, while others must be bought. Usually, they are downloaded from the platform to a target device, but sometimes they can be downloaded to laptops or desktop computers. For apps with a price, generally a percentage, 20-30%, goes to the distribution provider (such as iTunes), and the rest goes to the producer of the app. The same app can therefore cost the average Smartphone user a different price depending on whether they use iPhone, Android, or BlackBerry 10 devices.

The term "app" is a shortening of the term "application software". It has become very popular, and in 2010 was listed as "Word of the Year" by the American Dialect Society. In 2009, technology columnist David Pogue said that newer smartphones could be nicknamed "app phones" to distinguish them from earlier less-sophisticated smartphones [Education Conference. 2014]. Although the components of a mobile application may be created by a single developer, an application may typically be built from resources provided by different development roles. An application developer builds the application data and the user interface logic either as an application or as a reusable program that can be used in an application feature. An application assembler gathers different application features into a single application and puts them in a user-friendly, navigable order. An application deployer ensures a controlled application deployment. For example, deployment of MAF applications may require certificates and uploads to public vendor sites such as the Apple App Store or GooglePlay. Depending on the application development team size and your organization, one person may fill many different roles.

Typically, you perform the following activities when building a MAF application:

- Gathering requirements
- Designing
- Developing
- Deploying
- Testing and debugging
- Securing
- Enabling access to the server-side data
- Redeploying
- Retesting and debugging
    - Publishing

The steps you take to build a MAF application may be similar to the following:

- Gathering requirements: Create a mobile use case (or user scenario) by gathering user data that describes who the users are, their essential tasks, and the location or context in which they perform them. Consider such factors as the type of information required to complete a task, the information that is available to the user, and how it is accessed or delivered.
- Designing: After you construct a use case, create a wireframe that illustrates all of the steps (and associated user views) in the application's task flow. When creating a task flow, consider how, and when, different users may interact. Does viewing data (such as a push notification) suffice to complete a task? If not, how much data entry does the task require? To frame these tasks within a mobile context, compare completing tasks using a desktop application to a mobile application. A single desktop application may enable multiple functions that might be partitioned into several different mobile applications (or in the context of MAF, several different application features embedded in a mobile application). Because mobile applications are generally used in short bursts (about two minutes at a time), they must be easily navigable and accommodate the limited data entry of a mobile device.
- During the design and development phases, keep in mind that mobile applications may require a set of mobile-specific server side resources, because they may not be able to consume the amount of data

delivered through complex web services. In addition, a mobile application may require extensive client side logic to process the copious amounts of data returned by the service.

- **Developing:** Select the technology that is best suited for application. While the MAF web view supports remote content which may be authored using Apache Trinidad (ADF Mobile browser) or ADF Faces Rich Client components, these applications do not support offline use. Applications authored in MAF AMX, which runs on the client, however, integrate with device services, enabling end users to not only view files and utilize GPS services, but also collaborate with one another by tapping a phone number to call or text. The MAF AMX component set includes data visualization tools (DVTs) that enable you to add analytics that render appropriately on mobile screens. A MAF AMX application supports offline use by transferring data from remote source and storing it locally, enabling end users to view information when they are not connected.
- MAF provides a set of wizards and editors that build not only the basic application itself, but also the application features that are implemented from MAF AMX and local HTML content. Using these tools provides such artifacts as descriptor files for configuring the mobile application and incorporating its application features, a set of default images for splash screens, springboards, navigation bar items that are appropriate to the form-factors of the supported platforms.
- **Deploying:** You deploy the MAF application not only in the context of publishing it to end users, but also for testing and debugging, because MAF applications cannot run until they have been deployed to a device or simulator. Depending on the phase of development, you designate the credential signing options (debug or release). For testing, you deploy the application to a mobile device or simulator. For production, you package it for distribution to application markets such as the Apple App Store or Google Play.
- To deploy an application you first create a deployment configuration that describes the target platform and its devices and simulators. Creating a deployment configuration includes selecting the splash screen and launch icons used for the application in different orientations (landscape or portrait) and on different devices (phone or tablets). For more information.
- **Testing and debugging:** During the testing and debugging stage, you optimize the application by deploying it in debug mode to various simulators and devices and then review the debugging output provided through Oracle Enterprise Pack for Eclipse (OEPE) and platform-specific tools. For more information.

- **Securing:** Evaluate security risks throughout the application development process. While mobile applications have unique security concerns, they share the same vulnerabilities as any application that accesses remotely served data. To ensure client-side security, MAF provides such features as:
- Whitelists that prevent such injection attacks as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).
- APIs that generate a strong password to secure access to the SQLite database and encrypt and decrypt its data.
- A set of web service policies that support SSL.
- A cacerts file of trusted Certificate Authorities to enforce deployment in SSL
- MAF's security configuration includes selecting a login server, such as the Oracle Access Mobile and Social server, or any web page protected by the basic HTTP authentication mechanism, configuring the session management (session and idle timeouts) and also setting the endpoint to the access control service web service, which hosts the application's user roles.
- Enabling access to the server-side data: After ensuring that your application functions as expected at a basic level, you can implement the Java code to access the server-side data.
- Redeploying: During subsequent rounds of deployment, ensure that after adding security to your application and enabling access to the server-side data, the application deployment runs as expected and the application is ready for the final testing and debugging.
- Retesting and debugging: During the final round of testing and debugging, focus on the security and the server-side data access functionality, ensuring that their integration into the application did not result in errors and unexpected behavior.
- Publishing: Deploying the application to the production environment typically involves publishing to an enterprise server, the Apple App Store, or Google Play. After you publish the MAF application, end users can download it to their mobile devices and access it by touching the designated icon. The application features bear the designated display icons and display as appropriate to the end user and the user's device.

2.8 Mobile App Frameworks

Oracle Mobile Application Framework is a hybrid mobile architecture, one that uses HTML5 and CSS to render the user interface in the web view, Java for the application business logic, and Apache Cordova to access device features such as GPS activities and e-mail. Because MAF uses these cross-platform technologies, you can build the same application for both Android and iOS devices without having to use any platform-specific tools. After they are deployed to a device, MAF applications behave as applications created using such platform-specific tools as Objective C or the Android SDK. Further, MAF enables you to build the same application for smartphones or for tablets, thereby allowing you to reuse the business logic in the same application and target various types of devices, screen sizes, and capabilities. The content of a MAF application is comprised of one or more embedded applications known as application features, which are represented as icons within the application's springboard or navigation bar.

Application features are essentially the building blocks of a mobile application. Each application feature that is integrated into a MAF application performs a specific set of tasks, and application features can be grouped together to complement each other's functionality. For example, you can pair an application feature that provides customer contacts together with one for product inventory. Because each application feature has its own class loader and web view, features are independent of one another; a single MAF application can be assembled from application features created by several different development teams. Application features can also be reused in other MAF applications. The MAF application itself can be reused as the base for another application, allowing ISVs (independent software vendors) to create applications that can be configured by specific customers.

In addition to hybrid mobile applications that run locally on the device, you can implement application features as any of the following mobile application types, depending on the requirements of a mobile application and available resources:

- Mobile web applications—These applications are hosted on a server. Although the code can be portable between platforms, their access to device features and local storage can be limited, as these applications are governed by the device's browser.
- Native applications—These applications are authored in either Xcode or through the Android SDK and are therefore limited in terms of serving both platforms. Reuse of code is likewise limited.

Below are some of the mobile application framework;

1. jQuery Mobile

jQueryMobile is a robust mobile development framework to build **cross-mobile-platform app**. jQuery Mobile support a wide range of different platforms, from a regular desktop, smart phone, tablet, or an e-reader device like Nook or Kindle. Similar to its sibling, jQuery UI, jQuery Mobile comprises a number of UI that, in this case, **is optimized for mobile and touch-enabled device**s.
Sources from http://www.jquerymobile.com

2. Cordova / PhoneGap

PhoneGap is essentially based on Cordova. Cordova/Phonegap provides a set JavaScript APIs that **connect to the device's native functions** such as Camera, Compass, Contacts, and Geolocation. Cordova/Phonegap lets us **build a mobile application without the native programming language**; instead we can use a framework like jQuery Mobile. It will compile your app using the platform's SDK and will be ready to install on the platform it supports including iOS, Android, Windows Phone, Blackberry and Web OS.
Sources from http://www.cordova.apache.org

3. Sencha Touch

Sencha Touch is a mobile framework **powered by HTML5 and CSS3**, providing APIs, animations, and components that are compatible with the current mobile platforms and browsers. Sencha Touch supports both **Cordova** and **PhoneGap**; you can compile your app, and submit your app to the respective platform's App Stores. In addition, Sencha Touch provides a set of themes for iOS, Android, Blackberry, Windows Phone, Tizen, and a variety of other platforms to help your app feel like a native app.
Sources from http://sencha.com/products/touch/

4. Ratchet

Ratchet was originally used by Twitter as an internal tool to create their mobile app prototype which is then released publicly as an open source project. Ratchet comes with a **collection of User Interface and JavaScript plugins** for building simple mobile apps, providing **reusable HTML classes**. In version 2.0,

Ratchet is also shipped with its proprietary font icon set named **Ratcheticon** and two **pre-made UI themes for iOS and Android**.
Sources from http://www.goratchet.com

5. Ionic

If you are concerned with your app performance, Ionic is the right framework for you. Ionic is an HTML5 mobile framework with focus on performance, by leveraging **hardware acceleration**, and it requires no third-party JS library. It works best **together with Angular.js to build an interactive app**. Similar to Ratchet, Ionic is shipped with a nicely crafted font icon set, Ionicons, and a bunch of reusable HTML classes to build the mobile UI. .
Sources from http://www.ionicframework.com

6. **Lungo**

Lungo is a **lightweight mobile framework based on HTML5 and CSS3**. It has very nice default styles that you can use as a starting point to design your mobile app. Aside for the mobile UI components, Lungo brings a number of JavaScript API to control your app. Lungo supports the following platforms: iOS, Android, Blackberry and Firefox OS. . Sources from http://www.lungo.tapquo.com

7. jQT

jQT is a Zepto plugin for mobile framework primarily **designed for Webkit browsers**. jQT is easily **customizable** and **extensible**. It comes with a theme that can be modified using Sass/Compass, cool 3D transition that is adjustable via CSS3, plus developers could also extend jQT with their own required functionalities. . Sources from http://www.jqtjs.com

8. Junior

Junior is also a **Zepto plugin** for building a mobile app **similar to jQT**. But Junior is dependent on several external libraries for some features to work, namely Backbone.js, Flickable.js for creating a swipe-able slider, and Ratchet for the UI scaffold. Sources from http://www.justspamjustin.github.io/junior/#home

9. Jo

Jo supports a wide variety of mobile platforms including **Tizen and Chrome OS.** Jo also comes with a starter, that is powered by CSS3, so it makes it easy for web developers at any level of experience to grasp and start styling their apps. In addition, you can use Jo along with **PhoneGap** or **Cordova** to pack your app for use on mobile platform.
Sources from http://www.joapp.com

10. Famo.us

A new kid on the mobile framework block, Famo.us promises to **eliminate HTML5 performance issue on mobile devices** with its **lightweight JavaScript engine** (only 64k). Famo.us, reportedly, will a also launch a cloud-based service to package your app to publish to the AppStore – it sounds like **Phonegap** and **Sencha** will get a new competitor soon. You can sign up as a beta tester in www.famo.us to get your hands on it. Sources from http://www.famo.us.com

III. Discussion

According to Matt Weisfeld a developer and also a college professor, he has often get asked the following question: "If I want to develop mobile apps, how important is it that I learn object-oriented programming?" His initial response to this query is very straightforward thus: If you want to program just about anything these days, you'd better learn object-oriented programming. He has been writing about object-oriented programming (OOP) since the early '90s, and the topic of how important it is has been raised in one form or another, well, since the early '90s. While the release of the first version of Java in January 1996 significantly increased the interest in OOP, the buzz, in fact, started even earlier than that, with Smalltalk and C++. Regardless of the context, the answer to the question posed above is always the same: Learning object-oriented programming is crucial to all modern software development—including mobile apps.

It can safely be said that the object has been the driving force in the programming industry for a very long time and will continue to be so for the foreseeable future. The evidence to support this statement is pretty compelling! Today, just about every major software development methodology is based on objects. As a result, virtually all programming languages, scripting languages and application designs are object-oriented or object-based.

To back these statements up, all we have to do is take a quick look at the importance that objects play in mobile app development. This includes native apps for phones as well as hybrid apps that can be viewed on any browser (whether it is on a phone, PC, tablet, etc.). While mobile apps are the hot topic today in the software industry, they are just the latest technology in its evolution. To demonstrate how important object-oriented programming is to mobile app development, let's explore the following questions:

- What mobile devices are the most prominent in the marketplace today?
- What programming languages are used to program these devices?
- Why has object-oriented programming become so important?

What Mobile Devices Are the Most Prominent in the Marketplace Today?

The category of mobile devices is quite broad and can include many products, such as smartphones, tablets, music players, computers, and so on. Since many of the development platforms across manufactures are similar, for example, the development environment for an Android smartphone is similar to that of an Android tablet, we can focus in on the smartphone category to provide a good representation of mobile devices in general. So let's take a look at the distribution of the smartphone market. One article from CNET listed the smartphone market share as follows: Apple's iOS at 51.2%, Google's Android at 44.2%, Microsoft's Windows Phone at 2.6% and Other as 2%. A bit more searching revealed that RIM's Blackberry held about 1% of the market, thus adjusting the "Other" category to 1%. With this information defined, we can now explore what programming languages are used to create phone apps for each of the platforms.

What Programming Languages Are Used to Program These Devices?

This information can be displayed in short order, and does not need a lot of explanation. The list can be seen below.

- iOS

Language: Objective-C

- Android

Language: Java

- Windows Phone

Language: .NET (C#, Visual Basic)

- BlackBerry

Language: Java

Source: http://en.wikipedia.org/wiki/Mobile_application_development

Based on this information, we can safely make the following statements with regard to the primary programming languages used for smartphone app development: iPhone apps are developed using Objective-C, Android apps are developed using Java, Windows Phone apps are developed using C# .NET, and BlackBerry apps are developed using Java. The common thread? All of the languages listed are based on objects! So there really isn't much debate as to how important objects are to the mobile app industry. It is not often that you can say that a single technology dominates virtually all of a specific market. However, in this case we can easily demonstrate that object-oriented programming is used in virtually all mobile app development.

Why Has Object-oriented Programming Become So Important?

It's all about the network! The rise of object-oriented programming was fueled by the emergence of the Internet as a place of business. While various flavors of object-oriented languages have been around since the 1960s, it is no accident that its widespread adoption coincided with that of the Internet. If we consider the fact that Java, which was one of the first popular object-oriented languages, was based heavily on C++, the obvious question is why wasn't C++ used instead of Java? The answer is that Java was designed with networks in mind. Originally, in the early 1990s, Java's direct ancestor, Oak, was created as part of a smart appliances project.

The Oak development team initially attempted to use C++, but there were many features of C++ that presented limitations, including difficulty in networking. The team also wanted a language that was highly portable and included a graphical user interface. After considering extending C++, they decided to create a new language, which was named Oak. Regardless of how successful the Oak programming language might have been, the project was a bit (only a bit) ahead of its time. While the state-of-the-art in the early 1990s might not yet have

been smart appliances and cable TV, something very state-of-the-art presented itself in the mid 90s: the Internet.

In 1994, Sun decided that the Internet was evolving into something that looked rather similar to the Oak team's vision of what the cable TV industry might someday look like. Oak was renamed Java, and the Java Development Kit (JDK) 1.0 was released on January 23, 1996. The rest, as they say, is history. Because Java was originally created with many features designed for use on a network, it was a perfect match for the Internet, which was poised to experience explosive growth. Java was strategically positioned to share in this growth, and the stage was set for many other languages to emulate the features that made Java successful. Oh, and there was one other pretty important (and interesting) thing about Java — it was fully object-oriented. The fact that Java was able to ride the wave of the Internet was partially due to the fact that HTML only presented static content to web pages. One of the cool new features that Java provided was a feature called applets. Applets provided dynamic functionality such as animation, capturing input, and a variety of controls (buttons, check boxes, etc). Java also provided a library for the creation of user interfaces (AWT) that would render on multiple platforms, which was, at the time, virtually unheard of.

Interestingly enough, the importance of Applets eventually faded as newer technologies within Java emerged (and AWT was superseded by Swing). Thus, some of the features of Java that led to its initial acceptance were somewhat quickly abandoned. However, the object-oriented nature of Java, as well as its networking capabilities, laid the foundation for the future of programming languages that fueled the Internet and, eventually, mobile devices. Today, virtually all of the programming , scripting languages and frameworks used to create mobile applications rely on object technology: Java, C# .NET, Visual Basic .NET, Objective-C, Ruby, JavaScript, jQuery, jQuery Mobile, etc.

Finally, perhaps the most interesting back-story in this discussion is the fact that, as often happens, the technology has come full circle. Remember that the emergence of object technology began in the early 90s with Oak and its vision of smart appliances. Also remember that this vision was well ahead of its time. So consider this: years ago the Internet was the hot development topic, and today mobile apps are the hot development topic. Perhaps tomorrow's hot topic may well be the concept of "The Internet of Things," which is being promoted by companies such as GE and Microsoft. "The Internet of Things" refers to uniquely identifiable objects. Thus, the next generation of object development may be the realm of smart cars and smart homes, which contain smart appliances. Sound familiar?

IV. Conclusion

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data. The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data. Android is designed to run on many different types of devices, from phones to tablets and televisions. As a developer, the range of devices provides a huge potential audience for your app. In order for your app to be successful on all these devices, it should tolerate some feature variability and provide a flexible user interface that adapts to different screen configurations. To facilitate your effort toward that goal, Android provides a dynamic app framework in which you can provide configuration-specific app resources in static files (such as different XML layouts for different screen sizes). Android then loads the appropriate resources based on the current device configuration. So with some forethought to your app design and some additional app resources, you can publish a single application package (APK) that provides an optimized user experience on a variety of devices. If necessary, however, you can specify your app's feature requirements and control which types of devices can install your app from Google Play Store. This page explains how you can control which devices have access to your apps, and how to prepare your apps to make sure they reach the right audience. For more information about how you can make your app adapt to different devices, read Supporting Different Devices.

References

[1]. Alexander Sergeev in http://32dayz.com/blog/961/
[2]. "App" voted 2010 word of the year by the American Dialect Society (UPDATED) American Dialect Society".
[3]. Ballard, P. Sams Teach Yourself JavaScript in 24 Hours, Fifth Edition. Hour 7. What is Object Oriented Programming (OOP)?. Pearson Education, Inc. United States. November, 2012.
[4]. Bhave, M. Object Oriented Programming with C++, Second Edition. Chapter 4. Object Orientation: An Introduction. Section 4.1. Programming Paradigms. Pearson Education. India. May, 2012.
[5]. Chatzigeorgiou and G. Stephanides. Evaluating Performance and Power of Object-Oriented Vs. Procedural Programming in Embedded Processors. In Ada-Europe 2002, 2002.
[6]. Carnegie Mellon University (2008). What is Alice? Retrieved from http://www.alice.org/index.php?page = what_is_alice/what_is_alice.
[7]. Energy: An Overview. In 1994 Symposium on Low-Power Electronics, San Diego, CA, October 1994.
[8]. Frontiers in Education Conference (FIE), 2014 IEEE

[9]. Glinert, E. P. (1986). Towards "second generation" interactive, graphical programming environments. Proceedings of 2nd IEEE Computer Society Workshop on Visual Languages, Dallas, Texas, USA.
[10]. K. Naik and D.S.L. Wei. Software Implementation Strategies for Power-Conscious Systems. In Mobile Networks and Applications, June 2001.
[11]. Lifelong Kindergarten Group (2006). About Scratch. Retrieved from http://info.scratch.mit.edu/About_Scratch.
[12]. Matt Weisfeld derived from http://www.informit.com/articles/article.aspx?p=2036576
[13]. Peymandoust, T. Simunic and G.D. Micheli. Low Power Embedded Software Optimization using Symbolic Algebra. In IEEE Proceeding of the 2002 Design, Automation and Test in Europe Conference and Exhibition, 2002.
[14]. Siegler, MG (June 11, 2008). "Analyst: There's a great future in iPhone apps".
[15]. Vasappanavara, R. Object-oriented Programming Using C++ and Java.Chapter 1. Object-oriented Programming Basics. Section 1.3. Programming Paradigms. Pearson Education. India. May, 2011.
[16]. Turbak, F., Sandu, S., Kotsopoulos, O., Erdman, E., Davis, E., & Chadha, K. (2012). Blocks languages for creating tangible artifacts. The Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2012), Innsbruck, Austria. Retrieved fromhttp://cs.wellesley.edu/~fturbak/pubs/VLHCC-2012-paper-turbak.pdf.
[17]. Margulieux, L. E., Guzdial, M., & Catrambone, R. (2012). Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. Proceedings of the ninth annual international conference on International computing education research - ICER '12, 71. doi:10.1145/2361276.2361291
[18]. Hsu, Y. -C., Rice, K., & Dawley, L. (2012). Empowering educators with Google's Android App Inventor: An online workshop in mobile app design.British Journal of Educational Technology, 43(1), E1-E5. doi:10.1111/j.1467-8535.2011.01241.x
[19]. http://bgr.com/2012/08/16/blackberry-market-share-us-2012-usage/
[20]. http:// cordova.apache.org
[21]. https://developer.android.com/sdk/installing/studio.html
[22]. https://developer.android.com/guide/index.html
[23]. http://docs.oracle.com/middleware/mobile200/mobile/develop-oepe/oepe-maf-about.htm
[24]. http://en.wikipedia.org/wiki/C_Sharp_(programming_language)
[25]. http://en.wikipedia.org/wiki/Java_(software_platform)#History
[26]. http://en.wikipedia.org/wiki/Java_version_history
[27]. http://en.wikipedia.org/wiki/Mobile_application_development
[28]. http://en.wikipedia.org/wiki/Objective-C
[29]. http://en.wikipedia.org/wiki/Visual_Basic_.NET
[30]. http:// famo.us.com
[31]. http:// goratchet.com
[32]. http:// hongkiat.com/blog/mobile-frameworks/
[33]. http:// ionicframework.com
[34]. http:// joapp.com
[35]. http:// jqtjs.com
[36]. http:// jquerymobile.com
[37]. http:// justspamjustin.github.io/junior/#home
[38]. http:// lungo.tapquo.com
[39]. http:// sencha.com/products/touch/
[40]. https://software.intel.com/en-us/blogs/2013/08/13/the-hierarchy-of-developer-motivation (Wendy Boswell (Intel) on August 13, 2013)
[41]. http://news.cnet.com/8301-13579_3-57565106-37/iphone-wins-51-percent-of-u.s-smartphone-sales-says-report/
[42]. http://vogella.com/tutorials/Android/article.html
[43]. http://w3schools.com/xml/xml_whatis.asp