A MOBILE APPLICATION FOR POWER SYSTEM ANALYSIS

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Sarang Satish Deshpande

FALL
2016
A MOBILE APPLICATION FOR POWER SYSTEM ANALYSIS

A Project

by

Sarang Satish Deshpande

Approved by:

______________________________, Committee Chair
Dr. Pinar Muyan-Ozcelik

______________________________, Second Reader
Dr. Mahyar Zarghami

______________________________
Date
Student: -Sarang Satish Deshpande

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________
Graduate Coordinator

Dr. Ying Jin

Department of Computer Science
Abstract

of

A MOBILE APPLICATION FOR POWER SYSTEM ANALYSIS

by

Sarang Satish Deshpande

The purpose of this project is to design an educational software that teaches the design and analysis of electric power systems using mobile platforms. To the best of our knowledge, no mobile platform exists for this purpose yet. This project examines the desktop-based application used by CSUS-EEE students (called RadiRing) for electric power systems and makes a subset of the functionalities of this application available on mobile devices.

Some of the existing functionalities of RadiRing [14] app include drawing components of an electric power system such as bus, branch and shunt elements and connecting these components to each other. The app also allows user to move the connected components on screen.

To implement similar application using mobile technology, a Java-based cross-platform mobile application environment called Codename One (CN1) is used. The advantage of CN1 is that the application written in this framework can run on various
mobile devices with different operation systems (i.e., Android, iOS, Windows). Hence using CN1 shortens the development time of the project and increases the usability of the educational app in this project.

The project uses several different implementation techniques including Model View Controller architecture, Observer design pattern, and event-driven and interactive techniques such as pointer handling and component repainting.

_______________________, Committee Chair
Dr. Pinar Muyan-Ozcelik

_______________________
Date
ACKNOWLEDGEMENTS

I would like to express gratitude to my advisor Dr. Pinar Muyan-Ozcelik for the opportunity to work on this project, providing guidance and constant supervision. This project significantly broadened my knowledge on Java and CodeName One Development. I, again, thank her for continuously providing the feedback, help and support to complete the project successfully.

In addition, I would like to thank Dr. Mahyar Zarghami for his willingness to serve on the committee and give me the opportunity to work on this project.

I would like to thank my family and friends, for their co-operation and encouragement.

I would also like to thank the entire faculty and staff of the Department of Computer Science Engineering at California State University, Sacramento.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. RADIRING DESKTOP APPLICATION</td>
<td>3</td>
</tr>
<tr>
<td>3. CODENAME ONE</td>
<td>5</td>
</tr>
<tr>
<td>3.1 Codename One Development Basics</td>
<td>5</td>
</tr>
<tr>
<td>3.1.1 Why Build Servers?</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 Lightweight UI</td>
<td>7</td>
</tr>
<tr>
<td>3.1.3 Installation of Codename One in Eclipse</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Access Native Device Functionality</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Creating Project with Codename One</td>
<td>10</td>
</tr>
<tr>
<td>4. TECHNIQUES</td>
<td>14</td>
</tr>
<tr>
<td>4.1 Implementing Project with Codename One</td>
<td>14</td>
</tr>
<tr>
<td>4.2 MVC Architecture</td>
<td>15</td>
</tr>
<tr>
<td>4.3 Observer Pattern</td>
<td>17</td>
</tr>
<tr>
<td>4.4 Component Repainting</td>
<td>20</td>
</tr>
<tr>
<td>4.5 Pointer Handling</td>
<td>21</td>
</tr>
<tr>
<td>5. ORGANIZATION – UML DIAGRAM</td>
<td>24</td>
</tr>
<tr>
<td>6. EXECUTION RESULTS</td>
<td>26</td>
</tr>
</tbody>
</table>
6.1 Screen Shots of the app on iOS Skin ..........................................................26
6.2 Screen Shots of the app on Android Skin .................................................29
7. CONCLUSION ..................................................................................................32
8. FUTURE WORK .............................................................................................33
References ........................................................................................................34
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. RadiRing Application</td>
<td>3</td>
</tr>
<tr>
<td>2. Main Components of Codename One framework</td>
<td>6</td>
</tr>
<tr>
<td>3. Install Plugin</td>
<td>8</td>
</tr>
<tr>
<td>4. Add Repository</td>
<td>8</td>
</tr>
<tr>
<td>5. Setup Wizard Codename One</td>
<td>9</td>
</tr>
<tr>
<td>6. New Project</td>
<td>10</td>
</tr>
<tr>
<td>7. New Project Wizard</td>
<td>11</td>
</tr>
<tr>
<td>8. Select Project Name</td>
<td>11</td>
</tr>
<tr>
<td>9. Setting a Package Name</td>
<td>12</td>
</tr>
<tr>
<td>10. Default Project Structure</td>
<td>12</td>
</tr>
<tr>
<td>11. Default HelloWorld Codename One App</td>
<td>13</td>
</tr>
<tr>
<td>12. MVC Architecture</td>
<td>15</td>
</tr>
<tr>
<td>13. General Structure of MVC</td>
<td>17</td>
</tr>
<tr>
<td>14. Code Snippet of HPSPodel that extends Observable</td>
<td>18</td>
</tr>
<tr>
<td>15. Code Snippet of HPSView that implements Observer</td>
<td>19</td>
</tr>
<tr>
<td>16. Code Snippet of HPSController that Registers the Observer</td>
<td>19</td>
</tr>
<tr>
<td>17. Code Snippet of paint() Method</td>
<td>21</td>
</tr>
<tr>
<td>19. Code Snippet that Shows Implementation of pointer dragged Event</td>
<td>23</td>
</tr>
<tr>
<td>20. UML Diagram</td>
<td>25</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>21. Initial View On iOS</td>
<td>26</td>
</tr>
<tr>
<td>22. Placing Components on Screen (iOS)</td>
<td>27</td>
</tr>
<tr>
<td>23. Dragging these Components to Different Location (iOS)</td>
<td>28</td>
</tr>
<tr>
<td>24. Initial View on Android</td>
<td>29</td>
</tr>
<tr>
<td>25. Placing Components on Screen (Android)</td>
<td>30</td>
</tr>
<tr>
<td>26. Dragging these Components to Different Location (Android)</td>
<td>31</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

A Power System Analysis application or similar tool is not available on Play Store [1], App Store [2] or Windows Store [3] based on search options used on respective stores. There are only some basic electric circuit design applications, which can be useful to power related courses.

RadiRing is a desktop-based application used for power system analysis at Sacramento State. This app has been used for educational purposes in courses such as EEE 142 (Energy Systems Control and Optimization). More information about this app is provided in Section 2. In the current project, I have designed a mobile app to implement a subset of features available in RadiRing. The project is a cross-platform mobile application, which allows students to design and to analyze an electric power system. Mobile apps have advantage over desktop applications, considering widespread use of mobile devices. Specially, the cross—platform mobile apps are advantages due to the fact they can run on all major platforms such as iOS, Android, Windows. Hence, the app I develop in this project will improve students learning of power system analysis by allowing students to perform practical problems in class settings (using their mobile devices which may run various operating systems) while the professor gives a lecture.

The mobile app is designed using Java and cross-platform mobile application framework – CodeName One. The main functionalities of Codename One is explained in
Section 3. The main advantage of using this framework is its “Write Once Run Anywhere” characteristics. This not only saves the time but also it keeps the structure of application the same across all platforms.

This app uses object-oriented programming techniques to draw components such as branch, node and shunt in electric power systems, to connect them to each other, and to move the connected components as explained in Section 4. The adaptive design of Codename One allows the app to fit on various mobile platforms with any screen size. The app also uses MVC architecture to separate Model, View and Controller parts of the implementation. Section 5 provides UML that show the complete organization of the app. The project also uses techniques such as Observer design pattern, event-driven programming, pointer handling, and component repainting.
Chapter 2

RADIRING DESKTOP APPLICATION

The desktop-based application named ‘RadiRing’, has been developed and used at Sacramento State for designing a power circuit for power related courses. The basic idea in RadiRing is to form a circuit of components by selecting and placing them on the screen. Later, we can assign values to those components and perform power system analyses such as Load-Flow and Short-circuit. Below is the screenshot of desktop-based RadiRing [14] with some drawn components. The app I develop in this project allows drawing the circled components in Figure 1.

Figure 1 RadiRing Application
In the screenshot, there are various components including Bus, Load, Shunt Admittance, Connecting lines, Generator, and Capacitor. Among these, Bus, Load, Shunt Admittance, Generator and Capacitor can be drawn vertically or horizontally. A horizontal or vertical Bus can be drawn independently. In addition, based on additional rule, a horizontal shunt component can only be attached to a vertical bus and a vertical shunt component can only be attached to a horizontal bus.

In addition to drawing the abovementioned components, desktop-based RadiRing also allows user to save and later load the drawn components and provides zooming-in and zooming-out capabilities.
3.1 Codename One Development Basics

The main aim of Codename One project is to use a single set of tools, APIs, and services in order to develop a mobile application, which will work as a native application on all platforms.

Codename One [4] is a Java-based mobile application development environment which blends various technologies and concepts into a single tool as can be seen in Figure 2:

- **API** – Codename One API (Application Programming Interface) defines set of routines, protocols, and tools for building cross-platform Codename One applications that run across various mobile platforms.
- **Plugin** – Installed Codename One plugin supports the following features:
  - IDE Integration: It sets various preferences and has the ability to create a native build.
  - Simulator: Plugin enables the application to run on the simulator with any mobile platform and allows us to debug application.
  - GUI Builder: It supports designing components using ‘drag and drop’ technique instead of writing a code for it.
- **Build Servers** – Build servers convert the JARs sent by Codename One plugin, into native applications.
• Cloud Servers – Cloud servers support Push notifications, storage, cloud logging, and many more features.

Figure 2 Main Components of Codename One Framework

In my project, I mainly take advantage of API, IDE integration, and simulator components of Codename One. Although currently I do not utilize GUI builder and build/cloud servers, as explained in Section 8, I plan to also use them in the future work.
3.1.1 Why Build Servers?

Build servers are designed to build native iOS app without using a Mac or to build native Windows app without using a Windows machine. Codename One has simplified the process of building the native app with quick and easy steps. It maintains cloud servers that consist of Mac OS X with Xcode and Codename One, which will generate complete native iOS app.

This applies to other platforms as well.

3.1.2 Lightweight UI

The lightweight architecture of Codename One is its biggest advantage. A lightweight component is entirely written in Java, which has its own interfaces and handles its own events. So it has a huge advantage of portability. Using inheritance, overridden paint methods, and event handling, these components can be customized [4]. The theming and GUI builder concept enables live preview along with accurate reproduction on all platforms, as the same code executes everywhere.

3.1.3 Installation of Codename One in Eclipse

The following shows a step-by-step guide for installation of Codename One in Eclipse (i.e. the IDE I have utilized in my project):

Start the Eclipse, click Help and then Install New Software. A pop-up shown in Figure 3 appear.
Now, click the add button, and another pop-up of ‘Add repository’ will appear as shown in Figure 4. Fill out the details Name and Location with details given in the figure.

Next step is to mark the entries box and follow the setup wizard as shown in Figure 5.
3.2 Access Native Device Functionality

Native interfaces in Codename One enable the developer to use native methods, libraries, functions and widgets directly for every platform. It works as a bridge between Codename One and OS specific features [5].

The developer creates a method that extends NativeInterface, which will generate methods related to native platforms. In this method, the developer have to follow specific types of arguments/return types like – primitive types, arrays of primitive types, string and peer component.
After the developer code for the native interface, he can generate native code by right clicking on it. This will create native code into respective directories such as Objective-C code in iOS, Java Dalvik in Android and C# in Windows Phone directory.

The developer can also add JARs and iOS libraries into native directories of android and iOS, without any changes. The current mobile app doesn’t need this feature.

3.3 Creating Project with Codename One

We create a Codename One project inside eclipse using the following steps [4]:

First, we create a new project as shown in Figure 6.

![Figure 6 New Project]
Then, we choose Codename One project from new project wizard as shown in Figure 7.

![Figure 7 New Project Wizard](image)

Now, we create a project with a specific name as shown in Figure 8.

![Figure 8 Select Project Name](image)
Next step is to set a package name as shown in Figure 9.

![Figure 9 Setting a Package Name](image)

Using above steps, the project will be created and its structure will look as shown in Figure 10:

![Figure 10 Default Project Structure](image)
After executing this default Codename One app, it will look as shown in Figure 11.

Figure 11 Default HelloWorld Codename One app
Chapter 4

TECHNIQUES

4.1 Implementing Project with Codename One

RadiRing is a desktop-based application. To design a similar mobile app which is, more accessible, portable and cross-platform, Codename One is a proper technology to use. Doing so, we can test the designed app on simulators with different platforms without installing any additional software. As mentioned in Section 3, Codename One is Java-based framework. Java allows us to develop Object-Oriented applications. Object-Oriented programming [6] allows us to add following properties to the application:

1. Encapsulation and Abstraction – Encapsulation binds the data together and keeps the code safe from external interfaces [15]. Abstraction provides the functionality to the user by hiding the implementation details from user [16]. In the current project, as shown in Section 5, I divide the implementation to several different classes, which provides encapsulation and abstraction.

2. Inheritance – Inheritance is the property where a class can inherit properties of other class [17]. For instance, the current project has several tool classes. Hence, as shown in the Section 5, I have created an abstract parent class named Tools which has property to store location of a tool. This class is extended by concrete tools which inherit the properties of Tools.
3. Polymorphism – Polymorphism allows us to use reference of parent class type, instead of using sub class type [18]. For instance, in this project, I traverse through a collection type of Tools and call a draw() method in each element. draw() method is an abstract method in Tools which is implemented differently in concrete subclasses of Tools. Using polymorphism, I call a draw() method in each element of the collection without downcasting from Tools type to a sub-class type.

4.2 MVC Architecture

Model, View and Controller (MVC) is a software architecture used for implementing user interfaces. It separates the software application from its internal representation and the information that to be presented to the user [7]. Below, I provide the brief description of three main parts of MVC architecture [8].

![MVC Architecture](image)

Figure 12 MVC Architecture
**Model** – It represents the object, which carries and manages the data. Model fetches the data according to commands reside in the controller. It also instructs the view to change its presentation when this data changes.

**View** – It represents the visualization of data present inside the model. It generates a new view to the user whenever data in model is updated.

**Controller** – It controls execution flow of the program and send instructions to model to update the model, itself on change of data. It also takes care making a connection between model and view.

I have followed the MVC architecture to design my project as shown in Figure 12. I have used the term HPS (Hybrid Power System) in my design. As can be seen in Section 5, HPSModel, HPSView, and HPSController classes allow me to follow this architecture. The model includes a collection that includes various tools. The controller creates GUI components, which generates the screen layout, creates the model and view and registers the view to the model as its observer. Finally, the view (which extends from built-in Container class of Codename One) has the overridden paint() method which allows graphical display of all the tools that reside in the collection that is saved in the model.
4.3 Observer Pattern

Design patterns help in the development of the project by tested and proven methodologies [9]. I have used observer design pattern to implement the relationship between the model and view parts of the MVC architecture I have used in my project.

The main intent of observer pattern is “Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically [10].” Hence, observer pattern plays an important role in the implementation of MVC architecture.

The view part of my project implements the built-in Observer interface of Codename One. The model class of my project extends the built-in Observable class of Codename One. Hence, whenever new tool is added into the model, the model automatically notifies the view to update itself. Specifically, addTool() method in HPSModel calls notifyObservers() (a method defined in Observable class)
which automatically calls `update()` (a method declared in `Observer` interface) in `HPSView` which in turn calls the `repaint()` (a method which is explained in the next section) on the view. This way the newly added tool is displayed on the screen immediately. Figure 14, 15 and 16 show the model, view and controller code snippets showing that project implemented using observer pattern.

```java
package com.mycompany.myapp;
import java.util.ArrayList;
import java.util.Observable;

public class HPSModel extends Observable{
    static ArrayList<Tools> lstTools = new ArrayList<Tools>();

    public void addTools (Tools t){
        lstTools.add(t);
        setChanged();
        notifyObservers();
    }
}
```

Figure 14 Code Snippet of HPSModel that extends Observable
Figure 15 Code Snippet of HPSView that implements Observer

```java
public class HPSView extends Container implements Observer{

    public void update(Observable observable, Object data) {
        this.repaint();
    }
}
```

Figure 16 Code Snippet of HPSCController that Registers the Observer

```java
public class HPSCController extends Form {

    public HPSCController(){
        HPSModel model = new HPSModel();
        HPSView view = new HPSView();
        model.addObserver(view);
    }
}
```
4.4 Component Repainting

Component class of Codename One has repaint() method defined, which calls the paint() method. The component uses this method where there is a change/update about its display. To actually draw the component, the graphics parameter, which passed from repaint() method to the paint() method, is used. Repaint() and paint() both have different functionalities. Thus, direct invoking of the paint() method is not advised [11].

In my project, as a result of observer design pattern, the view (which extends from built-in Container class of Codename One which in turn extends Component class) calls repaint(), whenever a user adds a new component, or whenever the user drags the component from one point to another. The paint() method that is called from repaint() then calls a draw on each element of the collection of tools as shown in Figure 17.
4.5 Pointer Handling

ActionEvent is generated whenever pointer is pressed, dragged or released on components. Component class of Codename One provides the several pointer-related methods which can be overridden by user to handle events generated by the pointer [12] such as: pointerPressed(), pointerDragged(), and pointerReleased().

In my project, I have handled pointer pressed and dragged events to add different functionalities to my app as explained below:

```java
public class HPSView extends Container implements Observer{

    public void update(Observable observable, Object data) {
        this.repaint();
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        for(Tools t: HPSModel.lstTools){
            t.draw(g);
        }
    }
}
```

Figure 17 Code Snippet of paint() Method
Handling of pointer press event – User can select a tool and draw it anywhere on workspace by pressing on the display. To allow this functionality, I have overridden `pointerPressed()` method to put the tool at that exact point the user has pressed.

```java
@override
public void pointerPressed(int x, int y){
    curX = x;
    curY = y;
    if(isFlagBH()){
        Point pt = new Point(curX, curY);
        Tools t = new BusHorizontal(pt);
        hpsModel.addTools(t);
        setFlagBH(false); setFlagBV(false);
    }
    if(isFlagLH()){
        Point pt = new Point(curX, curY);
        for(Tools t: HPSModel.lstTools){
            if(t instanceof BusVertical){
                if(Math.abs(t.getP().getX()-curX)<=8){
                    if((t.getP().getY() < curY) && (curY < (t.getP().getY()+100))){
                        pt = new Point(t.getP().getX(),curY);
                        Tools newLH = new LoadHorizontal(pt,t);
                        hpsModel.addTools (newLH);
                        t.addConnectComponent(newLH);
                        break;
                    }
                }
            }
        }
    }
}
```

Figure 18 Code Snippet that Shows Implementation of pointer press Event
Handling of the pointer dragged event – User can drag connected tools to move and update their locations. To allow this functionality, I have overridden the `pointerDragged()` to figure out which connected groups of tools should be dragged and where they should be moved.

```java
@Override
public void pointerDragged(int x, int y)
{
    if(isFlagBHDragg()){
        for(Tools t: HPSModel.lstTools){
            if(t instanceof BusHorizontal){
                if(x < t.getP().getX()+100 && x > t.getP().getX()-20){
                    setFlagBVDragg(false);
                    Point p1=new Point(x,y);
                    if( y < t.getP().getY()+20 ){
                        for(Tools t1: ((BusHorizontal) t).connectedTools){
                            Point pt1=t1.getP();
                            float diffX= pt1.getX()- t.getP().getX();
                            float diffY= pt1.getY()- t.getP().getY();
                            Point(newPoint=new Point(x+diffX,y+diffY));
                            t1.setPoint(newPoint);
                        }
                    }
                }
            }
        }
    }
}
```

Figure 19 Code Snippet that Shows Implementation of pointer dragged Event
Chapter 5

ORGANIZATION – UML DIAGRAM

UML diagrams are effective tools to show the organization of the system. UML makes the designing of project easy and leads to the clean implementation practice. Specially, UML allows us to do the following [13]:

- Defining various elements and their characteristics
- Defining various tasks and communications within them
- Defining overall organization of objects, methods, parameters, association between objects.

Figure 20 shows the UML diagram that displays the organization of my project which follows the techniques mentioned in Section 4.
Figure 20 UML Diagram
Chapter 6

EXECUTION RESULTS

In this section, I provide execution results of my app by listing several screen shots. I provide results of running the app both on iOS and Android platforms.

6.1 Screen Shots of the app on iOS Skin

Figure 21 shows the initial view of the app on iOS skin (i.e. iPad skin).

![Figure 21 Initial View on iOS](image-url)
Figure 22 shows the display after the user draws several tools to the screen. In this figure, we see that the user has created two groups of connected tools:

1) a vertical shunt and a vertical load are connected to a horizontal bus,

2) a horizontal shunt and a horizontal load are connected to a vertical bus.
User can drag the connected tools to different locations on the screen as shown in Figure 23.

Figure 23 Dragging these Components to Different Location (iOS)
6.2 Screen Shots of the app on Android Skin

In this section, I provide the execution results collected from running the system on an Android skin (i.e. Nexus 7 tablet). Figure 24, 25 and 26 show the initial view, drawn tools and dragged tools, respectively.

Figure 24 Initial View on Android
Figure 25 Placing Components on Screen (Android)
Figure 26 Dragging these Components to Different Location (Android)
Chapter 7

CONCLUSION

In this project, I have developed a cross-platform mobile app that provides the subset of features of a desktop-based RadiRing application that is used for power system analysis.

This project aims to help in improving the students’ learning of this topic by making this educational application more accessible (i.e., students can run this app on various mobile platforms that run different operating systems). Codename One mobile application framework was a great fit for developing this app due to following two reasons: 1) it allows developing a cross-platform project which increases the accessibility of the app and reduces the development time, 2) it is Java-based which allows us to take advantage of object-oriented techniques in our implementation. In addition to basic object oriented techniques (i.e., encapsulation, abstraction, inheritance, and polymorphism), to implement my app, I have also used several other techniques such as MVC architecture, observer design pattern, component repainting, and pointer handling.

My results indicate successful execution of the app on the simulations of different mobile platforms. As a future work, as explained in Section 8, the app can be extended to fully support all features of the desktop-based application and then can be made available on various mobile marketplaces (i.e., Google’s Play Store, Apple’s App Store) for all students that study power system analysis.
Chapter 8

FUTURE WORK

The mobile application I have developed in this project supports only a subset of features that are provided in desktop-based RadiRing. Thus, the app can be extended with functionalities including, but not limited to adding support for drawing the remaining tools of RadiRing, assigning values to tools, and saving/, loading the drawn tools and its associated values. The execution of the app has been tested on simulator skins that correspond to different platforms. As a future work, the code can be sent to the build servers of Codename One to generate native code and then, the app can be installed to the physical devices. After it is tested thoroughly on these devices, the app can be published on various mobile marketplaces.
REFERENCES


[Accessed: October 31st, 2016]

[Accessed: November 1st, 2016]


[11] Dr. Pinar Muyan-Ozcelik, CSc 133 – Lecture Notes, Chapter 10 – Interactive Techniques, California State University, Sacramento.

[12] Dr. Pinar Muyan-Ozcelik, CSc 133 – Lecture Notes, Chapter 9 – Event Driven Programming (Part 2), California State University, Sacramento.


[Accessed: May 10th, 2016]

[Online]. Available: http://beginnersbook.com/2013/03/oops-in-java-
encapsulation-inheritance-polymorphism-abstraction/ [Accessed: September 15, 2016]

