Object-Oriented Programming Languages: Tools for Effective Communication on Application's User Interface

[1]ONU, Fergus U., [2]BAAH, Barida, [3]AKIENE, Promise T. K.

Department of Computer Science, Ebonyi State University, Abakaliki – Nigeria

[1]*uche.fergus[at]gmail.com;* [2]*baridakara1[at]yahoo.com;* [3]teepromise[at]yahoo.com

Abstract: ***To an application user, the application is as good as the user interface. The beauty of a good application's user-Interface is dependents on the effectiveness of the communication between the hardware, software and the humanware (user). This paper examines the fundamental concepts of object-oriented programming languages as a tool for an effective communication in a user-interface. Information from secondary sources was gathered, studied and through an object-oriented methodology, a detailed analysis and design was adopted to create a simple arithmetic and trigonometric drill application with a good user interface. MATLAB Version 7.7.0 was used as the object oriented programming tool to implement the application. The created simple arithmetic and trigonometric drill application has advanced features to perform mathematical and trigonometric functions effortlessly.***

Keywords: Object-oriented languages, Objects, Software Tools, Effective communication, User-Interface

1. Introduction

Most computer applications fail as a result of lack of effective communication between the user and the application. Every interaction between an application and the user take place at the interface of the system. Most applications fall short of good interface during the development of such system as a result of the sets goals for which such system was developed is non-realistic. But with the application of object oriented programming, most of these problems is brought to almost zero level because of the use of object while communicating, which makes communication very effective due to it visual nature of the interface system during the design of any application for use.

"Object-oriented programming means only messaging, encapsulating and hiding state and extreme late-binding of all things [1]. It could be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I'm not aware of them" stated [1]. The big idea is "messaging" that is what the kernel of Smalltalk/Squeak is all. The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be. Object oriented programming language is a programming paradigm base on the concepts of object which may contain data in the forms of fields often refers to as attributes; and code in form of procedure which is often refers to as method [2].

A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated. According to [3] computer programs are designed by making them out of objects that interact with one another. Again, object oriented programming could also be defined as a user-friendly technique with robust mechanism to holistically tackle the plan in solving complex problems. In the words of [4], it is a programming method that combines data and instructions for processing the data into a self-sufficient object that can be used in other programs.

Most Object-Oriented Programming Languages (OOPLs) come in visual form. The great merit of using visual methods to introduce object oriented programming to novice/amateaur programmers is that (1) it helps them to create a graphical user interface and (2) it can aid them to avoid syntax errors commonly seen while working with textual programming languages. In addition, the drawer analogy used for arranging puzzle pieces in visual games with similar function can reduce the need for novices to remember exact textual codes [5]. This can greatly reduce the potential cognitive load caused by programming with textual codes [6].

Application Inventor (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. It has drag-and-drop feature of visual programming, which lets designers see how different pieces come together, and how their programming relates to the behaviors of their artifacts/products especially in mobile apps [7]. 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.

2. Literature

Object oriented Programming languages support the development of graphical user interfaces, by providing toolkits of useful interface controls like in Java called widgets. Java Swing provides high-level organizational components that specify the overall layout of a graphical interface. The interface is implemented as a class. This is implemented with an algorithm like this: Construct an object frame, tell frame to organize its GUI components into a 2-dimensional layout and tell the frame to make it visible.

The following were pointed out by [8] as the seven stages of a typical user interaction with a generic interaction system:

1. Forming the goal; I want to do something in a program.
2. Forming the intention: I have to start the program.
3. Specifying the action: I have to click on a button to open the program.
4. Executing the action: I click on the button.
5. Perceive the system state: Note that the computer operates.
6. Interpret the system state: The computer opens (hopefully) the right program and
7. Semantically evaluating the interaction outcome: Note that the right program is open.

The design of graphical user interfaces (GUI) can also be described as visual design [9]. This can be seen as an integration of graphical components on the application interface. Visual design disciplines are communication-oriented, graphic design, industrial design and architecture. As mentioned earlier, visual design disciplines are communication-oriented, therefore the interaction between the human and the machine is the most important aspect. Interaction takes place with input and output devices. Typical input devices are a computer mouse, physical buttons, a touch screen, or a microphone. Typical output devices are screens, printers, or speakers. The user interface is responsible for connecting the system user to the input and the output devices.

According to [10] the importance of objects 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. In the opinion of [11], the adoption of OOP techniques in web Application development has brought about the following:

(1) Reduced number of code a programmer needs to write,
(2) Enable reuse of design and code function
(3) Transfer design knowledge and experience to a developer,
(4) Improved Maintainability of web application,
(5) Reduced cost and time of developing a web application and
(6) Improved application access restriction from unauthorized person,
(7) Created opportunity for web application development
(8) created opportunity for integration of external web application with other web application such as Intranet and Extranet, by allowing object/module to be accessible to other functions or programs
(9) Proper computation and organization of complex tasks.

Eight parameters have been identified by [12] as the basis for the choice of a good programming languages. These are: (1) Flexibility, (2) Interface design. (3) Completeness, (4) Speed, (5) Accessibility, (6) Compatibility, (7) Language Safety and (8) Responsiveness of the Language.

Thinking or becoming Object-Oriented

An object is an instance of an abstract data type. An object data types is implemented via a class. In other word an object can also be define as the collection of operations that share a state. The operations determine the messages (calls) to which the object can respond, while the state is hidden from the outside world and it is accessible only to the objects [13]. Objects are key to understanding *object-oriented* technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle. Real-world objects share two characteristics: They all have *state* and *behaviour*. Dogs have state (name, color, breed, hungry) and behaviour (barking, fetching, wagging tail). Bicycles also have state (current gear, current pedal cadence, current speed) and behavior (changing gear, changing pedal cadence, applying brakes). Identifying the state and behaviour for real-world objects is a great way to begin thinking in terms of object-oriented programming.

Take a minute right now to observe the real-world objects that are in your immediate environment. For each object that you see, ask yourself two questions: "What possible states can this object be in?" and "What possible behaviours can this object perform?" Make sure to write down your observations. As you do, you'll notice that real-world objects vary in complexity; your desktop lamp may have only two possible states (on and off) and two possible behaviours (turn on, turn off), but your desktop radio might have additional states (on, off, current volume, current station) and behavior (turn on, turn off, increase volume, decrease volume, seek, scan, and tune). You may also notice that some objects, in turn, will also contain other objects. These real-world observations all translate into the world of object-oriented programming.

Rules of Objects

There are five basic rules for object as identified by [1]. These include:

(1) Everything is an object
(2) A program is a set of object telling each other what to do by sending messages
(3) Each object has it own memory (made-up by other object)
(4) Every Object has a type
(5) All Objects of a specific type can receive the same message.

Object Classification

We will classify objects into two major parts thus:

- **Tangible Object:** If everything is an object, it then implies that there are those objects that we can feel with our hands in the real life situation examples of such object are fans and human being. This can also be modelled using object-oriented programming to portray the real life situation.
- **Non-tangible Object**: These are object that we cannot feel it with our hands, but we can only sense it, examples of such object in a real life situation are blue cloud, smoke from a car. But all this can be model in object-oriented programming.

User Interface

In information technology, the user interface (UI) is everything designed into an information device with and through which a human being interacts with the system. These include: display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it. In early computers, there was very little user interface except for a few buttons at an operator's console. The user interface was largely in the form of punched card input and report output.

Later, a user was provided the ability to interact with a computer online and the user interface was a nearly blank display screen with a command line, a keyboard, and a set of commands and computer responses that were exchanged. This command line interface led to one in which menus (list of choices written in text) predominated. And, finally, the graphical user interface (GUI) arrived, originating mainly in Xerox's Palo Alto Research Center, adopted and enhanced by Apple Computer, and finally effectively standardized by Microsoft in its Windows operating systems. The user interface can arguably include the total "user experience," which may include the aesthetic appearance of the device, response time, and the content that is presented to the user within the context of the user interface.

Effective Communication

Communication is an important aspect when it comes to object-oriented programming. This is because as a programmer you are communicating with the object-oriented language like MATLAB using two key plat forms. First is the graphical user interface platform which you do the creation of objects to be used in your interface. This in most visual OOP languages is called the *object view*. Secondly, you use the Editor which is the coding environment or *code view* where you do your coding. It is the code that builds life/intelligence into the objects you created using your graphical user interface (GUI).

To be able to put this to work there must be an understanding between the objects created and the code you have written. In MATLAB and any other object oriented language the code must conform to the set of syntax and semantics rules of the language. This implies that the code you are sending to the object to perform a particular task must be understood by the object for the object to give you feedback of the function you want the object to carryout then we can say communication is effective. The diagram in figure 1 below shows the effective communication between objects.

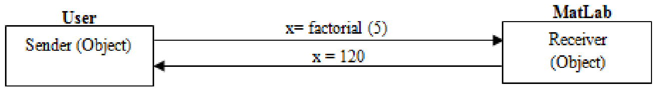


Figure 1: Schematic diagram showing effective communication between Objects

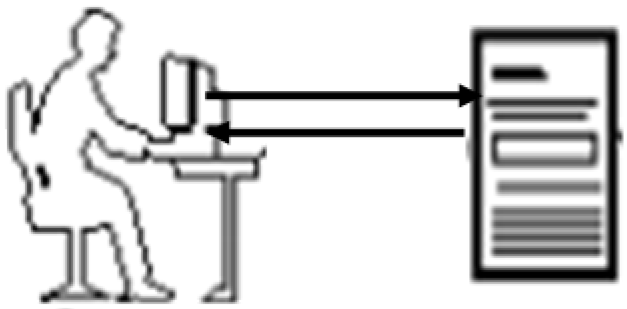


Figure 2: Diagram showing effective communication between Humanware and Application

MATLAB as Object-Oriented Programming Tool

A tool is simply defined as an instrument that is used in performing a specific tasks base on the specific functionality for which the tool is made of. Based on this definition matlab can been viewed as an object-oriented programming tool. The term MATLAB refers to Matrix Laboratory is a programming language use for technical computing, it also have a graphical user interface which is use for the creation of object. Object oriented Programming has a long history that takes its roots in the 1950-60. According to Wikipedia, many of the concepts inherent to object oriented Programming (OOP) were introduced with a version of Simula, a programming language born in Norway. Later on, these ideas were introduced in C++, a newer version of the C language created at the famous BELL lab (Most people know that's where the transistor was invented but for the neuroscientists in the audience, remember that fMRI was also invented there). The introduction of C++ (along with other languages variants like Object Pascal) pushed OOP into mainstream programming during the 80-90.

Matlab object oriented programming somehow took a troubled path. Mathworks first implementation of object oriented programming (in Matlab 5.0, year 1996, thanks for one reader for the info). In 2008, Mathworks released a complete new refurbished OOP that was not compatible with the previous syntax. This post will only use the newer syntax. I invite you to consult the documentation of the oldest syntax (which is still working in the latest Matlab to this date) if you are interested in that piece of Matlab history.

Object-oriented programming (OO) applies to software development using the standard science and engineering practice of identifying patterns and defining a classification system describing those patterns. Classification systems and design patterns enable engineers and scientists to make sense of complex systems and to reuse efforts by others. By applying classification systems and design patterns to programming, the OO approach improves your ability to manage software complexity—particularly important when developing and maintaining large applications and data structures [14].

3. System Design and Implementation

The design methodology used in this research work is the object oriented analysis and design (OOAD) methodology. This is a popular technical approach for analyzing and designing an application or system, by applying the object-oriented paradigm. The design objective will includes designing a system that is user friendly, users can effectively communicate with the system irrespective of their educational background and an ideal learning system that learners communicate and get appropriate feedback. The logical flow of the system is shown in figure 2.

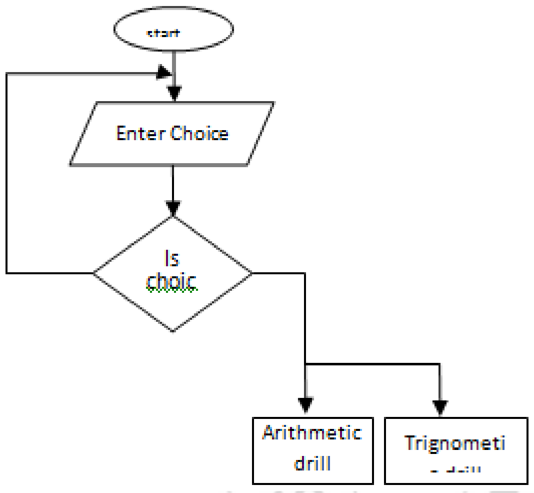


Figure 3: Logical procedure design for Simple Arithmetic and Trigonometric Application.

Implementation of a Simple Arithmetic and Trigonometric Application

In developing our simple arithmetic and trigonometric application, the following steps must be taking into consideration:

STEP 1: Load the MATLAB software; at the command window type **'guide'** to load the graphic user interface (GUI).
STEP 2: Create your GUI platform. Think how your application will look like by creating function and add buttons, text box etc. In our proposed ***Simple Arithmetic and Trigonometric Drill*** we have our designed as stated below:

1. **Arithmetic_Trig_Drill**
   Option 1 : Arithmetic_Drill
   Option 2: Trigonometry drill
2. **Arithmetic_Drill**
   Option1: Addition
   Option 2: Subtraction
   Option 3: Multiplication
   Option 4: Division
   Option 4: Factorial
3. **Trigonometry**
   Option1: Sine
   Option2: Cosine
   Option3: Tangent

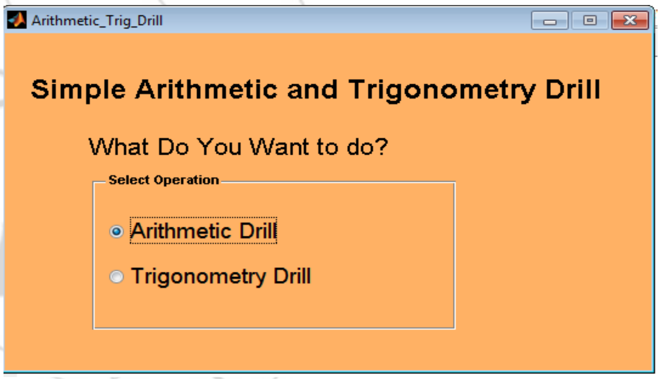
STEP 3: After you are done with all the items on the **simple arithmetic and trigonometry drill** and setting and you can now open the **Editor** section.
STEP 4: Finally we have to program each and every button functions as according to our requirement.
In programming there are some functions which are used many times like.
1 **Set**: this is used to set any block when this particular push button calls.

Like:
Pushbutton1……….
set (handles.text1, 'string', 'num1');
means whenever pushbutton1 will be called it set text 1 as num1

2 **Get**: this is used to take data from any block

```
num1=get(handles.text1,'String');
num2=get(handles.text2,'String');
num3=get(handles.edit3,'String');
numadd=strcat(num1,num2,num3);

numaddResult=eval(numadd);
set(handles.text6,'String',numaddResult);
```

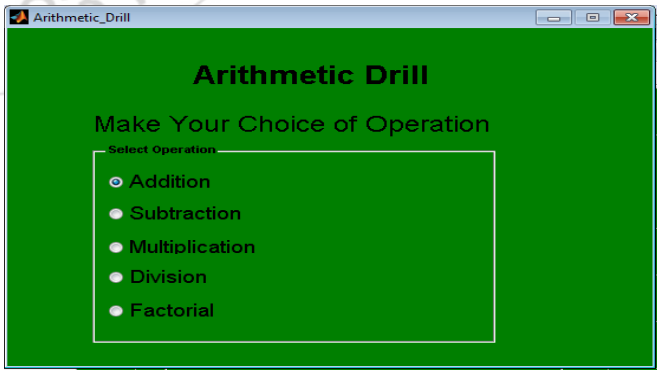


Simple Arithmetic and Trigonometry Drill

What Do You Want to do?

Select Operation
- [✓ Arithmetic Drill]
- Trigonometry Drill

Figure 4: Arithmetic and Trigonometry Drill Main Interface Design



Arithmetic Drill

Make Your Choice of Operation

Select Operation
- [✓ Addition]
- Subtraction
- Multiplication
- Division
- Factorial

Figure 5: Arithmetic Drill Interface

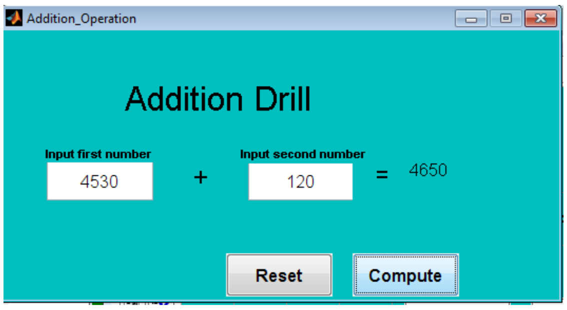

Addition_Operation
Addition Drill
Input first number: 4530 + Input second number: 120 = 4650

Figure 6: Addition Drill interface with data

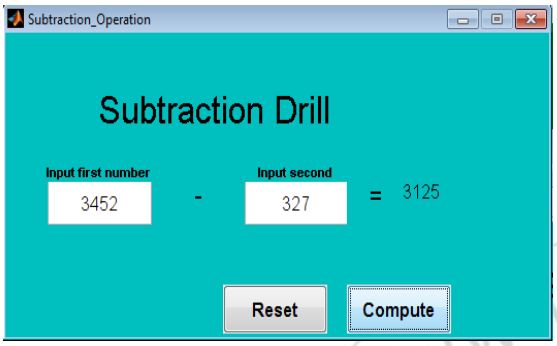

Subtraction_Operation
Subtraction Drill
Input first number: 3452 - Input second: 327 = 3125

Figure 7: Subtraction Drill interface with data on it.

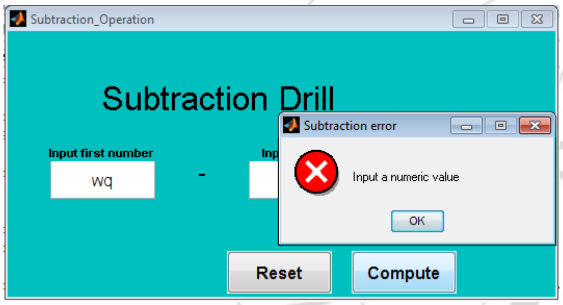

Subtraction_Operation
Subtraction Drill
Input first number: wq -
Subtraction error
Input a numeric value

Figure 8: Subtraction Drill interface showing error message when wrong input is made

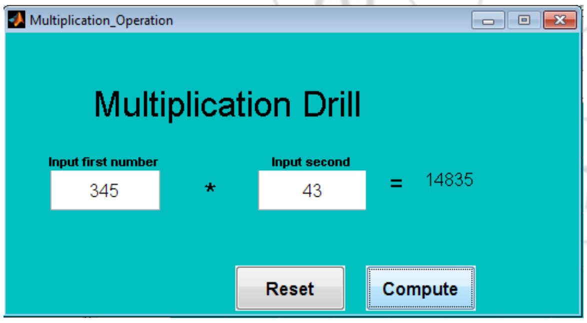

Multiplication_Operation
Multiplication Drill
Input first number: 345 * Input second: 43 = 14835

Figure 9: Multiplication Drill interface with data on it

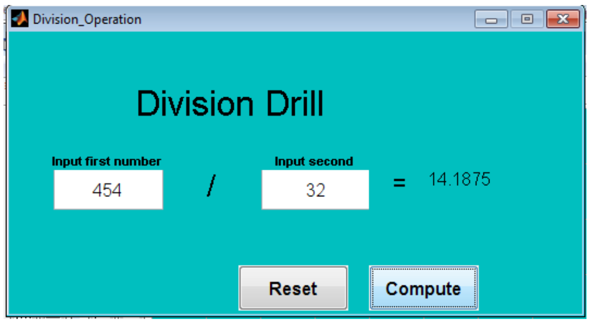

Division_Operation
Division Drill
Input first number: 454 / Input second: 32 = 14.1875

Figure 10: Division Drill interface showing with data on it

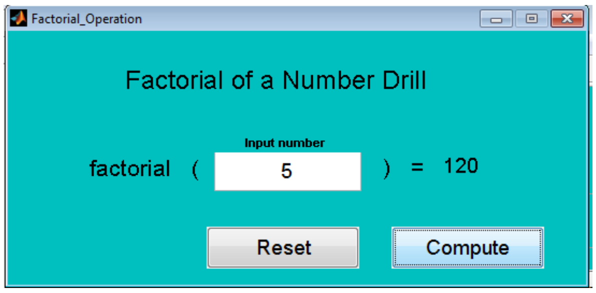

Factorial_Operation
Factorial of a Number Drill
factorial (Input number: 5) = 120

Figure 11: Factorial Drill interface with data on it

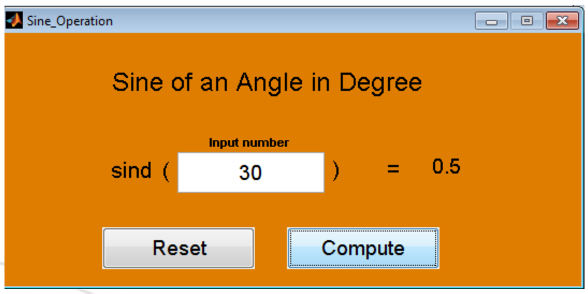

Sine_Operation
Sine of an Angle in Degree
sind (Input number: 30) = 0.5

Figure 12: Sine of an Angle in Degree showing result on it

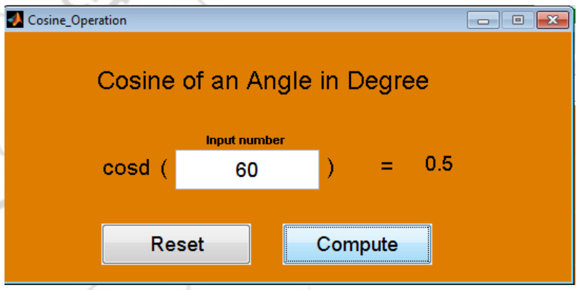

Cosine_Operation
Cosine of an Angle in Degree
cosd (Input number: 60) = 0.5

Figure 13: Cosine of an Angle in Degree

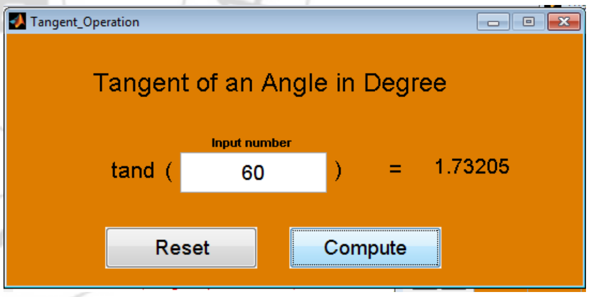

Tangent_Operation
Tangent of an Angle in Degree
tand (Input number: 60) = 1.73205

Figure 14: Tangent of an Angle in Degree

4. Results Presentation

Figure 4 shows the main interface of our Simple arithmetic and trigonometry Drill. When the user chooses *arithmetic drill*, figure 5 displays the options/sub-menus for that choice. Figures 6 through 10 show the interfaces for evaluation of the mathematical operations namely: addition, subtraction, multiplication and division. Figure 11 shows the interface for the factorial operation. And finaliy figures 12 to 14 display the trigonometric functions of sine, Cosine and tangent of angles.

5. Discussion

The user can now communicate effectively with the developed simple arithmetic and trigonometric drill application through its user interface. When the user first loads the MATLAB software and then type (‘**Arithemetic_Trig_Drill.fig’**) at the command window guide, then click on run. The application’s main interface shown in figure 4 displays. If the user click on the first option, figure 5 is displayed and the user can now make a choice by selecting any arithmetic operation. If the user selects Addition option it displays figure 6 for the user to enter number in the edit text box 1 and edit text box 2 and when the user clicks on the compute button, the application displays the result on the static text box created in the design as shown in figure 6.

In addition, our simple arithmetic and trigonometry Drill application has some in-built validation technique on the interface. This is to prevent wrong data entry. When a wrong entry is made, the application will send alert to the user with a context sensetive error message as shown in figure 8. This will enable the user to make correction by entering the correct values for all input in the edit text boxes as shown in figures 7-14.

6.0 Conclusion

In conclusion, object oriented programming languages have proved to become dispensable tools for effective communication on computer applications’ user interface. OOP Languages are useful in virtually every aspect of application development due to their flexibility, amiability, modularity, library of functions and the simplicity of coding each object that is generated on a graphical user interface. The simple arithmetic and trigonometric drill application which is developed in this paper has demonstrated the ascertion.

Code colourization, syntax checking intelligence and other tools built into OOP languages help programmers to write codes both quickly and correctly. By using an intuitive graphical debugger programmers can quickly find and fix programming errors that may have prevented their application from running correctly.

References

[1] Alan Kay (2010), Daniel Blog http://searchsoa.techtarget.com/news/962762/Smalltalk-with-object-oriented programming-pioneer-Kay retrieved 28 June 2016.
[2] Krubner, Lawrence.(2014) "Object Oriented Programming is an expensive disaster which must end". smashcompany.com. Retrieved 14 June, 2015. www.smashcompany com/technology
[3] Meyer B. (1988). “Object Oriented Software construction” prentice Hall International.
[4] Sarah E.H. and Stacey C.S. (2000). Computers, Communications and Information (Comprehensive Version). USA: McGraw-Hill Higher Education. Pgs. 10.24-10.25
[5] 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 on?? from *http://cs.wellesley.edu/~fturbak/pubs/VLHCC-2012-paper-turbak.pdf*
[6] 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*
[7] 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*
[8] Rene Sieber, Regula Stopper, Samuel Wiesmann and Olaf Schnabel(2012), Graphical User Interface-Layout and Design retrieved on 28 June 2016 at *http://www.e-cartouche.ch - Version from: 26.1.2012*
[9] Klaus Hinum (2004), Human centered Design for Graphical User Interfaces Vienna, *Institute of Software Technology and interactive Systems. [E188] submitted at the Vienna University of Technology, Faculty of Informatics*
[10] Onu F.U., Ikedilo O.E., Nwoke B.O., and Okafor P. (2015), Importance of Object-Oriented Programming in this era of Mobile Application Development, *IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. I*
[11] Onu F. U., Osisikankwu P. U., Madubuike C. E. and James G (2015), Impact of Object Oriented Programming on Web Application, *International Journal of Computer Applications Technology and Research Volume 4– Issue 9* pp. 706-710
[12] Baah B. and Taylor O. E. (2013); Parameters for the Evaluation for the Choice of a Good Programming Language, *Journal of Computing Technologies(JCT)*, Vol. 2, Issue 8 pp. 6-11
[13] Peter Wegner (1989); Concepts and Paradigms of Object-Oriented Programming, Expansion a keynote Talk, Brown University
[14] Stuart McGarrity, MathWorks (2008), Introduction to Object Oriented Programming in MATLAB, *Technical Articles and Newsletter*