Journal of Advances in Science and Technology
Vol. 22, Issue No. 4, September-2025, ISSN 2230-9659

DOI: https://doi.org/10.29070/d63z8c30

An analysis the oop difficulties and comprehending the problem for students and teachers

Maheshwari M [1] * , Dr. Kishan Kumar [2]

1. Research Scholar, Shri Krishna University, Chhatarpur, M.P., India
ouriginal.sku@gmail.com ,
2. Professor, Shri Krishna University, Chhatarpur, M.P., India

Abstract: This study analyzes the challenges faced by students and teachers in understanding and teaching Object-Oriented Programming (OOP). the researcher has found different techniques to learn and teach OOP and Java. Crucial components are object visualization and object interaction. An object, in OOP, is a persistently identified item, instance of a class, or thing. It identifies key difficulties such as grasping abstract concepts, applying inheritance and polymorphism, and managing the complexity of program design. For teachers, challenges include devising effective teaching strategies and addressing diverse student learning paces. In the analysis, maintenance, and design of complex systems, the object idea is a meaningful, useful, and adaptable organizational device from the perspective of the programmer. A great deal of progress has been made thanks to inheritance, the most crucial principle of OOP. The crucial semantics work can be carried out on polymorphism & subtyping.

Keywords: OOP, Java, Education, Teaching, Students, E-Learning

X

INTRODUCTION

Object-oriented concepts are hard for students. It is also difficult for lecturers or instructors to convey these ideas. These ideas represent a shift in pedagogy from the decade in question. In order to identify which concepts students are struggling with, authors will sometimes review the results of previous class exams. We found that students could use the skills to get around OOP's challenges. Using the results and course materials as a guide, they primarily focus on those parts of OOP that are necessary. Findings from this study highlight the many challenges educators confront when attempting to introduce pupils to programming. Students often struggle to grasp the more advanced ideas of OOP, such as classes, friend functions, overloaded constructors, abstraction, polymorphism, and more. With students coming from a procedural programming background, they find it quite difficult to transition to OOP. The pupils are wasting a lot of time trying to grasp these OO ideas. According to Abdullah Mohd Zin (2006), OOP allows for a program to contain a number of classes. In our opinion, object-oriented languages & frameworks, such as Java, are winning over more and more academics and businesses. There is a huge amount of OOP in use nowadays. Industry demand between wide edge & expertise of students in OOP. Instructors and students alike have a larger academic obligation to master OOP. The nature of the problems can be better understood when certain circumstances are met, which allows object-oriented approaches to overcome lower performance. Learning the fundamentals of OOPs in Java might be challenging for normal students since they don't have the same level of familiarity with the language as they have with C++.

DATA HIDING

The focus of design is moving from program design to procedure design, which in turn leads to data organization. When the aforementioned designs are converted, the program size increases. A collection of interconnected operations that work with the changed data is called a module. By dividing the program into modules and using the programming paradigm to decide on each one, the data is kept concealed. A predetermined paradigm is this data hiding principle. Even without data grouping, the procedural programming style is enough. Each and every one of a module's procedures undergoes the process of procedure design improvement. The definition of a stack module is seen here. The provided problem has good solutions: the stack's user interface is accessible through the functions push() and pop().Before being used for the first time, the stack is giving initial values.

```
// the module stack of characters interface can be declared as
const stack_size = 00;
char pop();
void push(char);
stack.h file can store the given interface and the other parts can be defined as…
#include "stack.h"
static char k[stk_size];
static char* pt = k; // Stack value is empty at start
char pop()
{
// conditions check for pop and underflow
}
void push(char ch)
{
// conditions check for push and overflow
}
```

It should be possible to swap out the stack with a linked list. The user must not be able to access the representation. In this case, k and pt define local, file-or module-specific static variables [Shivam 2013]. One way to use a stack is as follows:

```
#include "stack.h"
void function1()
{
char ch = pop(push('c'));
if (ch != 'c') error("impossible");
}
```

This function is not supplied by Pascal if the technique for hiding the remaining code is made local to the procedure. Nesting procedures & global data are both aided by this. A module is a group of related function & data definitions stored in a single source file; this is one way in which C differs from other programming languages. Achieving a degree of modularity is possible using C. Use a static declaration when no specific format is required. A large body of research indicates that OOP significantly improves students' ability to grasp core topics. These kinds of ideas are necessary for beginning computer science courses. When students conduct their programming projects independently, they often come up with their

own interpretations, which isn't always accurate or thorough enough to pass on to their teachers. One must familiarize oneself with theoretical definitions and make an effort to grasp formal aspects in order to comprehend programming principles. Both theoretical & practical knowledge are required, and the two go hand in hand. For OOP technology, learning kinds are necessary.

PROGRAMMING LEARNING BY STUDENTS

It is possible to uncover the significance of learning program in earlier studies; nevertheless, this question must be brought up to pupils at the time that they are comprehending the material. A number of writers have investigated engineering students' perspectives on the meaning & requirements of the learning program. Investigators looked studied how first-year students learn to code using a variety of approaches. Students must have a thorough comprehension of what it means to learn to program, as seen in both cases.

Utilization of resources by learners

Articles from journals & conference papers that provide resources for learning to program are referenced. Students in the current study and anyone in the field of computer science education can make use of the materials. After that, they looked into related work on the study's resources. Collaborative techniques, programming tools, & pair programming were some of the technologically-supported resources employed in problem-based learning. Students within technology-supported resources utilize the compilers in their current studies.

Programming language learning difficulties with thinking-based learning

The curriculum for learning a new way of thinking is based on personal experience. When it comes to the programming language, the learning is both challenging & confusing. To convey the then-standard method of learning to code, which takes the shape of programming thought, and to highlight the gaps in that knowledge. Some of the pupils really believed that programming required exceptional intelligence and skill. The majority of students must be able to recognize when a programming problem is present; a small percentage may even be able to demonstrate this using only their own knowledge or reasoning. Learning to program also makes use of its history and certain programming languages. Real thought and comprehension are required when programming. Student discussions center on the fundamental thinking skills needed for programming.

STATUS OF PARALLEL OOP

Parallel Object-Oriented Programming has been the subject of an overwhelming amount of study over the past decade. Many OOP languages have been developed, evaluated, and put into use for comparable purposes. There are a lot of problems and obstacles in the stages of parallel OOP. Different versions of the OOP languages are optimized for different tasks, such as inheritance, automatic memory utilisation, capacity, heterogeneity, debugging, & user-friendliness.

Inheritance Capability

Objects that can be partitioned in a network and that perform tasks such as handling and accepting messages sequentially are not inheritable in all languages. You can only inherit from one class at a time in

some languages that support base inheritance. When it comes to inheritance and the synchronization of code for parallel objects, the majority of languages fall short.

Ease of use

A wide range of experimental languages and C++ extensions are covered in Parallel OOP. It is not easy to study, implement, or utilize these extensions because they are big & complicated. Languages that are oriented toward interpretation, such as Java, Smalltalk, and others, do not offer efficient type checking or a large amount of runtime capacity.

Heterogeneity

Nowadays, there is a great deal of heterogeneity in the computing environments. Users get access to many platforms through personal computers & workstations that are either locally or globally dispersed. For consistent networks or precise high performance platforms, the majority of object-oriented languages are designed. It is possible to convert between different types of OO computing environments that use heterogeneous networks, even if they do not support the same compilation oriented languages.

OBJECT INTERACTIONS

A system's architecture and its behavior over time are defined by the interactions of objects utilizing structural relations. The finesse offered by communication links. Just by indicating a message link, the objects can be transmitted. It is common practice to identify message coupling after structural links have been resolved. In order to communicate objects, one must understand the functions that objects request. So, let's pretend that the association relations example is true, and that the plausible realization of a relation between an employee or boss is achieved through the switch of numerous messages. You should think about the paye, displayRes, and performActivity functions. Decisions made during the design and execution stages should alter object communication. The fundamentals of software engineering often include the bidding of data function and the tendency for modularity to strike with accomplishment objectives. In order to meet the requirements for the intended outcome, the means of communication between objects may evolve throughout the design and construction processes. [Holland 1997]

Issues with Interactions between Objects

a. Multiple Views

It is unnecessary to pay attention to other instances that make use of the object operation's utilities. For the items provided, different views can be defined for different customers. Imagine a classroom setting where the class teacher and each student, Stud, are responsible for their own education. The TeacherReg & StudentReg classes also have a managerial function. Teachers are considered to be students for the purposes of taking these classes. In certain instances, the success of the instructional framework is dependent on the teachers involved. We learned about object-oriented methods, but they can't handle objects with various perspectives. A graphical object written in languages like Owl, C++, & PAL. In general, the process that distinguishes between an object's subclasses, the object itself, and other client objects. It is not possible to differentiate between objects while dealing with various kinds of external client objects. Although it is not mandated by the language, the Smalltalk programming environment does

familiarize users with the idea of private operations. Adding a new object for each view makes it impossible to simulate numerous views.

a b
a b
a is registred in b
a follow course from b
(a) (b)
Teacher1
Registration1
Student1
Registration1
Teacher1 Teacher2 Student
Teacher1
Registration1
Student1
Registration1
Teacher1 Teacher2 Student

A Language-Database Integration and Queries Framework

Database problems in object-oriented methods are not resolved by software development processes such as structures, queries, transactions, and data preservation. There is a lack of ways that tackle database difficulties; all that is respected is the definition of preserving objects. The necessity to govern has taken a toll on two distinct languages & data structures. The goal of this framework's many management efforts is to integrate two independent systems using the object-oriented paradigm. In order to incorporate higher-level object-oriented models, computational models are used in both data management & application programming. Data handling, querying, and transmission are the three main components that modern object-oriented database systems rely on. However, there is a downside: combining language with systems databases is not going to be completely comfortable. An object-oriented computation model for systems like query languages would augment the traditional database methods; nonetheless, the programmer would still be required to deal with two separate systems. The developer is compelled to manage two separate systems due to the requirement of object type links, explicit object lookups, and the SQL interface. In most cases, encapsulation is compromised when the object-oriented language paradigm incorporates database-like characteristics. Only objects belonging to certain classes in smalltalk and gemstone can be queried. One issue with Orion's method is that queries are defined on all instances of a class, which results in sets. Unfortunately, these sets cannot be further filtered. Inquiries guide the direction of aggregates and classes. [Odunaike SA 2012]

OBJECT-ORIENTED FRAMEWORKS

One of the greatest things about Simula 67 is the object-oriented structure it provides. These libraries of subroutines were widely used, and they always functioned as a subprogram to the main program. Subroutines p and q are always called to handle the relation. The one and only exception to this rule is when a library subfunction q takes a user-written subprogram p as an input. In this instance, q can actually invoke p. All that has to happen for a call to go from the main program to q and then back to p is a single

call. Subprograms are allowed in an Object-Oriented framework, which allows for the inverse relationship between the main program and components. The programmer mimics the code that describes how the system's instances behave. There must be a connection between the domain being illustrated and the things being sold or the nuclear fuel rods. A simulation framework is another name for user-defined objects. It is the framework's responsibility to manage the user-populated goals. Although providing these functions may require a significant amount of effort, users are just required to optionally override the inherited behavior.

From Simula to Smalltalk

Smalltalk-72, the version utilized exclusively by Xerox Parc, was the most recent version of Smalltalk. Inspired by computer animations, the concepts of objects, classes, inheritance, and object references were developed. As a tiny computer, the instances idea is investigated by Smalltalk-72. Some object-oriented languages differ in the importance they place on inheritance, classes, and instances. Taking into account Smalltalk, C++, & local Lisp Object System, it classifies the features of modern instance-oriented languages. Important ideas, according to certain writers, are as follows.

- Determine demands for operations.
- Retrieve the requested instance.
- Create new instances as needed.
- The various objects can have the same functionality applied to them.
- An instance is a representation of an abstraction.
- Objects' provision of services.
- Customers have the option to request services.
- Encapsulating instances is being discussed.
- Classification is based on object services.
- Objects are able to share code.

Objects as Abstractions

In informatics, the word "abstraction" can mean various things. Functional programmers consider how an object's features have changed over time. With the passage of time & transatlantic movement of things, Simula's concept of procedural encapsulation got increasingly refined. Other characteristics of the simulation class, such active objects & modules for classes, have vanished. To convey the idea that an object's protocol is what matters when considering it in relation to other objects, the phrase "abstraction" is used. That which it does in reaction to such messages, or sets of messages, is sometimes called the object's interface. The essential idea of abstraction is that while we use an object, its internal structure remains concealed from view. Although it is implicit in the concept of abstraction, it is one of the fundamental principles underlying objects. Modeling is instead discussed, which emphasizes the significance of interacting with the item; nonetheless, the information concealing component is neither mentioned nor

addressed. For example, let's pretend it's unfortunate that the Car process's variable attributes couldn't be contained in a subblock, with the procedures defining the Car's behavior exposed.

QUANTIFIERS AND UNDERSTANDINGS

When describing items, there are well-known quantifiers that come in handy. A set of understandings discovered by mathematicians through the practical reduction of notation. A list of other functional languages that make optimal use of comprehension notations. Both of these descriptive languages feature generalized quantifiers & comprehensions, which are vital and helpful in their own right. Whereas generalized quantifiers are a kind of notational shorthand & comprehensions are literals for modeling types. The key features are the usefulness of comprehension notations or set comprehensions. So, let's say we're thinking of using a generalized quantifiers number that adds up all the numbers 'r' that satisfy the limits predicate and 0<=r && r\args. The set determines the length and the body. Quantifiers that are invariant in nature are utilized in a loop. Spec language designers face multiple challenges when dealing with expressions, the most common of which being quantifiers & comprehension. As an example, we can see how sequentially objects-oriented algorithms describe & validate garbage collection sensitivity when non-garbage objects quantify just quantifiers.

PROGRAMMING TEACHING

Learning to code is an essential ability for anyone majoring in computer science. There is a correlation between programming quality & ability to bind ideas through design and testing utilizing one's computer skills. Programming students learn a lot of useful abilities, like analytical thinking, depreciative substances, & component details, which are useful for various professions. The programming shown in the articles written by researchers must be a part of any respectable curriculum. Teaching programming in the conventional way has its fair share of problems. According to numerous academics, learning programming is an incredibly intriguing pastime. More than half a century after its introduction, it is still seen as a difficult subject to master. This is because the programming process is not well-understood in current pedagogical practices. Teaching programming is an even more daunting task than practicing the sophisticated art form itself, which involves a close relationship between science & art. C++ is an introductory, widely used programming language that is covered in intermediate courses at several other colleges. The current demands for high dependability or security, the difficulties students have while learning new languages and improving their programming efficiency, and the need to reorganize the course objectives to make them easier to understand without sacrificing the quality of instruction. The primary driver behind that was the new C++ standard, & academic community will learn about the key changes either later on or sooner.

Computer Science Programs at Our Institution

C++ is being developed primarily as an introductory programming language for bachelor degrees. Prior to the application of program paradigms and languages, the fundamentals of procedural & required programming are covered in the introduction to programming. Afterwards, we move on to items that introduce learners to new concepts. Introduction courses cover topics such as data types, arrays, files, operators, or technology that can be utilized for program testing. The most solemn call for paradigm shift

was the transition from procedural languages to object-oriented programming, or OOP. For the first few weeks of class, students learn the fundamentals of the language through object-oriented programming, with an emphasis on the language's operators, functions, expressions, and components of the traditional procedural model. Acquiring proficiency in these methods is fundamental to their training, and they face a variety of challenges when studying OOP. It integrates aspects of object-oriented design analysis and applies comparable research in software technology & OO programming.

OBJECT-ORIENTED PROGRAMMING

Move Semantics

Our starting element is the rvalue reference type. The use of rvalue references makes the implementation of move semantics simple. That absence of perspective is the most important part of C++11. The compiler was able to swap out the expensive process of object copying for the less expensive "state movement" procedures made possible by move semantics. C++ makes learning OO programming more of a challenge than languages like Smalltalk, Java, or Eiffel, which are pure OO languages. Problems arise with C++'s code efficiency and logic, which are inherited from C. Every class with an encapsulating complicated data structure, data members, and external resource pointers should have three methods: copy, constructor, & destructor. This is an example of object-oriented programming. Copies of both the constructor & assignment operator reliably establish the semantic value or result. Duplicating resources might be a pain, but there are occasions when it's necessary and useful for data copying and flawless operation.

Assume for a moment that we are thinking about a case where two objects are swapped, such in string concatenation, container reallocation, or swap operations.

```
template <class T>
void swap1(T & p, T & q)
{
T temp1(p);
p = q;
 q = temp1;
}
```

When the parameters are either integers or Booleans, the construction of swap() works. Even when objects are only containers, the downside of this method becomes more apparent. One striking example of incapability is the copying that is done utilizing the container reallocation mechanism. Using the new memory buffer to store already-allocated container elements and calling the copy constructor on each element of the new buffer are both necessary steps. Restoring the original items & freeing up memory for the original buffer are both accomplished via the destructor. To build the semantics of moving, which will be used to design code for dynamic memory resource transfers between objects. Swapping the values of strings can be illustrated by exchanging the values of internal pointers & sizes of objects. A unique function is needed for the building of this operation. The transfer of resources from an old reference type provides these features. Even before C++11 was standardized, the major compilers were available for use with C++10. When combined with the aged reference type, coding move semantics is a breeze. Reference semantics works precisely, which is another point of the program that cannot be changed. Standard

Template Library (STL) changes occur in tandem with language progress. Container operations & move semantics technique are both optimized for use. The STL approach greatly improves the semantics for outcome efficiency when dealing with large & complicated structures. Students are the future of computer programming, thus teaching them to produce reliable, efficient code should be a top priority. A unique location in OOP for semantic copying. For the same reason, we can't disregard the semantics of the motion. Regardless of whether they have received specialized training, students are expected to utilize the containers & algorithms belonging to the Standard Template Library. Students in the basic course learn about char strings and how to exchange two strings utilizing the operator std::swapi(s1,s2). They also utilize the template version of convert. For the purpose of guiding the architectural and design careers of the future. The definition of a transform assignment operator & move semantics for a specific class's constructor is required. The following example demonstrates basic syntax formation learning & presence of a memory pointer buffer for individual data members [Shivam 2013].

```
class K
{
char* da;
public:
K(K&& other):da(std::move1(other.da))
{
// move1 constructor
other.da = nullptr;
}
K& operator=(K&& other)
{
//move1 assignment operator
if(this == &other) return *this;
if(da) delete[] da;
da = std::move1(other.da);
other.da = nullptr;
return *this;
}
//.....
};
```

Assigning & initializing pointers and member functions requires setting them to their default initial values first. Here, the destructor releases resources, like dynamic memory, in a continual fashion. The benefits of transfer are not limited to classes or individual functions. The student suggests familiarizing yourself with std::move1() and utilizing it frequently. Predefining the template function swap1() and placing it below is a good way to execute this.

```
Template < class T >

Void swap1(T& x, T& y)

{

T buf (std::move1(x));
```

```
x = std::move1(y);

y = std::move1(buf);

}
```

The complex & complicated nature of object orientation causes many students to become confused when they begin certain sessions. Problems prompt some students to drop out of the class. Even after teachers give them the correct solutions, students often repeat the same bugs. Until they are unable to solve issues on their own, a teacher or instructor will explain what they need to know. If a student is struggling, they should talk to a teacher or instructor about it.

There are a number of reasons why this could happen:

- It's not always easy to tell how often students make the same mistake, which makes it hard to identify trends in their approaches to solving problems.
- It is challenging to know how to overcome the preconceived notions of many students. For instance, there are students who are resistant to studying object-oriented programming since they already know a lot about procedural languages like Basic. Interpreting problems from the student's perspective can be challenging, and there are situations when the student knows but doesn't comprehend.
- It's challenging to examine each pupil individually and account for all common mistakes or learnt material.
- In a large class, there isn't enough time to meet the needs of each student, and the teacher or instructor doesn't know why some students are struggling.

According to some researchers, students with learning disabilities can be helped by using a one-on-one teaching approach that provides personalized attention. This tutoring system is finding applications in both academia & industry, where it is helping employees learn new skills. Nowadays, OOP is getting a lot of attention in the IT world, particularly in relation to procedural programming languages like Java & C++.

ENCAPSULATION AND DATA SECRECY

The notion of information concealing is the process of taking an instance, erasing its internal data, and then providing a communication medium between objects in a way that solves potential outcomes. Think about a scenario where the object is accessible anytime data and instance variables that are meant to hide information or provide a protective model are needed. The use of OOP allows them to provide classes with clearly defined interfaces to their objects. Here we take a look at a C++ example that uses public, protected, & private members as well as general binding protection appliances. Members can access public data and member methods from any location. Only students in a certain class have access to private members. Subclasses are the only ones that can access protected members. All list array data must be kept private. Creating data binding relies heavily on the design of distinct instance or object classes that share a common user interface. All of the objects respond in the same way to the same message, but they'll all use class-specific actions to carry it out. The program's implementation up to the point of receiving objects through message sending, with an emphasis on reducing dependencies & increasing the quantity of code

that can be exchanged and reused. Assume, for the sake of argument, that the vehicle and driver interact with one another via the same object, and that the car's engine is one such example. Because every driver is familiar with this protocol or object, they may apply it to any vehicle's engine. Information technology details are disseminated from the driver and the remainder of the vehicle. Data abstraction, which takes into account binding of data & polymorphism, is a perk of the object-oriented paradigm.

DIFFICULTIES IN RURAL EDUCATION FOR STUDENTS' LEARNING

In a 1963 lecture delivered at an education conference in Chicago, the term "learning difficulties" was coined by psychologist Dr. Samuel Kirk. To put it simply, learning disabilities are "a complex group of disorders exhibit by consequent difficulties in acquiring and efficient utilization of listening, reading, speaking, writing, reasoning & mathematical abilities," according to the National Joint Committee on Learning Disorders (NJCLD). According to Pandey (2014). Students in remote areas may encounter numerous out-of-classroom learning challenges nowadays. There is a correlation between the exterior environment of colleges or educational organizations and learning challenges, which are problems with processing information based on the nervous system. While previous studies on the topic have shown that students' net-based difficulty impacts their competence, conduct, learning quality, adoption, & thinking power, they have also raised some questions. Finding the forms of difficulty with OOP learning and how to solve problems using computer-based learning are our primary study objectives. The primary goal of this study is to identify areas of OOP learning difficulty. A lot of academics have their own ideas about how to describe OOP learning difficulty. To function independently in a numerical society, one must possess strong mathematical abilities, which impact one's educational and occupational prospects, and ultimately one's socioeconomic standing. The development of numeracy as a concept & symptoms of difficulty in acquiring and using it are both crucially important topics to comprehend. It is critical that young boys develop self-confidence in their abilities to read, write, & think critically in today's high-tech, more linked world. This confidence will help them succeed in school and in life. A single, agreed-upon definition of a learning disability among experts is crucial. There are a number of reasons why this group does not learn, but academics do agree on one thing.

ASSISTING WITH THE IMPLEMENTATION OF ICT IN EDUCATION

With the use of inclusive pedagogy and information and communication technology (ICT) tools, inclusive education offers a fresh vision for how students with special needs can participate in regular classrooms just like their typically developing peers. In order to accomplish this, we had to make sure that the necessary conditions were in place to help students in rural areas overcome the obstacles they faced while trying to learn. These prerequisites are achieved, in particular, through the training of ICT specialists in SNE, the development of ICT infrastructure for SNE, & integration of ICTs into SNE curricula (UNESCO, 2006). To ensure that students with special needs have access to the best possible learning environment, an inspiring information and communication technology infrastructure for special needs education is essential. It is impossible to construct any kind of comprehensive educational region under current circumstances without using the appropriate information and communication technology instruments. For students with special needs to actively participate in their education, it is necessary to implement assistive technologies that make use of specialized methods and instruments. Affirm the incorporation of new learning

experiences and curricular revisions.

This allows for the possibility of meeting the unique educational requirements of various student populations, including those with learning disabilities. There is a great deal of variety and variety in the specific uses of ICTs, however they can generally be classified into the following broad groups:

- Compensation uses.
- Didactic uses.
- Communication

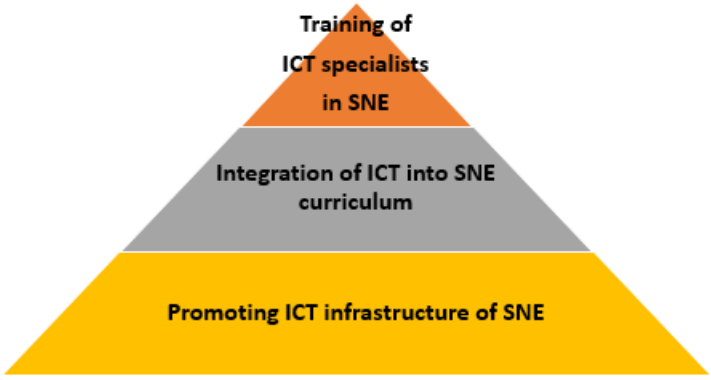

Figure 1: ICT infrastructure

CONCLUSION

The multifaceted challenges in learning and teaching Object-Oriented Programming, emphasizing the need for a more structured and engaging approach. For students, hands-on practice, real-world problem-solving scenarios, and interactive tools are critical for bridging conceptual gaps. Teachers, on the other hand, require robust teaching resources, ongoing training, and adaptive curricula to effectively impart OOP concepts. This study details the challenges that students and learners have when attempting to learn OOP in both urban and rural settings, and how they were able to overcome these challenges through the use of computer-based E-learning technologies. The government will have an easier time providing educational programs & facilities through networks and ensuring educational equity in the country when this sort of framework technology is used in educational institutions or universities.

References

1. Alepis, E., & Virvou, M. (2010). Object oriented architecture for affective multimodal e-learning interfaces. Intelligent Decision Technologies, 4(3), 171-180.
2. Bashir, G. M. M., & Hoque, A. S. M. L. (2016). An effective learning and teaching model for programming languages. Journal of Computers in education, 3, 413-437.
3. Fachrizal, M. R., & Ramadhan, F. (2018, August). Design of web-based e-learning application. In IOP Conference Series: Materials Science and Engineering (Vol. 407, No. 1, p. 012138). IOP Publishing.

4. Jayanthi, M. K. (2009). *Object Oriented Analysis and Design of Learning Objects And Applications of Agent Based Reusable Learning Objects in e-Learning System Design* (Doctoral dissertation, SRI Chandrasekharendra Saraswathi Viswa Mahavidyalaya).

5. Michael Blumenstein, Experience in teaching object-oriented concepts to first year students with diverse background, ITCC-2014.

6. Soly Mathew Biju, Difficulties in understanding object-oriented programming concepts, Netherlands, p.p-319-326,2013.

7. Taheri, S. M., Sasaki, M., & Ngetha, H. T. (2015, July). Evaluating the effectiveness of problem solving techniques and tools in programming. In 2015 Science and Information Conference (SAI) (pp. 928-932). IEEE.

8. Vaishnavi J.deshmukh et al (2013) "cloud computing system for E-learning: A design and development approch" international journal of advanced research in computer science and software engineering, vol-3, issue 5, may 2013, ISSN:2277 128X .

9. Wei, F., Moritz, S. H., Parvez, S. M., & Blank, G. D. (2005). A student model for object-oriented design and programming. *Journal of Computing Sciences in Colleges*, *20*(5), 260-273.

10. Xiao-dong Zhu, Teaching adaptability of object oriented programming language curriculum, International Education studies, Vol-5, No-4, 2012.