DERIVATION OF OBJECT ORIENTED PROGRAMMING FROM REAL LIFE HUMAN SOCIETY

Abstract

We can observe that all programming principles (object-oriented or structured) are taken from real-life human civilization throughout this entire talk. Programming principles may be easily grasped if we pay attention to our daily lives. In other words, we should pay attention to the environment and brush up on our programming and logistical skills. A class is a collection of variables and functions. Why is it possible for functions to exist independently of objects? Why is encapsulation important and what is its purpose? What is the origin of the notion of inheritance? Why is it necessary to have constructors and destructors ? Generic and polymorphic programming have their genesis in this question. What is the origin of multitasking and how is it applied in programming?

Authors

Arif Md Sattar
Assistant Professor
Department of Computer Applications and Information Technology
Anugrah Memorial College, Gaya
amsattargaya@gmail.com

Md. Shadab Shams
Research Scholar
Magadh University
Bodh-Gaya, Gaya - Bihar
itsmesshadab@gmail.com

I. INTRODUCTION

An item is something that can be handled by a human being. An object is a sort of element that can be recognized by others. Each item must have a unique set of distinguishing characteristics in order to be recognized as distinct from the others. Personal information such as a student's first and last names and a student's roll number are examples of personal information. A property that serves as an identifier is termed a variable in the C programming language, a data member in C++, an instance variable in a DBMS field, etc. To put it another way, variables may trace their origins back to physical objects. In actual life, everything is enclosed or bound in some way. Data binding, often known as encapsulation, is the act of tying together variables and the functions they correspond to into a single object. In the case of a car, for example, a nut, bolt, and screw are used to join several elements together into a single unit. Encapsulation of various organs in the human body is also a form of the human body itself. Encapsulation in programming, in my opinion, is a natural outgrowth of real life.

It can provide a variety of benefits, including data concentration, rapid data transportation, security, and a small footprint. Encapsulation may be used to get all of these features. Some jobs can be done by any given item. The object's existence will be deleted from the system if it is unable to execute any task. For example, when people reach old age and are unable to accomplish anything for themselves, they are doomed by God. When a person is unable to work or is incapacitated, they are stripped of their rights and privileges in society.

In order to do any activity, one must use some type of give-and-take process (Len Den), in which one provides and one receives. Electricity is provided and air is created in the operation of a fan; similarly, electricity is supplied and light is emitted in the operation of a bulb. Our job is to instruct pupils and earn money at the same time. Mathematically, any task is a transaction and the function y = f (x) denotes it. An input argument x is passed to the function, and the function returns an output argument y. The idea of a function in programming is derived from the aforementioned concept. Task, in my opinion, is on par with function in terms of importance. To say "task" in Hindi is to say "Kriya," which is a verb. As a result, all verbs are considered functions. Humans, for example, are able to move, eat, see, hear, read, and write. It's well known that function may take on several shapes, including the following:

In this case, **y = f (or y = 5), or y=5**, which is a constant function (no argument).

As a single parameter of **f(x), y = 2x+5** is equivalent to

y = f is a multiple argument (a,b,c). In this case, y = ax2+bx2+cx2

To put modular programming into practice, programmers use a wide variety of functions. A job cannot be completed without the presence of function, so an item must have function if it is to carry out its intended function. In other words, each item contains two parts.

1. Data members, Properties, or Features
2. Functions/Methods

It's fascinating that the same functions are found in a certain collection of items. For example, I am able to see, and all the people around me can see. There are some exceptions to this rule. All students and instructors have the same set of abilities and responsibilities (reading, writing, math, science, etc.). Variables, on the other hand, are rare. Students' names, roll numbers, and registration numbers do not all match. Some elements, such as nationality or the name of the university, may be the same regardless of the idea. As a result, objects can be used as placeholders.

X = (Ax, Bx, Cx, Dx,...)

When we say "collection," we mean "variable" or "function."

As (A, B, C, D,.....)

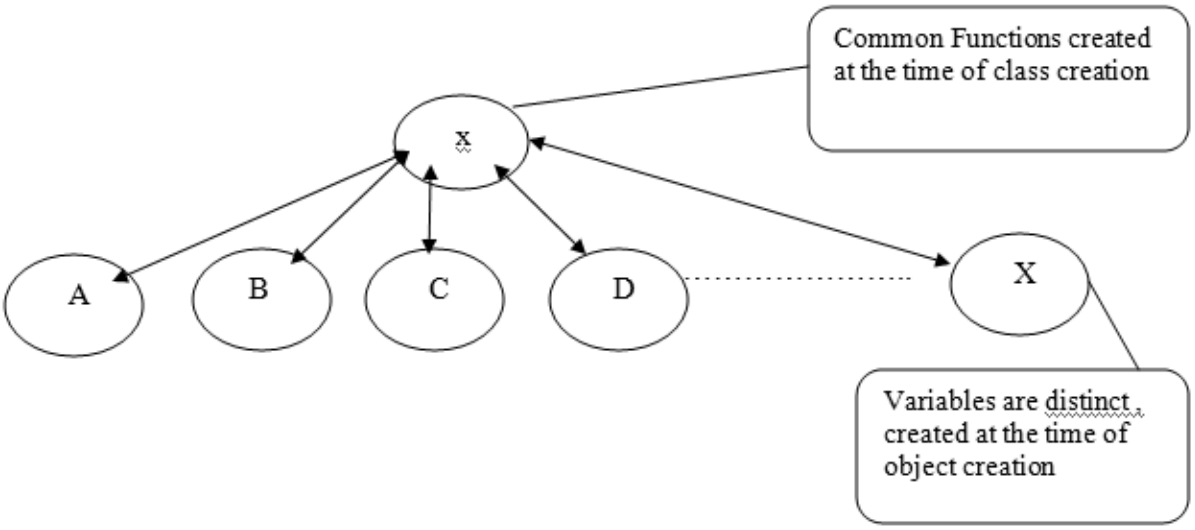

Figure 1: Structure of Set of Objects in Distributed Form

The same arrangement may be found in everyday life. As a result, things might be considered to be the variables and functions that can be recognized uniquely Humans have the same functions or methods but distinct variables or attributes.

I have some functions such as reading, writing, comprehending, eating, and watching. All of those functions existed in the world before I came. All of the people who came before me must have utilized the same functions that I have. When the earliest humans came, they invented functions. Human is the default name (class in programming). That is, when a collection or class is generated, functions are also produced. That, I believe, is why the working region of a specific function is indicated. We are unable to take food utilizing the see function. Eating for function is required. Clearly, when a collection or class is created, functions are produced, and just one set of functions is created for all of its objects. In programming, a set or class is formed during program compilation, along with functions. As a result, the definition of function may be given in the same way as it is in real life.

Variables are not the same for all objects; they differ for each one. When objects are formed, variables are generated as well. For example, once I landed on the planet, variables Names, father's names, residences, phone numbers, email addresses, and other information are produced and saved. The same idea is included into programming Finally, functions are produced when a set (class) is established, and variables are created when an object (indivisual) is created. After functions and variables are coupled to make a complete object, the object is generated. There are no functions.

They are existent within things, however they are only related to objects from outside the objects (as illustrated in the picture). For example, when a teacher strikes a pupil on the ear, the function's linkage is destroyed. After a while, it will be automatically connected. Some serious issues (long-term disconnection) may emerge, but that function can be reconnected with the assistance of a doctor. If we believe that our minds understand function, then mad individuals must grasp everything, but this is not the case.

Object-oriented programming may so be defined as follows:

Object-Oriented Programming is a method of programming in which functions and variables allocate memory at separate times and locations, and then connect them together during compilation or execution to form a complete object that serves as the core component of the program. Object-Oriented Programming is a type of programming in which data members and member functions allocate partitioned memory at different times, and at the time of compilation or execution, a link is established between data member and member function to create a complete object, which is the basic component of programming.

II. SOME FEATURES OF OBJECT ORIENTED PROGRAMMING

1. Encapsulation (binding) of data and abstraction
2. Succession (Uttaradhikar)
3. Genetic variation
4. Destructor and Constructor
5. General programming
6. Multithreading

Every object must have certain functions in order to fulfill a task. That sort of item cannot exist, hence it cannot execute any tasks. In C++, functions that are contained in objects are referred to as "member functions." A set of objects share member functions. So, while creating the object, the member function makes a single copy and allocates memory to it. That is, when the class is declared, a single duplicate of the member function is produced. At the moment of object creation, each data member allocates memory. During compilation or execution, an object might be formed. To generate a full object, a binding procedure is done between data members and member functions. This technique of binding is also known as encapsulation. If the data member allocates their memory statically during compilation, the binding procedure is likewise executed during compilation. Compile-time binding, often known as early binding or static binding, is a sort of binding. If a data member dynamically allocates memory during execution, the binding procedure is likewise executed during execution. This is known as run-time binding, execution-time binding, late binding, or

dynamic binding. Binding results in the creation of a complete object that may be utilized as a single component in Object Oriented Programming.

Because we know that member functions are shared by a group of objects, just one copy of the member function is made and shared by all of them. However, data members are produced separately for each object. This signifies that data members are not shared by all objects. In real life, certain information is very confidential and should not be shared with others, such as an ATM PIN, login, password, and personal information. It is referred to as PRIVATE in programming. Some human society information is shared with others, although some limits apply, such as personal phone numbers, account numbers, work information, and so on. In programming, this sort of data is referred to be PROTECTED. Some knowledge is shared by all members of human civilization. They are known as PUBLIC in programming. Access specifiers or visibility are raised in programming languages to specify common and distinct components. In programming, there are three forms of visibility:

- Private
- Protected
- Public

Because functions are shared by a set of objects, they are usually found in the public area because all of the objects employ the same functions. We believe that God created all sets (classes), hence functions are likewise created by God. That is why, in my opinion, the notion of God is public, as is the concept of just one God. Variables are unique to each object, hence they are often stated in the private or protected section. Private and protected areas are inaccessible from the outside of the object. The private portion cannot be inherited, while the protected and public sections may. ATM pins, usernames, and passwords are not transferable since they are private. Phone numbers, email addresses, and other personal information may be shared with others since they are secure.

In general, we know that a set is a collection of items that share the same attributes and duties. In programming, this is referred to as a class. A class or set is the conventional name for a collection of things. A class or set is frequently referred to as a hypothetical design or blueprint of an item. A class is sometimes referred to as the logical design or blueprint of things. Finally, we can conclude that a class is a user-defined type that is used to construct a new type based on the user's needs by mixing several standard variables and their related functions. A class may also be defined as a user-defined type since it allows the user to construct a new type by encapsulating variables and their related functions according to the user's needs by utilizing suitable visibility such as private, protected, and public. Variables cannot allocate memory when a class is defined, hence they cannot be initialized at the moment of class declaration, just as values cannot be saved when a class is designed. Because functions are shared by a group of objects, the function allocates memory for them at the time of class declaration, allowing the user to populate data members in the functions at the time of declaration. Functions can be defined in a class or set because their memory is allocated at the time the class or set is designed. It is a method of building a bond between a parent and a kid.

We know that UTTARADIKAR is exclusively utilized in parent-child interactions. In this case, parents are a set or class, and children are likewise a set or class. So inheritance is a

mechanism that allows us to construct a new class from an existing one. The current class is referred to as the parent class. The new class is known as the "generic child class." The primary goal of inheritance is to enable the reuse of existing code. Parent class functions and variables are propagated to the child class to take new values in human civilization. We can put the principle into action by using inheritance.

The following are the primary benefits of inheritance:

1. Existing code may be reused in other classes by using inheritance. That is, code re-use may be accomplished via inheritance since parent attributes are stable and liabilities are automatically inherited and reused by the child.
2. Inheritance boosts programmer efficiency since prefabricated elements are passed down from parent to kid.
3. Because many of the parent's functions and features are inherited, product and maintenance costs can be decreased, enhancing product acceptance.
4. Inheritance also improves code security by adding an extra layer of protection around the component at each level of inheritance.
5. Code repetition may be eliminated.

Parent child relationships

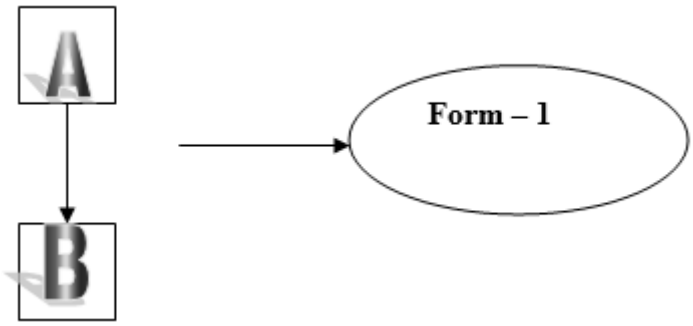

Figure 2: Parent-Child Relationship – Form – 1

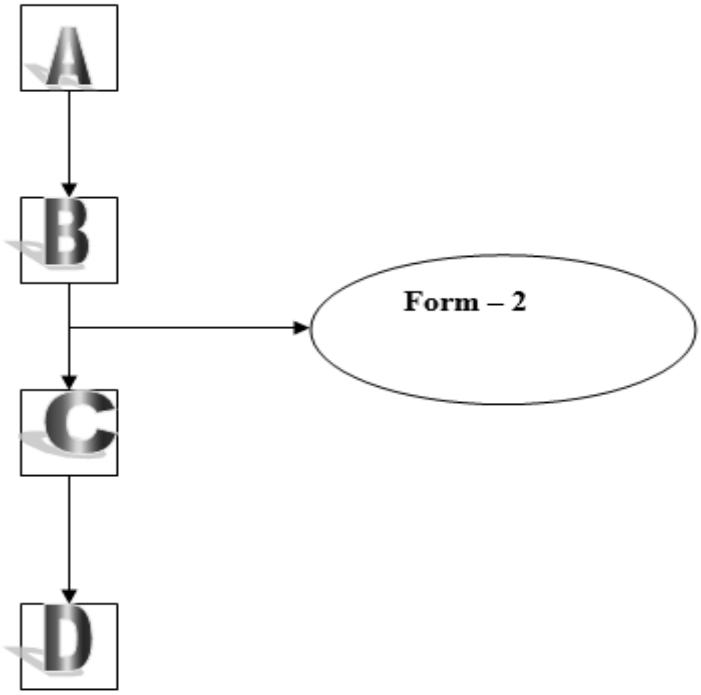

Figure 3: Parent-Child Relationship – Form – 2

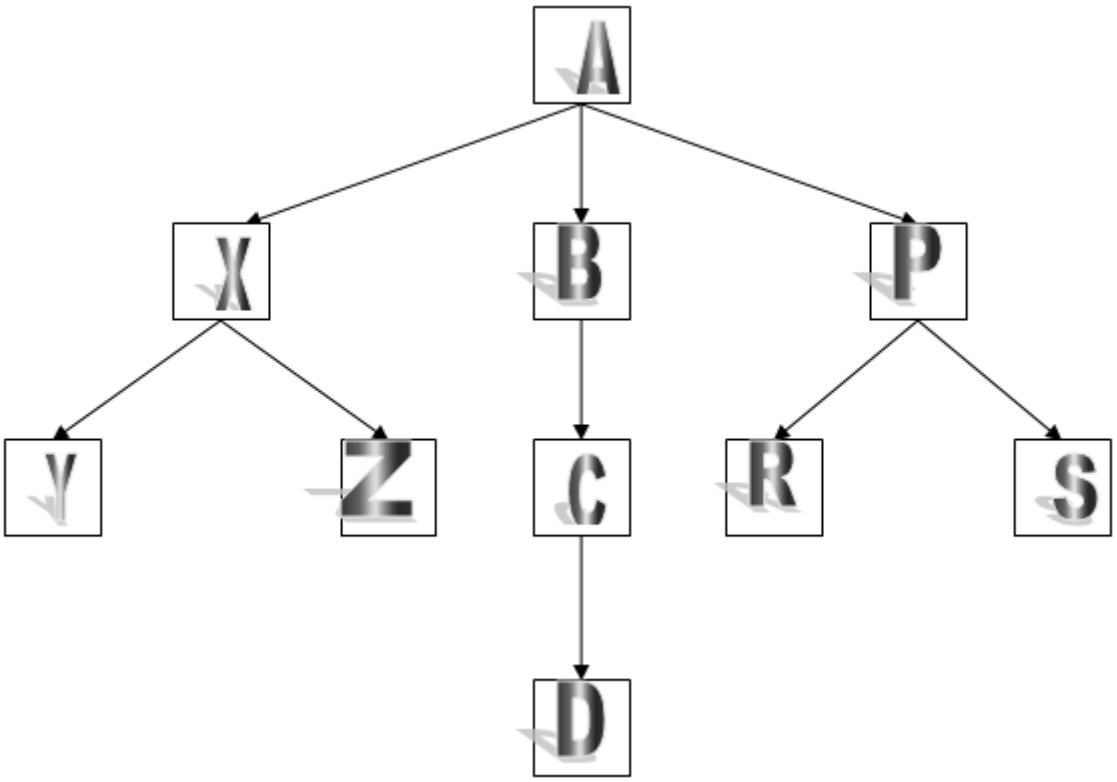

Figure 4: Parent-Child Relationship – Form – 3

In real life, everything has a distinct maker or inventor, such as Samsung, Sony, Voltas, Himalaya, and so on. Objects cannot be generated in the absence of a creator. When the creator is not there, we say God is the creator. That implies that every object must have a creator. The constructor is the object creator in programming. Constructors are special member functions that have the same name as the class and are automatically executed when an object is created. The constructor should be defined in the class's public section. In the private and protected sections, constructor can be declared. As in human civilization, constructors have no return type, not even null or void. Constructor can or cannot accept argument. Constructors can get overburdened. That is, like in human society, several constructors might exist in a single class. In our society, several creators exist for each class. There are several creators in the human class, including Durga, Mahadev, Kalimaa, God, Allah, and others. Constructor, like the actual world, cannot be inherited into the child class. In the actual world, one God cannot be inherited by another. When an object's scope expires, nature automatically eliminates it from the real world; this work is carried out by YAMRAJ. It does not accept arguments and does not return a value. As a result, it cannot be overloaded. In programming, destructor is used to implement this idea. Destructor is a special member function that has the same name as the class and is automatically executed once the object's scope expires. The destructor must be defined in the class's public section; it is not permissible to declare the destructor in the class's private or protected sections. Destructor does not have a return type, not even null or void. Because destructor takes no arguments, it cannot be overloaded. Before the actual name of the destructor, a tilde () symbol must be present. The primary function of the destructor is to free the memory locations held by the object. Destructor operates in the inverse direction of constructor. That is, the destructor releases the memory locations held by the object in the same sequence that the objects were created. For example, a building is built from the ground up and dismantled from the ground up. In real life, one thing can take several shapes, such as parent, brother, kid, husband, customer, and so on. Similarly, one function takes several manifestations, such as singing, quarreling, sobbing, and so on. This is referred to as polymorphism. Polymorphism refers to the same item in several forms. Polymorphism is a programming language notion in which a specific function or operator can assume numerous forms utilizing different types of arguments or different number of arguments and different types of operands and accomplish different tasks as a result. Function overloading refers to a programming system in which a function can take numerous forms and accomplish distinct functions. Similarly, if a certain operator may execute multiple jobs on different sorts of operands in addition to the operator's initial duty, this type of programming method is known as operator overloading. In a word, polymorphism is a programming concept in which a single function may execute many tasks by utilizing the concepts of function overloading and operator overloading. In other words, polymorphism is a programming notion that is achieved by function overloading and operator overloading. In actual life, every object is overloaded, as are the member functions of every object. That is, in real life, a particular item may adopt multiple forms depending on its position, and functions included in the object are likewise overloaded and can execute different tasks using a single function depending on its shape.

The idea of function overloading is used to achieve polymorphism. That is, depending on the quantity and type of parameters, a single function can execute many jobs. As a consequence, memorizing the function name is redundant; just the parameter types and number should be kept. The notion of operator overloading is also used to build polymorphism. That is, in addition to the operator's initial purpose, a single operator can

execute different actions on distinct sorts of operands. That is, a single operator may conduct any sort of operation that is ordinarily not conceivable. The idea of generic programming has the potential to broaden the scope of the operators' job. Operator overloading is a programming method in which a single operator may execute many sorts of operations on various types of operands. The overloaded operator is the operator used to implement operator overloading.

The + operator, for example, may execute addition operations on any sort of numeric operand. Strings, objects, structures, type variables, and any other form of variable may be introduced utilizing the idea of operator overloading. This suggests that an operator may be overburdened. Although the idea of operator overloading exists in C++, it does not exist in Java. Except for the following, practically all operators in C++ may be overloaded:

1. Scope Resolution Operator ::
2. Sizeof() Operator
3. Conditional/TurneryOperator?
4. Class Member Access Operator ->

In practice, one logic is utilized to handle several sorts of objects. A single logic, for example, is used to handle any form of cold drink. Similarly, the same holds true with hot beverages. In a word, a single universal logic may handle many things regardless of their names. This principle is achieved in programming by utilizing generic programming or templates. Generic programming is a programming technique in which type-independent code for all sorts of operands using the same method may be generated. That is, generic programming is a programming idea that may generate generalized code that can be run based on the type specified by the user. Using this notion, a user may develop type-independent generic functions that can be run on any type that has the same logic or algorithm. This programming idea is known as function templates, and the function is known as a generic function. For example, the swap function, which interchanges two numeric values, differs from other numeric types solely in the variable type. A user can develop a single code for all numeric kinds to swap two numeric values using function templates. This is known as a templated function, and the programming idea is known as function template. A single class can also be generalized to include all kinds. Using the same technique, this retains the same function and distinct types. This programming idea is known as a class template, and the class is known as a templated class. When a class is a templated class, all of the class's functions are likewise templated functions. "Meng Lee" and "Alexandor Stepanov" pioneered this approach.

Function template example

```
Template<class T>
Void swap (T & x,T & y)
{
T t;
T=x;
X=y;
Y=t;
}
```


Example Class template

```
Template<class m>
Class array
{
Private:
m  a[100];
Int n;
Public:
  Void readarray ();
  Void printarray ();
};

Template<class m>
Void array<m>:: readarray()
{
 Cout<<"\nHow many no";
Cin>>n;
For (int i=0;i<n;i++)
{
 Cout<<"\nenter  a no";
Cin>>a[i];
}
}

Template<class m>
Void array<m>:: printarray()
{
 Cout<<"\ncontent of array\n";
For (int i=0;i<n;i++)
{
 Cout<<setw(4)<<a[i];
}
}
```

III. CONCLUSION

In real life, we can accomplish many tasks concurrently rather than concurrently to achieve multitasking. For example, when we eat a meal, we can typically eat more than one thing in a set amount of time in cyclic sequence. This is an instance of multitasking. Multithreading is used to accomplish this notion. Multithreading is a programming technique that is used to accomplish the multitasking idea. Thread refers to a little program that may be performed at the same time. To achieve multitasking, more than one tiny application is performed simultaneously or concurrently over a short period of time utilizing round robin scheduling. Because we know that JVM is an operating system that runs on top of the physical operating system, it contains a scheduler that is in charge of implementing multithreading.

We can see from the above explanation that all programming principles (object oriented or structured) are taken from real-life human civilization. So, if we closely watch our culture, we may easily grasp the notion of programming. As a result, we must closely examine our surroundings and understand additional logistical and programming ideas. Some notions in programming are also taken from real-life human experiences.

REFERENCES

[1] Beltramelli, T. (2018, June). Pix2code: Generating code from a graphical user interface Screenshot. In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (p.3).ACM.
[2] Parnianifard, A., Azfanizam, A., Ariffin, M. K. A.M., & Ismail, M. I. S. (2017). Anoverview on Robust design hybrid metamodeling: Advanced Methodology in process optimization under Uncertainty.
[3] Mongkhonvanit, K., Zau, C. J. Y., Proctor, C., & Blikstein, P. (2018, June). Testudinata: a Tangible interface for exploring functional Programming. In Proceedings of the 17th ACM Conference on Interaction Design and Children (pp. 493-496). ACM.
[4] Daniel, B. K. (2018). Reimaging Research Methodology as Data Science. Big Data and Cognitive Computing, 2(1), 4.
[5] Berger, E. D., Hollenbeck, C., Maj, P., Vitek, O., & Vitek, J. (2019). On the Impact of Programming Languages on Code Quality. arXiv preprint arXiv:1901.10220.
[6] Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., ... & Ko, A. J. (2019). A Theory of instruction for introductory Programming skills. Computer Science Education, 1-49.
[7] Colapinto, C., Jayaraman, R., & Marsiglio, S. (2017). Multi-criteria decision analysis with Goal programming in engineering, management And social sciences: a state-of-the art review. Annals of Operations Research, 251(1-2), 7-40.
[8] Wyrich, M., Graziotin, D., & Wagner, S. (2019). A theory on individual characteristics of Successful coding challenge solvers. Peer J Computer Science, 5, e173.
[9] Ketschau, T. J., & Kleinhans, J. (2019). Concept And Implementation of a Two-Stage Coding Scheme for the Development of Computer-Based Testing (CBT)-Items in Traditional Test Software. J, 2(1), 41-49.
[10] Samara, G. (2017). A Practical Approach for Detecting Logical Error in Object Oriented Environment. arXiv preprint arXiv:1712.04189.
[11] Deulkar, K., Kapoor, J., Gaud, P., & Gala, H. (2016). A novel approach to error detection and Correction of c programs using machine Learning and data mining. International Journal On Cybernetics & Informatics, 5(2), 31-39.
[12] Lee, J., Song, D., So, S., & Oh, H. (2018). Automatic diagnosis and correction of logical Errors for functional programming assignments. Proceedings of the ACM on Programming Languages, 2(OOPSLA), 158.

