**ITELIMA FAITH IBIFUBARA**

**MAT. NO.: 18/SCI01/106**

**CSC 302**

1. I). -Unstructured programming languages: Unstructured programming is a procedural program – the statements are executed in sequence as written(The statements execute in order you write). In this you have to use goto statements that allow the control to pass. When a goto statement is executed, the sequence continues from the target of the goto. Thus to understand how a program works, you have to pretend to execute it. This means that it is often difficult to understand the logic of such a program. An example of unstructured programming language is JOSS, an acronym for JOHNNIAC Open Shop System, It pioneered a number of features that would become common in languages from the 1960s into the 1980s, including line numbers as both editing instructions and targets for branches, statements predicated by Boolean decisions, and a built-in editor that can perform instructions in "direct" or "immediate" mode, a conversational user interface.

-Structured programming languages: It is a programming method which aimed at improving quality, clarity and access time of computer program by the use of block structures, subroutines, for and while loops. This programming features will be helpful when concept of exception handing is needed in the program. It uses various control structures, sub routines, blocks and theorem. The theorems involved in structure programming are Sequence, Selection, Iteration and Recursion. An example of structured programming language is C. How C works as a structured programming language: In order to accomplish any task, C-language divide the problem into smaller modules called functions or procedure each of which handles a particular job. That is why C-language is also called as the structured programming language. The program which solves the entire problem is a collection of such functions.

-Modular programming language: Modular programming is the process of subdividing a computer program into separate sub-programs. A module is a separate software component. It can often be used in a variety of applications and functions with other components of the system. Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality. One of the most important concepts of programming is the ability to group some lines of code into a unit that can be included in our program. An example of modular programming language is Ada. It has built in support, extremely strong typing, explicit concurrency, tasks, synchronous message passing, protected objects and non-determinism. Ada improves code safety and maintainability by using the compiler to find errors in favor of runtime errors.

-Object Oriented Programming: Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects. It is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. An example of OOP is Java. The way Java works as an OOP: OOP concepts in Java are the main ideas behind Java’s Object Oriented Programming. They are an abstraction, encapsulation, inheritance, and polymorphism. Grasping them is key to understanding how Java works. Basically, Java OOP concepts let us create working methods and variables, then re-use all or part of them without compromising security. OOP, concepts in Java work by letting programmers create components that can be re-used in different ways, but still maintain security.

-Aspect Oriented Programming: Aspect-Oriented Programming (AOP) is a programming paradigm which complements Object-Oriented Programming (OOP) by separating concerns of a software application to improve modularization. The separation of concerns (SoC) aims for making a software easier to maintain by grouping features and behavior into manageable parts which all have a specific purpose and business to take care of. AOP enables you to move the security (or any other) aspect into its own package and leave the other objects with clear responsibilities, probably not implementing any security themselves.

-Activity oriented programming:

-Event oriented programming: Although similar in some ways to the idea of concurrency, event-driven programming is a strategy for designing the control flow of a program which must react to events that are external to the program itself. Such programs may use threads, routines, continuations, generators, and other facilities to accomplish the goal of processing external events. These events are monitored by code known as an event listener. If it detects that an assigned event has occurred, it runs an event handler (a callback function or method that's triggered when the event occurs). In theory, all programming languages support the event-driven style of programming, although some language features, such as closures, make it easier to implement. Other programming environments, such as Adobe Flash, are specifically tailored for triggering program code by events. An example of this id Visual Basic.NET

II). a)Scientific domain: FORTRAN

b)Business domain: COBOL

c)Artificial intelligence domain: LISP, Prolog

d)General purpose domain: Pascal, Delphi, Java

e)Web programming domain: Java, C#, PHP

f)Mobile programming domain: Java(micro edition), C#

g)Embedded devices domain: Java

III)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S/N | PROGRAMMING LANGAYAGE | DATE | AUTHOR | CLASSIFICATION | PARADIGM | DOMAIN |
| 1. | Java | 1995 | James Gosling | Compiled | OOP | Web programming, General purpose, Mobile Programming, Embedded devices. |
| 2. | C | 1972 | Brian Kernighan, Dennis Ritchie | Compiled | Structured | Embedded devices, Web programming |
| 3. | Visual Basic.NET | 2001 | Microsoft | Compiled | Event Oriented | Web programming |
| 4. | JOSS | 1963 | Cliff Shaw, RAND |  | Unstructured |  |
| 5. | Python | 1991 | Guido Van Rossum | Interpreted | OOP,Structured, Modular programming | Web programming, Artificial Intelligence |
| 6. | Ada | 1980 | Jean David Ichbiah, S. Tucker Taft | Compiled | Structured, Modular programming | General Use |

1. **Evolution of programming languages in a chronological order:**

**(1840s)**

**1840 – Analytical Engine Code**  
The Analytical Engine was a theoretical (i.e., never built) mechanical general-purpose computer, created by British mathematician Charles Babbage. Ada Lovelace came across the idea, and created some code for the Analytical Engine. That’s why she’s considered the first programmer ever.

**(1940s)**

**1943 – ENIAC Coding System**  
The ENIAC is regarded as the first electronic general-purpose computer. Both the computer and its coding were created by John von Neumann, John Mauchly, and J. Presper Eckert.

**1949 – Brief Code (Later Short Code)**  
Initially proposed by John Mauchly, it was one of the first attempts of an assembly language.

**(1950s)**

**1952 - Autocode**

This family of “simplified coding systems” was created in the 1950s specifically for use with the digital computers at the universities of Manchester, Cambridge and London. Considered by many to be the first complied programming language ever invented, Autocode was developed by Alick Glennie to be both comprehensible and high-level.

**1957 – Fortran**  
One of the most popular high-level programming languages. It was created by John W. Backus at IBM as an easier alternative to programming in assembly. Fortran is a general-purpose, imperative programming language suited to numeric computation and scientific computing. In use for over half a century, Fortran was developed by IBM in 1957 for both scientific and engineering applications.

**1958 – LISP**  
Created by John McCarthy, one of the pioneers of AI as well.

**1959 – COBOL**  
The name stands for Common Business-Oriented Language, as the language was aimed mainly at banks, financial institutions and companies.

**(1960s)**

**1964 – BASIC**

Beginner’s All-purpose Symbolic Instruction Code, a family of general-purpose, high-level programming languages whose design philosophy emphasizes ease of use.

**1968 - ALGOL68**

Short for Algorithmic Language 1968, Algol 68 was an imperative programming language designed as a successor to Algol 60. With a wider scope of application and rigorously defined syntax, this language was the first to be fully defined before it was implemented.

**(1970s)**

**1970 – Pascal**  
Pascal is an influential imperative and procedural programming language, designed in 1968–1969 and published in 1970 by Niklaus Wirth as a small and efficient language intended to encourage good programming practices using structured programming and data structuring. Pascal enabled programmers to define their own complex datatypes and made it easier to build dynamic and recursive data structures like lists, trees and graphs.

**1972 – Smalltalk**  
The language that started to inflate the popularity of object-oriented programming.

**1972 – C**  
Created by Dennis Ritchie and Ken Thompson at the AT&T Bell Labs. It’s simplicity and efficiency made it one of the most popular languages around the world. C program source text is free-format, using the semicolon as a statement terminator and curly braces for grouping blocks of statements.

**1972 – SQL**  
Created at IBM, it became the standard for dealing with databases.

**(1980s)**

**1983 – C++**  
Originally named “C With Classes”, it brought object-orientation to C (which is technically a subset of C++). This programming language was designed mainly for system programming but has expanded to be used in desktop, servers and performance-critical applications. It inherited most of C’s syntax and has imperative, object-oriented and generic programming features.

**1987 – Perl**  
Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages. It borrows features from other programming languages, such as C, AWK and sed. Originally, the only documentation for Perl was a single manual page, but it has gone through several revisions and changes.

**(1990s)**

**1991 – Python**  
A high-level language that emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C, C++ or Java. Python’s design philosophy focuses on readability. Its a successor to the ABC language.

**1995 – Java**  
Java is the most popular object-oriented programming language around, and it was created to have as few implementation dependencies as possible. It’s widely used in commercial and business applications. It is intended to let application developers “write once, run anywhere,” so that Java can run on any platform that supports it without the need to recompile.

**1995 – PHP**  
PHP is a server-side scripting language used for both Web development and general-purpose programming. This language can be mixed with HTML code or used in combination with templating engines and Web frameworks. It was originally not meant to be a programming language, but grew organically over time.

**(2000s)**

**2001 – C#**  
This multi-paradigm programming language was developed by Microsoft within the .NET framework. It was intended to be simple, modern and object-oriented. The most recent version of C# was released in 2012.

**2001 – Visual Basic.NET**  
A successor to the original Visual Basic language, Visual Basic .NET is a high-level programming language implemented on the .NET framework. It uses statements to specify actions and is one of the two main languages targeting the .NET framework, along with Visual C#.

**(2010s - Present)**

**2014 –Swift**  
Swift was created by Apple for iOS and OS C development. It was introduced in 2014 at Apple’s Worldwide Developers Conference. Designed to work with Apple’s Cocoa and Cocoa Touch frameworks, Swift is meant to be more concise and resilient to erroneous code.

1. An object-oriented program usually contains different types of objects, each corresponding to a particular kind of complex real world object or complex data to manage, or perhaps to a real-world object or concept such as a bank account, a hockey player, a bank customer, a departmental store or a bulldozer. Object-oriented programming implies that your program is, well, oriented towards objects. It says nothing about modules within your application but demands that logical pieces that represent some ideas within the application are modeled via classes and objects.

Modular programming (also called "top-down design" and "stepwise refinement")paradigm is a software design technique that emphasizes separating the functionality of a program into independent, meaningful interchangeable modules, such that each module contains everything necessary to execute only one aspect of the desired functionality. **Modular programming** just implies you have two (or more) modules, but says nothing of how they achieve what they achieve. The modules can use object-oriented approaches or not at all and use procedural C-style programming.

Here is an example:

Consider you have a program. A few input fields and a button. Then some calculations are made and the program outputs something.

This program can have 2 modules: The input/output one and the calculation one. The modules can be implemented as classes, that is indeed correct. However, modules are meant to be logically separate pieces of the programs and as such it doesn't make sense to have them as classes, as you can have many different objects of a class. In the example, object-oriented programming there would be classes defining the input fields and buttons, or maybe a class that is used as a calculator. We could even go to greater depths and define a Calculator interface that could be implemented as SumCalculator, ProductCalculator etc., and maybe even throw in some factories so the user can choose between different calculations performed by your program. Yes, there could be singleton classes such as LayoutModule (which would keep track of objects of InputField and Button type) and LogicModule (which would keep track of the Calculator implementations).