**SUBAIR CALEB**

**17/SCI01/078**

**1i)**

**Unstructured Programming Languages.**

These are the historically earliest programming paradigm. Codes are written as a single whole block and the whole program is taken as a single unit. They have a limited number of data types like numbers, arrays, strings.

***Examples of Unstructured Programming*** Languages.

* earlier versions of BASIC (such as MSX BASIC and GW-BASIC)
* early versions of COBOL
* early versions of FORTRAN
* early assembler systems language (without procedural metaoperators)
* some scripting languages such as MS-DOS batch file language
* JOSS
* Machine Language
* FOCAL
* TELECOMP
* MUMPS

**Structured Programming Languages**

Structured programming is a procedural programming subset that reduces the need for “go-to” statements. It facilitates program understanding and modification and has a top-down design approach, where a system is divided into compositional subsystems.

**Examples include;**

* Delphi
* PL/I
* C++
* ADA
* PASCAL
* ALGOL
* C
* C++
* Java
* Alma-0
* QBasic
* Perl
* BLISS

**MODULAR PROGRAMMING LANGUAGES.**

Here, functions of a program are separated into self-contained components called modules which allow for edition and modification without the entire project being affected.

**Example include;**

* MATLAB
* Haskell
* C (some specific versions)
* Java
* Erlang
* RPG (IBM)
* IBM/360 Assembler
* Ruby
* Perl
* Modula

**OBJECT ORIENTED PROGRAMMING LANGUAGES**

These are different because in object oriented programs, a module can be called from a different program. It is based on the concept of "objects", which can contain data, in the form of attributes, and code, in the form of procedures (or methods).

**Examples include;**

* Smalltalk
* C#
* C++
* Java
* Python
* Objective C
* Ruby
* Scala
* Emerald
* Eiffel
* COBOL 2002
* ABAP

**ASPECT ORIENTED PROGRAMMING LANGUAGES**

The main aim of aspect oriented programming is to increase modularity

**Examples include;**

* Aspectj
* SPRING AOP
* Java (via Aspectwerkz)
* Aspect C++
* AspectAda (extension for Ada)
* A04BPEL (extension in BPEL)
* A04AADL (extension in AADL)
* EOS (extension for C#)

**ACTIVITY ORIENTED PROGRAMMING LANGUAGES**

They did not stand the test of time.

**EVENT ORIENTED PROGRAMMING LANGUAGES**

This is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs or threads.

**Examples include;**

* Visual Basic
* Visual C++
* Java
* JavaScript
* ActionScript
* Elm
* Delphi
* KanabosFramewoek
* Python
* Scratch

**1ii)** There are 7 domain requirements for programming languages. They are listed below with examples below;

* Scientific Domain: FORTRAN, PL/1, MATLAB, ALGOL, APL, Julia, Java
* Business Domain: COBOL , PL/B, Java, C++, Python, Objective C, ABAP
* Artificial intelligence Domain: Lisp, Prolog, Java, Python, C++ (2011 version and higher), R, Smalltalk, Haskell, Perl.
* General Purpose Domain: C++, Python, C#, java, JavaScript, Ada, ALGOL, BASIC, C, Python, Ruby, Visual basic, Dart, Erlang, PHP, RPG, Pascal, Scala, COBOL, F#.
* Web Programming Domain: PHP, Java, Python, CSS, C++, C, Ruby, JavaScript, Erlang, Haskell.
* Mobile Programming Domain: Java, C#, Python, PHP, C++ Objective C, Scala, Ruby, Perl, Actionscript.
* Embedded Devices Domain: Java, C, Python, Ada, JavaScript, C#, Embedded C++.

**1iii)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Scientific | Business | Artificial intelligence | General purpose | Web programming | Mobile Programming | Embedded Devices |
| Unstructured | FORTRAN | COBOL |  | BASIC |  |  |  |
| Structured | ALGOL | C++ | Java | C | C | Perl | C |
| Modular | Java | Java | Java | RPG | Haskell | Perl | Java |
| Object oriented | Java | ABAP | Smalltalk | Ruby  | Ruby | Python | Python |
| Aspect oriented | Java | Java | Java | Ada(aspectada) | C++ | C#(via EOS) | C#(via EOS) |
| Event oriented | Java | C++ | Java | Visual basic | Python | Python | Java script |

**2)**

|  |  |  |
| --- | --- | --- |
| YEAR | LANGUAGE | Author(s) |
| 1951 | Regional assembly language | Stan Poley |
| 1954 | IPL (Information Processing Language) | Allen Newell,Cliff Shaw, Herbert A. Simon |
| 1955 | FLOW-MATIC | Remington Rand, Grace Hopper |
| 1957 | FORTRAN | John Backus |
| 1957 | COMTRAN | Bob Bemer |
| 1958 | Lisp | Steve Rusell, Timothy P Hart, Mike Levin. |
| 1958 | ALGOL 58 | Friedrich L. Bauer, Hermann Bottenbruch, Heinz Rutishauser, Klaus Samelson, John Backus, Charles Katz, Alan Perlis, Joseph Henry Wegstein |
| 1959 | FACT | Computer Sciences Corporation, Honeywell |
| 1959 | COBOL | ODASYL, ANSI, ISO |
| 1959 | RPG | IBM |
| 1962 | APL | Larry Breed, Dick Lathwell, Roger Moore |
| 1962 | Simula | Kristen Nygaard |
| 1962 | SNOBOL | David J. Farber, Ralph E. Griswold, Ivan P. Polonsky, and Bell Labs |
| 1963 | CPL | Christopher Strachey et al. |
| 1964 | Speakeasy | Speakeasy Computing Corporation |
| 1964 | BASIC | John G. KemenyThomas E. Kurtz |
| 1964 | PL/1 | IBM |
| 1966 | JOSS | Cliff Shaw, RAND |
| 1966 | MUMPS | Neil Pappalardo |
| 1967 | BCPL | Martin Richards |
| 1968 | Logo | Bolt, Beranek and Newman |
| 1969 | B | Ken Thompson, Dennis Ritchie |
| 1970 | Pascal | Niklaus Wirth |
| 1970 | Forth | Charles H. Moore |
| 1972 | C | Dennis Ritchie & Bell Labs |
| 1972 | Smalltalk | Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Diana Merry, Scott Wallace, Peter Deutsch and Xerox PARC |
| 1972 | Prolog | Alain Colmerauer, Robert Kowalski |
| 1973 | ML | Robin Milner and others at the University of Edinburgh |
| 1975 | Scheme | Guy L. Steele and Gerald Jay Sussman |
| 1978 | SQL | ISO/IEC |
| 1980 | C++ | Bjarne Stroustrup |
| 1983 | Ada |  Ada 83: Jean Ichbiah |
| 1984  | Common Lisp (CL) | Scott Fahlman, Richard P. Gabriel, David A. Moon, Kent Pitman, Guy Steele, Dan Weinreb |
| 1984 | MATLAB | Cleve Moler, MathWorks |
| 1986 | Eiffel | Bertrand MeyerDeveloperEiffel Software |
| 1986 | Objective C | Tom Love and Brad Cox |
| 1986 | LabVIEW | National Instruments |
| 1986 | Erlang | Joe Armstrong |
| 1987 | Perl | Larry Wall |
| 1988 | Tcl | John Ousterhout |
| 1988 | Wolfram | Stephen Wolfram |
| 1989 | FL | John BackusJohn WilliamsEdward Wimmers |
| 1990 | Haskell | Lennart Augustsson, Dave Barton, Brian Boutel, Warren Burton and others |
| 1991 | Python | Guido van Rossum |
| 1991 | Visual basic | Alan Cooper |
| 1993 | Lua | Roberto IerusalimschyWaldemar CelesLuiz Henrique de Figueiredo |
| 1993 | R | Ross Ihaka and Robert Gentleman |
| 1995 | Ruby | Yukihiro Matsumoto |
| 1995 | Ada 95 | Tucker Taft |
| 1995 | Java | James Gosling |
| 1995 | Delphi | Anders Hejlsberg |
| 1995 | JavaScript | Brendan Eich |
| 1995 | PHP | Rasmus Lerdorf |
| 1997 | Rebol | Carl Sassenrath |
| 2000 | Actionscript | Gary Grossman |
| 2001 | C# | Microsoft |
| 2001 | D | Walter Bright, Andrei Alexandrescu |
| 2002 | Scratch | Mitchel Resnick, Brian Silverman and Paula Bonta |
| 2003 | Groovy | James Strachan |
| 2003 | Scala | Martin Odersky |
| 2005 | F# |  |
| 2006 | Powershell | Jeffrey Snover |
| 2007 | Clojure | Rich Hickey |
| 2009 | Go | Ken Thompson, Robert Griesemer |
| 2010 | Rust | Graydon Hoare |
| 2011 | Dart | Lars Bak and Kasper Lund |
| 2011 | Kotlin | JetBrains |
| 2011 | Elixir | José Valim |
| 2011 | Red | Nenad Rakocevic |
| 2012 | Julia | Jeff Bezanson |
| 2012 | Typescript | Anders Hejlsberg |
| 2014 | Swift | Chris Lattner |
| 2014 | Crystal | Ary Borenszweig, Juan Wajnerman, Brian Cardiff and more |
| 2014 | Hack | Facebook |
| 2016 | Reason | Jordan Walke |
| 2017 | Ballerina | Sanjiva Weerawarana, James Clark, Sameera Jayasoma, Hasitha Aravinda, Srinath Perera, Frank Leymann and WSO2 |

3) Modular programming is a coding method that entails the use of routines, sub routines and functions in order to organize and execute specific sets of instructions pertaining to a given task to be done.

WHILE

Object Oriented Programming is a coding method that entails the use of objects and their relationships in order to describe, programmatically, the problem to be solved.