NASIR FIRDAUS OPEYEMI

17/SCI01/051

CSC302

1. (I) Examples of Structured Programming language are C, C+, C++, C#, Java, PERL, Ruby, PHP, ALGOL, Pascal, PL/I and Ada

The structured programming language allows a programmer to code a program by dividing the whole program into smaller units or modules. Structured programming is not suitable for the development of large programs and does not allow reusability of any set of codes.

(ii)A typical example of non-structured if BASIC. other **unstructured** include JOSS, FOCAL, TELCOMP, assembly **languages**, MS-DOS batch files, and early versions of BASIC, Fortran, COBOL, and MUMPS.
Unstructured programming is a procedural program – the statements are executed in sequence as written(The statements execute in order you write)

(iii)Example of procedural programming language are; FORTRAN, C, PASCAL, e.t.c

Procedural programming is a **programming** paradigm, derived from structured **programming**, based on the concept of the **procedure** call. Procedures, also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out.

(iv)Modular object oriented: Languages that formally support the module concept include Ada, Algol, BlitzMax, C#, Clojure, COBOL, D, Dart,
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.

(V)Example of Aspect oriented is Haskell Language.
Aspect-Oriented Programming (AOP) is a programming paradigm which complements Object-Oriented Programming (OOP) by separating concerns of a software application to improve modularization.

(Vi)Example of Activity Oriented:
Activity-oriented knowledge representation methods. The main goal of **activity-oriented** methods is not to describe the contents of a topic in its. entirety. Rather, a network of labelled nodes and links is used to structure the knowledge in. ways that support different

(Vii)Example of Event Oriented is C++
event-driven programming 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.

2. • EVOLUTION OF PROGRAMMING LANGUAGES, 1940's machine level:

– use binary or equivalent notations for actual numeric values

EVOLUTION OF PROGRAMMING LANGUAGES

1950's: "assembly language"

1. – names for instructions: ADD instead of 0110101, etc.
2. – names for locations: assembler keeps track of where things are in memory; translates this more humane language into machine language
3. – this is the level used in the "toy" machine
4. – needs total rewrite if moved to a different kind of CPU

EVOLUTION OF PROGRAMMING LANGUAGES, 1960's

• "high level" languages -- Fortran, Cobol, Basic

– write in a more natural notation, e.g., mathematical formulas

– a program ("compiler", "translator") converts into assembler

– potential disadvantage: lower efficiency in use of machine

– enormous advantages:

accessible to much wider population of users

portable: same program can be translated for different machines

more efficient in programmer time

EVOLUTION OF PROGRAMMING LANGUAGES,1970's

• "system programming" languages -- C

– efficient and expressive enough to take on any programming task

writing assemblers, compilers, operating systems

– a program ("compiler", "translator") converts into assembler

– enormous advantages:

accessible to much wider population of programmers

portable: same program can be translated for different machines

faster, cheaper hardware helps make this happen

EVOLUTION OF PROGRAMMING LANGUAGES,1980's

• "object-oriented" languages: C++

– better control of structure of really large programs

better internal checks, organization, safety
– a program ("compiler", "translator") converts into assembler or C
– enormous advantages:
portable: same program can be translated for different machines
faster, cheaper hardware helps make this happen

EVOLUTION OF PROGRAMMING LANGUAGES, 1990's
• "scripting", Web, component-based, ...:
Java, Perl, Python, Visual Basic, Javascript, ...
– write big programs by combining components already written
– often based on "virtual machine": simulated, like fancier toy computer
– enormous advantages:
portable: same program can be translated for different machines
faster, cheaper hardware helps make this happen.

EVOLUTION OF PROGRAMMING LANGUAGES, 2000's
• so far, more of the same
– more specialized languages for specific application areas
Flash/Action script for animation in web pages
– ongoing refinements / evolution of existing languages
C, C++, Fortran, Cobol all have new standards in last few years
• copycat languages
– Microsoft C# strongly related to Java
– scripting languages similar to Perl, Python, et al
• better tools for creating programs without as much programming
– mixing and matching components from multiple languages

Others include;
* 2000 – ActionScript
* 2001 – C#
* 2001 – D
* 2002 – Scratch
* 2003 – Groovy
* 2003 – Scala

* 2005 – F#
* 2006 – PowerShell
* 2007 – Clojure (
* 2009 – Go
* 2010 – Rust
* 2011 – Dart
* 2011 – Kotlin
* 2011 – Red
* 2011 – Elixir
* 2012 – Julia
* 2012 - TypeScript
* 2014 – Swift
* 2016 – Ring

3. Modular programming and object programming are two safe approaches to the logical organisation of a program, permitting the reusability and the modifiability of software components. Programming with objects in Objective CAML allows parametric polymorphism (parameterized classes) and *inclusion/subtype polymorphism* (sending of messages) thanks to late binding and subtyping, with restrictions due to equality, facilitating incremental programming.

Modular programming allows one to restrict parametric polymorphism and use immediate binding, which can be useful for conserving efficiency of execution.

The modular programming model permits the easy extension of functions on non-extensible recursive data types. If one wishes to add a case in a variant type, it will be necessary to modify a large part of the sources.

The object model of programming defines a set of recursive data types using classes. One interprets a class as a case of the data type.

Also,
An object-oriented program usually contains different types of objects, each corresponding to a particular kind of complex data to manage, or perhaps to a real-world object or concept such as a bank account, a hockey player, or a bulldozer.
While
Modular programming (also called "top-down design" and "stepwise refinement") 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.