Polymorphism

- Is a Greek term, which means the ability to take more than one form. An operation may exhibit different behaviours in different instances. The behaviour depends on the data used in the operation, for example, the operation addition on 2 nos will generate a sum. If the operands are strings, then the operation would produce a string by concatenation.
- Allows different objects having different internal features to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ.
- Extensively used in implementing inheritance.

Dynamic Binding

- Refers to linking of a procedure call to the code to be executed in response to the call. This means that the code associated with a given procedure call is not known until the time of the call at run – time.

- Associated with polymorphism and inheritance

- Function call associated with a polymorphic reference depends on the dynamic type of that reference.

Message Passing

An O-O Program consists of a set of objects that communicate with each other. The process of programming in OOP language involves the ffg. basic steps:

1. Creating classes that defines the objects and their behaviour
2. Creating objects from class definition, and
3. Establishing comm. among objects.

Message passing involves specifying the name of the function or method and the information to be sent. E.g student.Grade(matricNO); lecturer.salary(name);

This concept makes it easier to talk about building systems that directly model or simulate their real-world counterparts.

Benefits of OOP

- Through inheritance, we can eliminate redundant code and extend the use of existing classes.
- We can build programs from standard working modules that communicate with each other, rather than having to start writing the code from scratch. This leads to saving development time and high productivity.
- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by codes in other parts of the program.
- It is easy to partition the work in a project based on objects.
- O-O systems can be easily upgraded from small to large systems
- Message passing techniques for communication between objects makes the interface description with external systems much simpler.
- Software complexity can be easily managed

Areas of OOP Application

Promising areas of OOP application includes Real-Time Systems, Simulation and Modelling, O-O databases, AI, Expert Systems, Neural Networks, Decision Support Systems.

General Principles in Writing and Testing Programs

- **Plan your program**: do not start coding straight away. Write an outline showing the main steps in sequence.
- **Develop in stages**: revise your outline as often as required, breaking down each main step into a sequence of simpler steps. Repeat this process until your steps corresponds to program statements.
- **Define variables**: while developing your programs, think about the main variables and arrays you will require to represent the information. Choose names which suggest their usage and write down each name with its type and dimensions if an array.
- **Modularize:** use functions and subroutines not only to avoid repetitive coding, but more importantly, to keep your program simple and its structure clear by putting the details in separate units.

General Principles in Writing and Testing Programs(2)

- **Provide for exceptions**: design your programs to cope with invalid data by printing informative error messages rather than simply failing e.g. due to attempted division by zero. This is important most especially if your program will be used by others
- **Clarity**: use indentation to clarify its logical structure and include explanatory comments freely.
- **Testing:** try running your program using suitable data once you have eliminated all the syntax errors. Calculate what the result should be and check that the actual results correspond.

SRQ

- “To see it right set it right”. Discuss this assertion with reference to a good programming style.

Introduction to Java PL.

- Is an OOP with a relatively simple grammar.
- It omits rarely used, poorly understood, confusing features of C++ such as header files, pointer arithmetic, structures, unions, operator overloading, and templates.
- There is no global data i.e. All methods, fields and constructors are local to the classes.
- It supports static methods and fields, exception handling, inheritance, control structures such as while loops, for loops and if-else statements.

Fundamental features of JAVA

- **Simplicity**: the designers were trying to develop a language that a programmer could learn quickly. They eliminated some constructs in C and C++ that are complex such as pointers.
- **O-O:** Everything is an object in Java. Focus is on the data and the methods that operate on the data in the application. It doesn’t concentrate on procedures only
- **Platform independent:** it has the ability to move from one computer to the other or from one OS to another without any difficulty.

Fundamental features of JAVA(2)

- **Robust:** it is a strongly typed language**.** Requires explicit declaration and has exception handling features. It doesn't have a pointer or pointer arithmetic, so the programmer doesn't need to worry about memory allocation.
- **Security:** totally secured. It provides a controlled environment for the execution of the program.
- **Distributed:** can be used to develop applications that are portable across multiple platforms, OS and GUI
- **Multi-threaded:** Java programs uses multithreading to perform many tasks simultaneously

Creating and Running Java Programs

There are many windows - based integrated packages for editing and running Java programs((Java NetBeans, JCreator, JBuilder, Oracle JDeveloper, e.t.c) which could be downloaded from the Internet. , But if you do not have access to these packages, you can use the command windows,

Writing your first program

- A simple Java code looks like this:

```
public class Welcome
{
        public static void main(String[] args)
        {
                System.out.println("Welcome to Java Class");
        }//End main method
}//End class Welcome
```

Note: Java is case sensitive. Capital letters must be capital e.g. String, System.

Analyzing the Program

- The 1st line could be regarded as the heading of the program. Java sees a program as a class and the name of the class must be the same as the file name i.e the name you use to save the file.(Welcome.java).

Note: Any non-empty string of letters and digits can be used for the class name as long as it begins with a letter and contains no blanks.

- The 2nd line begins with the left brace character, just like BEGIN in Pascal. There must be a corresponding right brace at the last line of the program(like END in Pascal). This two braces form the program block, which encloses the program's body

Analyzing the Program(2)

- The 3rd line : Every class must have a method or function that will be used to manipulate the data in the class. The default name of that method is **main.**
- The main method has some *descriptors* associated with it (public, static, and void)

  public: means that the contents of the following block(i.e the function or method) are accessible from all other classes.

  static: means that the method being defined applies to the class itself rather than to the objects of the class.

  void: means that the method being defined has no return value.

  main: means this is the name of the method being defined, just as Welcome is the name of the class being defined.

Analyzing the Program(3)

The 4th line behaves like the 2nd line

The 5th line: contains the single executable statement.

```
System.out.println("Welcome to Java Class");
```

The message in quote will be printed out as they are written.

println: is the method that tells the system how to do the printing, which means that the cursor should be moved to the next line after the message is printed.

Note: the parenthesis and the semicolon usage . The ; is a terminator for each executable line.

The two closing braces marks the end of the program. The first closes the main method and the second closes the class.

Inserting comments into your programs

- Comments are very good in programs. They enhances easy comprehension/readability of the codes, section by section.

- Also they are veritable tool for future program maintenance. Anybody can pick the code in the future, and with the help of the comment lines, modify, upgrade or correct the program for some errors.

- Java embraces both C and C++ styles of comments.

Note: Adding comments is called documenting your code and comments are normally ignored during compilation.