AL-HIKMAH UNIVERSITY, ILORIN, NIGERIA

Adeta Road, Adewole Housing Estate, P.M.B. 1601, Ilorin

.....learning for wisdom and morality.....

CENTRE FOR ICT AND DISTANCE LEARNING (IDL)

e-CONTENT DEVELOPMENT (DL) UNIT

e-Note for SEN 102

A. Course Lecturer's Details

Name: OYELAKIN, Akinyemi Moruff

Mobile Number:+2348062738520

Official Email Address: amoyelakin@alhikmah.edu.ng

Personal Email Address:moyelakin80@gmail.com

B. Faculty Department and Programme

Faculty: Natural and Applied Science

Department: Computer Science

Programme: B.Sc. Software Engineering

C. Course Title: Principles of Programming I

D. Course Code: SEN 102

E. Course Credit: 03

F. Course Description

Basic principles of programming, fundamentals of procedural and object-oriented programming using a suitable high-level object-oriented language. Emphasis should be placed on such principle as procedural and data abstraction, encapsulation, code reuse and composition. The programming language to be used in teaching the course is Java.

G. Learning Objectives

At the completion of this e-note, it s expected that learners will achieve the following:

1. Understand the basic concepts of programming principles
2. Acquire the necessary skills on data types, variable declaration and handling of programming problems in Java
3. Exhibit mastery of principles for writing good Java programs
4. Be able to independently write small-size programs using Java programming Language

Before we begin

This course is aimed at exposing the software engineering students to some basic approaches in programming computers. The choice programming language for teaching is an OOP language named Java. Before we proceed, some concepts that will aid the understanding of the students will be introduced.

Computer Programming is the act of writing instructions that computer systems work upon. Programming of a problem is achieved with the use of suitable programming language.

The choice of a programming language for a given problem depends on a number of factors. These factors include the programming needs, the domain of the problem (education, enterprise, hospitality, banking, entertainment, and aviation), the programmer programming skills and much more. OOP as it is popularly called is a programming paradigm that supports development of robust and more powerful modern applications. The development of software using OOP revolves around conceptualizing software solutions and their components as objects.

Note: This aspect of what influence the choice of programming language goes beyond the context of this course.

Levels of Programming Language

There are three levels of programming languages.
They are: Machine Language;
Low Level Language (LLL) or Assembly Language; and
High Level language (HLL).

Examples of High Level Languages are: C++, C. PASCAL, Ada, Delphi, PL/1, C Sharp (C#), FORTRAN, Python, Java, BASIC, Smalltalk, Delphi, and so on. A programmer in whatever level (beginner, intermediate and professional) has to follow the syntax of the chosen language he/she wants to use to develop the targeted applications/softwares.

High Level Languages use language translator to be executed on a computer/machine. The translators used by high Level languages are either interpreter or compiler. In few cases, we have language that use both compiler and interpreter. They are termed Hybrid Software Implementations.

Note:

The high level languages can be classified into Object Oriented Languages, Structured Languages, Imperative and Functional Programming Languages and so on.

Java is the teaching programming language. It is very important for the students to show interest in logic thinking, mastery of basic concepts and an understanding of the approach to develop an effective software/application.

Features of Java as a Programming Language

i. Java is case-sensitive. That is, the programming language differentiates between uppercase letter and lower case letter.
ii. Java is a fully object oriented programming language
iii. It is a language that has support for developing applications in science, business, entertainment and other domains.
iv. Java is robust
v. Java is platform independent.

Computer Programming refers to as the process of writing instructions/codes that are executed on a computer system and allied hardware device.

Programming does not require cramming. Rather, it requires that the learner masters both the syntaxes and semantics of the language. Then, for every programming problem, the programmer is expected to master the program logic/business logic.

If the syntax of a programming language is violated while writing a program, we will have what is called **SYNTAX ERROR. Until the programmer corrects the syntax error(s) in a program, the output will not be generated.**
When you encounter this error in a program, there is no need for alarm. Rather, you are expected to closely look at what is causing the error.

A syntax error can be as a result of misspelling of keywords, omission of comma, wrong use of a keyword and so on.

It is important for every learner of a programming language to learn about the syntaxes of such language.
Then, such a learner is expected to be practicing from time to time.
Do not bite more than you can chew.
Start from simple program to a very difficult one.
A typical Java program has the below syntax:

```
//Comments about what the program does
Import statement (as may be required in a programming problem)
public class OyelakinFirstExample
{
public static void main (String args[])
{
//declaration of variables to be used in a program
Program codes for handling the problem at hand
}
}
```

If a programmer violates any of the syntaxes of a programming language, syntax error occurs.

Examples of syntax errors abound in every programming language. God willing, some of the examples in Java programming will be introduced during lectures.

SECTION TWO

Techniques for achieving good programming skills

Programming is not about cramming. Programming is an art. It is expected that you keep learning and practicing so as to have a solid background in Computer Programming.
To be able to master the art of programming, it is important to point out that students should know some techniques that will aid them.
Programming involves problem solving and it is expected that problem solving skills and tools be acquired. Each time you are given a programming task, you are expected to think deeply, understand the problem and strategise the best approach in solving the programming problem.

This is required so that the proposed solution can be used to attain the desired goal.

From time to time, a would-be good programmer must as well practice a lot.
You are expected to think out of the box, and practice out codes that you want to use to address a particular programming issue.

Moreover, you are expected to master the syntaxes (rules) of a particular programming language prior to using it to build programs.
For instance, to learn Java in this course, one of the steps to start well is to master the syntaxes of Java.

As a beginner while writing programs, you need to be reminded that we make use of System/Software Development Life Cycle (SDLC). The stages in SDLC include:

Problem Definition/Problem Analysis/User Requirement Analysis
Program Design/System Design

Coding
Testing and Debugging
System/Program Implementation
Evaluation and Maintenance

Each time you are given a programming problem (academic or real life situation, you are expected to follow the SDLC process to develop the application.
Constant practice saves you a lot of troubles while programming.

Basic Programming Principles in Programming:

1. The 'KISS' Principle:

This is one of the most important principles that you need to keep in mind if you want to create great and possible practical works. Simply this means "keep it simple, stupid" principle. You need to follow this principle if you are into medium-to-large programming projects of an enterprise.

2. The 'DRY' Principle:

Another important principle to achieve clean and easy code in programming is named DRY. This means "don't repeat yourself" principle. This basically means not to repeat the coding that you have already applied. Therefore when you are writing any code ensure that you do not repeat any code thus avoiding any duplication of any coding.

3. Open/Closed Principle:

Another principle that you need to keep in mind among the Basic Programming Principles is that to keep the code is open to extension but closed to modification. This you need to follow when you are writing code in Java or Modules in the Python. Therefore, it creates a big separation from core behavior to modified behavior

4. Composition Over Inheritance Principle:

In this, the coding of the project with complex behavior should have their own instance of the object rather than depending on inheriting a class as well as adding new behavior. However, if you omit this principle this can lead to two major issues in the coding of the project.

5. Single Responsibility Principle:

This principle basically suggests you write a code that has its own functionality. That is single Responsibility refers to provide each module of the code with its own function so that you can modify it according to your needs in the future.

6. The Separation of Concerns Principle:

According to this principle, you should write a non-overlapping encapsulation. Apart from that, this encapsulation should not know each other. In easy language, Separation of Concerns is similar to the Single Responsibility but in a more abstract level of programming. The perfect example that can be taken is the Model-View-Controller (MVC) paradigm.

8. Abstain from Premature Optimization:

This coding principle ensures that you only write only the codes that will optimize the tendency to speed up the algorithms. No Premature optimization is similar to the YAGNI principle.

9. Refactor Principle:

Accept the hard truth about programming that you will not get to the full results until you make the full coding for the project. Thus this principle explains that the "code rarely comes out right the first time.You must use refactor principle often while writing programs.

10. Clean Code Over Clever Code Principle:

This principle will suggest you to write clean code rather than clever codes. It's hard to leave the ego to write clever code but as we mentioned above that it is important to write clean code. This is to ensure that you don't mess it up.

11. Create Useful Program based on the need of the Customer:

A programmer should focus on developing application that meets the specification of the end user. It is really important to know what exactly the customer wants from the program.

13. Practice, Practice, Practice:

As a programmer, you have to practice a lot. This is a very important principle that is needed in everyday life. The more you practice the more you will figure out the wrong places. This will make you more prone to bring out more practical and quality work.

SECTION THREE

Java as an OOP Programming Language: An Introduction

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]). Java is a complete Object Oriented Programming Language and has support for developing desktop, mobile and web-based applications of various sizes and functionalities.

Programming in Java requires that we make use of objects and classes. Learners are expected to think programming in terms of objects and classes.

Java Keywords

A keyword is a word that has pre-defined meaning to the Java compiler. These keywords can not be used as constant or variable or any other identifier names. They are used for carrying out various programming tasks. The following list shows the reserved words in Java.
They are: **abstract, main, boolean, int, short, break, class, scanner, package,else, emun,if,goto,import,native, instance of,interface,implements,long,const,case, private, public,char, string and so on.**

The general structure of a Java Program. Every Java program must have a main class and a main method.
A Java program is expected to have the following parts:

The comment (this is not compulsory but it is encouraged). The purpose of comment is for proper documentation of the program.
Main class
Main Method
Variable declaration part
Codes (for solving the identified problem).

Sample Java Program Structure

```
//This is a sample general structure that every Java program follows
public class OyeDemo1
{
public static void main (String args[])
{
variable declaration;
program codes
}
}
```

Comments in Java

Java supports single-line and multi-line comments very similar to C and C++. All characters available inside any comment are ignored by Java compiler. The purpose of comments in a Java program is to document.

Example of valid comments in a Java program

(a)
```
/* This is my first java program.
 * This will print 'Hello World' as the output

 * This is an example of multi-line comments. */
```

(b)
```
// This is an example of single line comment
```

(c)
```
/* This is also an example of single line comment. */
```

Tools for Programming Java

The tools for programming Java can be classified as console-based or GUI based.

The console based tool is the Java Development Kit (JDK) that allows programmer to enter Java codes into a text editor (e.g. notepad) and then compile and interpret the Java codes from the command prompt or DOS prompt.

The other category is the Graphical User Interface (GUI) tool. Examples of such tools are popular (Integrated Development Environments) IDEs. Examples are: Netbeans Java, Eclipse java, JBuilder, JDeveloper and so on.

However, for this course, the choice IDE tool is Netbeans Java. Netbeans is an open source tool that supports Java Programming. Neatbeans Java can be downloaded free of charge from **www.netbeans.org.** Then, you are expected to install it on your system.

Software Tools You Will Need to Program Java

Software Requirements

To carry out practical implementation of software applications in Java environment, one will also need the following softwares −

- Linux 7.1 or Windows XP/7/8/10 operating system as platform
- Java JDK 7 or 8 or 9 or new version
- Microsoft Notepad or any other text editor. Most times, the text editors are already available with the operating systems that users are using.

- Java IDE Tool such as Eclipse, Netbeans, JDeveloper,JBuilder e.t.c (For this class we chose Netbeans Java).

Explanation of Java Integrated Development Environments (IDEs) and Console tools

To write your Java programs, you will need a text editor or a Java IDE. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following −

- **Notepad** − On Windows machine, you can use any simple text editor like Notepad

- **Netbeans** − A Java IDE that is open-source and free which can be downloaded from https://www.netbeans.org/index.html.

- **Eclipse** − A Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org/.

- **JDeveloper**
- **JBuilder** and many others.

Note for beginners:

It is suggested herein that you must cultivate the habit of developing good programming practice skills, write your codes neatly in a sheet of paper or practical note and then key them in into the java compiler that you are using.

Some good examples of Java Program
Example 1:

A Sample Java Program.

```
public class MyFirstJavaProgram {
```

```
  public static void main(String []args) {
    System.out.println("Hello Sandwich Computer Science students");
  }
}
```

Another example of a simple Java Program that compute sum of two integer values

```
/* This is a Java program to add two initialised values */

public class MultiplicationProgram
{
public static void main (String args[])
{
int value1=45;
int value2=12;
int result=value1*value2;
System.out.println(result);
  }
}
```

Some concepts in Java Programming: Brief Introduction

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods, and instance variables mean.

- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

- **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

- **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

- **Core Re-usability**: Java as an object oriented programming language has full facilities to support code re-usability. This statement means that Java allows programmers not to re-invent the wheel. Rather, it allows them to re-sue code on a particular software project.

Data Types in Java

The data types supported in Java are many. Variables are reserved memory locations to store values. This means that when you create a variable you reserve some space in the memory.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

A variable in Java must be a specified data type. When we want to use a variable of a particular data type, we are expected to declare the variables. For examples, variables of different types can be declared as follows:

```
int myNum = 8;              // Integer (whole number)
float myFloatNum = 5.99f;   // Floating point number
char myLetter = 'C';        // Character
boolean myBool = true;      // Boolean
String myText = "Welcome to Al-Hikmah University";     // String
```

Data types in Java are divided into two groups:

- Primitive data types - includes byte, short, int, long, float, double, boolean and char
- Non-primitive data types - such as String, Arrays and Classes (you will learn more about these in a later chapter)

There are two classification of data types available in Java. They include:

- Primitive Data Types
- Reference/Object Data Types

Note: At the level of this course emphasis will only be on Primitive data types only. Now lets us discuss Primitive data types.

Primitive Data Types

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a keyword. Let us now look into the eight primitive data types in detail. Examples of the primitive data types are:

byte
short
int
long
float
double
Boolean and character (char)

Basic Syntax of Java

To write efficient Java programs, it is very important to know the syntax. For example, it is very important to keep the following points in one's mind.

- **Case Sensitivity** − Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.

- **Class Names** − For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

- **Every Java program must have a main class and a main method**.

- **With Java, you can build a wide range of applications**: desktop, distributed web-based applications.

- **In this course, we are only going to consider building desktop**-based Java applications.

  Example: *class MyFirstJavaClass*

- **Method Names** − All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

Example: *public void myMethodName()*

- **Program File Name** − Name of the program file should exactly match the class name.

  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

  Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*.

  For you to understand what I am talking about better, let us quickly write the format of how a program looks in Java Language. A typical java program can look as follows:

```
/* A java program to show how a program codes in java look */
public class ProgramExample1
{
public static void main(String args[])
{
Variable declaration
Java program codes;
}
}
```

Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called **identifiers**. It is important for every Java programmer to master this in the process of building applications.

In Java, there are several points to remember about identifiers. They are as follows −

- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).

- After the first character, identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value.
- Examples of illegal identifiers: 123abc, -salary.

These identifiers are used to accept values into the program and they must follow the laid down rules.

For instance, a Java identifier can start with a letter, an underscore or a dollar sign
A Java identifier must not be a keyword
A valid Java identifier must not have blank space

Java Modifiers

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers −

- **Access Modifiers** − default, public , protected, private
- **Non-access Modifiers** − final, abstract, strictfp

We will be looking into more details about modifiers in the next section.

Java Variables

Following are the types of variables in Java −

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static Variables)

SECTION FOUR
Object-Oriented Programming in Java

If you've never used an object-oriented programming language before, you'll need to learn a few basic concepts before you can begin writing any code. This lesson will introduce you to objects, classes, inheritance, interfaces, and packages. Each discussion focuses on how these concepts relate to the real world, while simultaneously providing an introduction to the syntax of the Java programming language.

Object

An object is a software bundle of related state and behavior. Software objects are often used to model the real-world objects that you find in everyday life. This lesson explains how state and behavior are represented within an object, introduces the concept of data encapsulation, and explains the benefits of designing your software in this manner. That is, Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Class

A class is a blueprint or prototype from which objects are created. This section defines a class that models the state and behavior of a real-world object. It intentionally focuses on the basics, showing how even a simple class can cleanly model state and behavior.

Inheritance

Inheritance provides a powerful and natural mechanism for organizing and structuring your software. This section explains how classes inherit state and behavior from their superclasses, and explains how to derive one class from another using the simple syntax provided by the Java programming language.

Interface

An interface is a contract between a class and the outside world. When a class implements an interface, it promises to provide the behavior published by that interface. This section defines a simple interface and explains the necessary changes for any class that implements it.

Package

A package is a namespace for organizing classes and interfaces in a logical manner. Placing your code into packages makes large software projects easier to manage. This section explains why this is useful, and introduces you to the Application Programming Interface (API) provided by the Java platform.

Illustration of the concepts based on examples

Following is a sample of a class.

Example

```
public class Dog {
   String breed;
   int ageC
   String color;

   void barking() {
   }

   void hungry() {
   }

   void sleeping() {
   }
}
```

A class can contain any of the following variable types.

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

- **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

Following are some of other important topics that need to be discussed when looking into classes of the Java Language.

Constructors

When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Following is an example of a constructor − **Example**

```
public class Puppy {
  public Puppy() {
  }

  public Puppy(String name) {
    // This constructor has one parameter, name.
  }
}
```

Java also supports Singleton Classes where you would be able to create only one instance of a class.

Note − We have two different types of constructors. We are going to discuss constructors in detail during the class lectures.

Creating an Object

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the **new** keyword is used to create new objects.

There are three steps when creating an object from a class −

- **Declaration** − A variable declaration with a variable name with an object type.
- **Instantiation** − The 'new' keyword is used to create the object.

- **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Note: Effort will be made to teach and demonstrate how this new keyword is used in Java for creating an instance of a class, God willing.
Following is an example of creating an object −

Example

```
public class Puppy {
  public Puppy(String name) {

    // This constructor has one parameter, name.
    System.out.println("Passed Name is :" + name );
  }

  public static void main(String []args) {

    // Following statement would create an object myPuppy
    Puppy myPuppy = new Puppy( "Ajasco" );

  }
}
```

If we compile and run the above program, then it will produce the following result −

Output

```
Passed Name is :Ajasco
```

Accessing Instance Variables and Methods

Instance variables and methods are accessed via created objects. To access an instance variable, following is the fully qualified path −

```
/* First create an object */
ObjectReference = new Constructor();

/* Now call a variable as follows */
ObjectReference.variableName;

/* Now you can call a class method as follows */
ObjectReference.MethodName();
```

Example

This example explains how to access instance variables and methods of a class.

```
public class Puppy {
  int puppyAge;
```

```
   public Puppy(String name) {

      // This constructor has one parameter, name.
      System.out.println("Name chosen is :" + name );

   }

   public void setAge( int age ) {
      puppyAge = age;
   }

   public int getAge( ) {
      System.out.println("Puppy's age is :" + puppyAge );
      return puppyAge;
   }

   public static void main(String []args) {
      /* Object creation */
      Puppy myPuppy = new Puppy( "tommy" );

      /* Call class method to set puppy's age */
      myPuppy.setAge( 2 );

      /* Call another class method to get puppy's age */
      myPuppy.getAge( );

      /* You can access instance variable as follows as well */
      System.out.println("Variable Value :" + myPuppy.puppyAge );

   }
}
```

If we compile and run the above program, then it will produce the following result −

Output of the Above Program

```
Name chosen is :tommy
Puppy's age is :2
Variable Value :2
```

Java Package Explained

In simple words, it is a way of categorizing the classes and interfaces. When developing applications in Java, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must as well as makes life much easier. A package is made up of collection of several related classes.

Note: The default package in every Java program is language package and it is of the format **import java.lang.***

Others are I/O Package, Networking Package, Utility Package, and AWT package.
Note: At your level (beginning programmer), you are only expected to master the use of lang,I/O and utility package.

Import Statements

Import statement is used for handling the use of packages and classes in a program that require them. In Java if a fully qualified name, which includes the package and the class name is given, then the compiler can easily locate the source code or classes. Import statement is a way of giving the proper location for the compiler to find that particular class.

For example, the following line would ask the compiler to load all the classes available in Input/Output package

```
import java.io.*;
```

Another example is the utility package containing several classes
Import java.util.*;
Note:
Depending on the program that is being targeted, you are expected to import the appropriate packages and classes.
Solution
First of all write the Java equivalent of the mathematical expression given above as
Y=3*x-15*B
The targeted variable is Y, while input variables are x and B
Since we are entering values from the keyboard, we use I/O package and Utility package

```
/This is a Java program to find the value of y given that y=3x-15B
import java.io.*;
import java.util.*;
public class EquationExample
{
Public static void main (String args[])
{
float y,
float x;
float B;
Scanner value=new Scanner (Syetem.in);
x=value.nextFloat();
B=value.nextFloat();
Y=3*x-15*B
System.out.print(Y);
}
}
```

Access Control Modifiers

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are −

- Visible to the package, the default. No modifiers are needed.
- Visible to the class only (private).
- Visible to the world (public).
- Visible to the package and all subclasses (protected).

Non-Access Modifiers

Java provides a number of non-access modifiers to achieve many other functionality.

- The *static* modifier for creating class methods and variables.
- The *final* modifier for finalizing the implementations of classes, methods, and variables.
- The *abstract* modifier for creating abstract classes and methods.
- The *synchronized* and *volatile* modifiers, which are used for threads.

Note: Efforts will be made to explain and implement some of these concepts during class lectures and practical sessions

Basic Operators in Java Language

In this section, we will be discussing about Basic Operators used in Java Language. The tips provided will give you an overview of how these operators can be used during application development.
Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups −

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Misc Operators

Effort will be made to teach these operators and how they are used in Java programs during class lectures.
Loops and LOOPING in Java

This section explains about loop control in Java programming. The unit will describe various types of loops and how these loops can be used in Java program development and for what purposes they are being used.

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A **loop** statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages

Java programming language provides the following types of loop to handle looping requirements

For loop in Java

FOR LOOP is used to traverse collection of elements including arrays.

Syntax

Following is the syntax of enhanced for loop −

```
for(declaration : expression) {
   // Statements
}
```

- **Declaration** − The newly declared block variable, is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.

- **Expression** − This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

```
    for(int x : numbers ) {
       System.out.print( x );
       System.out.print(",");
    }
    System.out.print("\n");
    String [] names = {"Ibrahim", "Larry", "Bola", "Lakunle"};

    for( String name : names ) {
       System.out.print( name );
       System.out.print(",");
    }
  }
}
```

This will produce the following result/output −

Output

10, 20, 30, 40, 50,
Ibrahim, Larry, Bola, Lakunle,

Decision Making in Java

Decision making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

To be able to enter values from a keyboard, we make use of a class called Scanner Class. We use the new operator to create an instance of the scanner class called scanner variable.
Lets use a sample program below so that you can get the point better

Programming Problem

Write a Java program to find the average of three integer values entered from the keyboard

Solution

```
/*This is a Java program to find average of three integers */
import java.io.*;
import java.util.Scanner;
public class AvgCalculator
{
public static void main (Stirng args[])
{
Scanner p=new Scanner (System.in);
int x1;
int x2;
int x3;
int total,average;
x1=p.nextInt();
x2=p.nextInt();
x3=p.nextInt();
total=x1+x2+x3;
average=total/3;
System.out.println(average);
}
}
```

SECTION FIVE
Practice Questions and Exercises on Object-Oriented Programming using Java

Use the questions and exercises presented in this section to test your understanding of some of the concepts that are being taught so far.

i. List and explain the stages involved in Software Development Life Cycle
ii. Write th general format of a Java program
iii. Write short notes on each of the following: (i) Java Keyword (ii) Program (iii) Syntax

iv. Given an Employee class that has both class and function members, write a Java program to accept and display the details of an employee at XYZ Nig Limited

v. Write a Java program to that has StudentAge as class name. Then, your program should be able to compute the age of student given the birth date and the current date of the student

vi. Write a Java program having XCalculator as its class name. Thereafter, write program procedures that should be able write to find the value of X given that X=2.0B +17.0K/3.0

vii. Briefly explain any four features of Java as a programming language

viii. Write a Java program to find the value of X given that X=2.0B +17.0K/3.0

ix. Enumerate any ten principles that can guide the development of a reliable software solution.

x. Write a Java program that can accept scores of a students in seven subjects and then compute the average score.
xi. Write a Java program to find the sum of integers from 10 to 130
xii. Define each of the following: (i) Looping (ii) Class (iii) Instance variable
xiii. Write a Java program that can compute the sum of forty integer values entered from the keyboard
xiv. Write the general structure of FOR LOOP statement in Java
xv. Identify any four Java IDE tools that you know

xvi. Write a Java program to accept the an item name, unit price and quantity and then computer the amount that a consumer incur while buying that particular item.

Note: This material is an introduction but contains vital points that can actually be of great help to you in further learning of programming. So, take the points contained in it very serious. The lecture notes/jottings in class will expose some of the concepts discussed herein better. The lecturer will strive to provide many more relevant programming examples and exercises during lectures.

I wish you all the best in your academic pursuit.

Should you have any question outside the class concerning this course, feel free to reach me.

I wish you all the best in your academic pursuits.

J. References

Deitel & Deitel (2015). Java: How to Program (Early Objects) 9th Edition, Paul J. Deitel, Deitel & Associates, Inc.
TutorialPoint (n.d.). Java Tutorial retrieved from https://www.tutorialspoint.com/java/index.htm

Vishal Goenka (1999). Runtime Programming in Java: A Technology Primer

K. Recommended books/Materials /Web Documents

GFG Tutorials (2017). Programming Section

TutorialPoint (n.d.). Java Tutorial, retrieved from https://www.tutorialspoint.com/java/index.html