Day3

Parameter versus Argument

```
int sum( int a, int b ) //a, b => Function Parameters / Parameter
{
    int result = 0;
    result = a + b;
    return result;
}
int main( void )
{
    int x = 10;
    int y = 20;
    //int result = sum( 10, 20 ); //10, 20 => Function Argument / Argument
    int result = sum( x, y );//x, y => Function Argument / Argument
    printf("Result  :   %d\n", result);
    return 0;
}
```

Command Line Argument

- Consider code in C language

```
./Main.exe Sandeep 33 45000.50    //Windows
./Main.out Sandeep 33 45000.50    //Linux
//"./Main.out" "Sandeep" "33" "45000.50"
```

```
/*
* argc :
    - argument counter.
    - Function Parameter
    - It keeps counter of arguments including file name
* argv :
    - argument vector.
    - Function Parameter
    - It is array of character pointer which stores address of string.
*/
int main( int argc, char *argv[ ] )
{
    //argv[ 0 ] => "./Main.out"

    //argv[ 1 ] => "Sandeep";
    char *name = argv[ 1 ];

    //argv[ 2 ] => "33"
```

```
    int empid = atoi( argv[ 2 ] );

    //argv[ 3 ] => "45000.50"
    float salary = atof( argv[ 3 ] );

    return 0;
}
```

- Consider command line argument in Java

```
javac Program.java  => Program.class
java Program Sandeep 33 45000.50f
//java Program "Sandeep" "33" "45000.50f"
```

```
class Program{
    public static void main(String[] args) {
        //args[ 0 ]   =>  "Sandeep"
        String name = args[ 0 ];

        //args[ 1 ]   =>  "33"
        int empid = Integer.parseInt( args[ 1 ] );

        //args[ 2 ]   =>  "45000.50f"
        float salary = Float.parseFloat( args[ 2 ]);
    }
}
```

Java Buzzwords / Features

1. Simple
2. Object Oriented
3. Architecture Neutral
4. Portable
5. Robust
6. Multithreaded
7. Secure
8. Dynamic
9. High Performance
10. Distributed

Java is a Simple programming language.

- C language invented during 1969-1972.
- C++ language invented in 1979.
- Java language invented in 1991.

- Java language is derived from C and C++. In others words, Java follows syntax of C and Concepts of C++.
- Syntax of Java is simpler than syntax of C and C++.
    1. No need to include header file.
    2. Do not support structure and union. But it supports enum.
    3. Do not support default argument.
    4. Do not support constructure member initializer list.
    5. Do not support delete operator and destrctor.
    6. Do not support friend function and friend class.
    7. Do not support copy constructor and operator overloading.
    8. Do not support private and protected mode of inheritance.
    9. Do not support multi-class inheritance in other words, It doesn't support multiple implementation inheritance.
    10. We can not delcare/define global variable and function.
    11. Do not support pointer.

Java is a Object Oriented programming language.

- Alan Kay -> Inventor of OOPS and Simula.
- Grady Booch : Inventor of UML : Author of : Object Oriented Analysis and Design With Application.
- According to Grady Booch there 4 major and 3 minor pillars/parts/elements of oops.
- 4 major pillars of oops
    1. Abstraction
    2. Encapsulation
    3. Modularity
    4. Hierarchy
- According to Grady Booch, if we want to consider any language OO then it must support 4 major pillars of OOPS.
- 3 minor pillars of oops
    1. Typing / Polymosphism
    2. Concurrency
    3. Persistence
- If language support to above features then it will be considered as useful but not essential to classify language OO.
- Since Java support to all major and minor pillars of oops hence it is conisidered as object oriented.

Java is Architecture Neutral

- CPU Architectures : X86, X64, ARM, POWER PC, SPARK, APLHA etc.
- Java compiler convert java source code into bytecode.
- Native CPU can not execute bytecode code directly.

- Execution engine of JVM converts bytecode into native code.
- .class file contains bytecode which is CPU neutral code makes Java architecture neutral.
- Since Java is architecture neutral, java developer need not to worry about underlying hardware and operating system.

Java is Portable programming language

- Term Portable is related to executable.
- Java is portable because Java is architecture neutral.
- Size of data types on all the platform is constant/same.
    1. boolean : Not Mentioned
    2. byte : 1 byte
    3. char : 2 bytes
    4. short : 2 bytes
    5. int : 4 bytes
    6. float : 4 bytes
    7. double : 8 bytes
    8. long : 8 bytes
- Since Java is portable, It doesn't support sizeof operator.

Java is Robust programming language.

- Java is robust programming language because of 4 reasons:
    1. Java is architecture neutral.
    2. Java is object oriented programming language.
    3. Java's memory management.
    4. Java's Exception Handling.

Java is Multithreaded Programming language.

- JVM is responsible for managing execution of Java application.
- Thread : Light weight process / sub process is called thread.
- When JVM starts execution of Java application then it also starts execition of 2 thread i.e main thread and garbage collector.
- Because of main thread and garbage collector, every java application is Multithreaded.
- Main Thread
    1. It is a user thread / non daemon thread.
    2. It is responsible for invoking main method.
- Garbage Collector / Finalizer
    1. It is a daemon thread / background thread.
    2. It is responsible for deallocating memory of unused objects.

Scanner Demo : For Input

- Scanner is a final class declared in java.util package.
- Instantiation
    - Process of creating instance from class is called Instantiation.

```
java.util.Scanner sc = new java.util.Scanner(System.in);
```

```
import java.util.Scanner;
Scanner sc = new Scanner(System.in);
```

- Methods:
    1. public String nextLine()
    2. public int nextInt()
    3. public float nextFloat()
    4. public double nextDouble()
- If we want to call non static method then it is ncessary to use object reference.

Eclipse Introduction

print, println,printf

- print method print o/p on console and keep cursor on same line.
- println method print o/p on console and move cursor to the next line.
- printf is used to print formated output on console.

```
String nm1 = "Prashant Lad";
int id1 = 1234;
float sal1 = 25000.50f;
System.out.printf("%-15s%-5d%-10.2f\n", nm1, id1, sal1 );
```

OOPS Concepts

- Consider following examples:

1. Let us consider Date:
    - "24/11/2020"
    - day, month, year => int
    - day, month and year are of type int which a related to int.

```
class Date{
    int day;
    int month;
    int year;
}
```

2. Let us consider Color
    - Color value is represented using RGB.
    - red, green, blue => int

- red, green, blue are of type int which a related to Color.

```
class Color{
    int red;
    int green;
    int blue;
}
```

3. Let us consider Account
    - number : int
    - type : String
    - balance : float
    - number, type, balance are realated to Account.

```
class Account{
    int number;
    String type;
    float balance;
}
```

4. Let us consider Employee
    - name : String
    - empid : int
    - salary : float
    - name, empid and salary are related to Employee.

```
class Employee{
    String name;
    int empid;
    float salary;
}
```

- If we want to group related data elements together then we should use Class.
- class is a keyword in Java.
- If we want to define class then first it is necessary to understand problem statement.
- A variable declared inside class is caled field.
- If we want to store value inside non static field then we must create object of the class.
- In Java, object is called as instance.
- class is non primitive / reference type.
- If we want to create instance of a class then we should use new operator.
- Java instance get space on Heap.
- Non static field get space once per instance according order of their declaration inside class.
- In java all the instances are anonymous.

- If we want to perform operations on instance then it is necessary to create object reference / reference.
- Instance w/o reference is called anonoymous instance.
- If we want to give controlled access to the field then we should declare field private and give access to it using method.
- If we want to perform operations on instance then we should define method inside class.