Day 2

Data Type

```
* Data describes 3 things
    1. Memory : how much memory is required to store the data.
    2. Nature : which type of data can be stored in memory
    3. Operation : Which Operations can be performed on the data.
* Types of data type
    1. Primitive Data type
    2. Non Primitive Data Type
```

Primitive Data Types(8)

```
* It is also called as value type.
* There are 8 Primitive/value types in java.
* variable / instance of primitive/value type get space on stack segment.
```

```java
class Program
{
    public static void main(String[] args)
    {
        int number = 10;
    }
}
```

```
* Following are primitive types
    1. boolean  Not Specified.
    2. byte     ( 1 byte )
    3. char     ( 2 bytes )
    4. short    ( 2 bytes )
    5. int      ( 4 bytes )
    6. float    ( 4 bytes )
    7. long     ( 8 bytes )
    8. double   ( 8 bytes)
* In Java, primitive types are not classes. e.g int is not class.
```

Wrapper class

```
* For every pritive type java has given a class. It is called "Wrapper
class".
* e.g
    int : primitive type
    Integer : Wrapper class
1. boolean  :   java.lang.Boolean
2. byte     :   java.lang.Byte
3. char     :   java.lang.Character
4. short    :   java.lang.Short
5. int      :   java.lang.Integer
6. float    :   java.lang.Float
7. long     :   java.lang.Long
8. double   :   java.lang.Double
* All the wrapper classes are final.
```

Default values

```
1. boolean  :   false
2. byte     :   0
3. char     :   +u0000
4. short    :   0
5. int      :   0
6. float    :   0.0f
7. long     :   0L
8. double   :   0.0d
* In general, variable of value type, by default contains 0 value.
```

Non Primitive Data Types(4)

```
* It is also called as reference type.
* There are 4 non primitive/reference type in java
    1. Interface
    2. Class
    3. Enum
    4. Array
* Instance of non primitive / reference type get space on Heap section.
```

- In java, if we want to use any local variable then it is mandatory to store value inside it.

```
class Program
{
    public static void main(String[] args)
    {
        int x;  //OK
        System.out.println(x);  //Error

        int y = 10;  //OK
        System.out.println(y);  //Ok

        int z;  //Ok
        z = 20; //OK
        System.out.println(z);  //Ok
    }
}
```

Widening

```
* Process of converting state/value of variable of narrower type into
wider type is called widening.
```

```
int num1 = 10;
//double num2 = ( double )num1;   //OK
double num2 = num1; //OK : Widening
```

```
* In case of widening, explicit type casting is optional.
```

Narrowing

```
* Process of converting state/value of variable of wider type into
narrower type is called Narrowing.
```

```
double num1 = 10.5d;
int num2 = ( int )num1; //OK : Narrowing
//int num2 = num1;    //Not Ok
```

```
* In case of Narrowing, explicit type casting is mandatory.
```

Boxing

```
* Process of converting state/value of variable of primitive/value type
into non primitive/reference type is called boxing.
```

```
int number = 10;
//Boxing
String strNumber = Integer.toString(number);
```

```
* valueOf() is static method of String class which is used to convert
state of variable of value type into String.
```

```
int number = 10;
//Boxing
String strNumber = String.valueOf(number);
```

```
int number = 10;
//Boxing
Integer n1 = Integer.valueOf(number);
```

UnBoxing

```
* Process of converting state/value of instance of non primitive/reference
type into  primitive/value type is called unboxing.
```

```
 String str = "125";
int num1 = Integer.parseInt(str); //UnBoxing
```

```
* parseXXX() is a method of wrapper class which is used to convert string
into primitive type.
* If string does not contain a parsable numeric value then parseXXX()
method throws NumberFormatException.
```

```
String str = "abc";
int num1 = Integer.parseInt(str); //NumberFormatException
```

Java Buzzwords

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 Simple

```
* Java language is derived from C & C++ i.e. Java language follows syntax
of C and Concepts of C++.
* Syntax of java is simpler than C/C++ hence java is simple
    1. No need to include header files
    2. Java do not support structure and union
    3. Java do not support delete operator and destructor
    4. It doesn't support friend function and class
    5. It doesn't support operator overloading
    6. It doesn't support multiple class inheritance hence therse is no
diamond problem and virtual base class.
    7. It doesn't support constructors member initializer list and default
argument
    8. There is no concept separate declaration and definition. Everything
is definition.
    9. We can not define anything globally.
    10. Java do not support casting operators
    11. Java do not support pointer arithmetic.
    12. Java do not support private and protected mode of inheritance.
* Since size of software which is required to develop java application is
small, java is simple.
```

Java is Object Oriented

```
* 4 Major pillars of OOPs
    1. Abstraction
    2. Encapsulation
    3. Modularity
    4. Hierarchy
* 3 minor pillars
    1. Typing
    2. Concurrency
    3. Persistance
* Since java support all major and minor pillars of oops, it is considered
as object oriented.
```

java is Architecture Neutral Language

```
* CPU Architecture's
    1. x86/x64
    2. ARM
    3. Power PC
    4. ALPHA
    5. Sparc
* After compilation, java compiler generates bytecode which is cpu
architecture neutral code which makes java architecture neutral.
```

Java is Portable programming Language

```
* Java is protable because it is architecture neutral.
* Java's Slogan is :
"Write Once Run AnyWhere"
* JVM is platform dependant
* JVM is integral part of JRE hence JRE also platform dependant
* JDK is platform dependant
* bytecode/.class file is architecture neutral
* Size of data types on all the platforms is constant hence java is truely
portable.
    1. boolean  Not Specified.
    2. byte     ( 1 byte )
    3. char     ( 2 bytes )
    4. short    ( 2 bytes )
    5. int      ( 4 bytes )
    6. float    ( 4 bytes )
    7. long     ( 8 bytes )
    8. double   ( 8 bytes)
* Since java is portable, it doesn't support sizeof operator.
```

Java is Robust Programming Language

```
1. Architecture Neutral
    * Since java is architecture neutral, developer need not not to do
hardware / OS specific code.
2. Object Oriented
    * Hierarchy allows developer to reuse existing code which reduces
developers effort, development time and cost.
3. Memory Management
    * In java developer can allocate memory for object but deallocating
that memory is a job of Garbage Collector / Finalizer.
4. Exception handling
```

```
    * Java compiler helps developer to handle exception which reduces
developers effort.
```

Java is multithreaded

```
* Program in execution is called process/task.
* In other words, running instance of a program is called process.
* Light weight process / sub process is called thread.
* In other words, thread is seperate path of execution, which runs
independantly.
* An ability of OS to execute single process at time is called
singletasking.
* e.g. MSDOS
* An ability of OS to execute multiple process at time is called multi
tasking.
* e.g All modern OS.
* It is possible to achive multitasking using process as well as thread.
* If any application uses single thread for execution then it is called
single threaded application.
* If any application uses multiple thread for execution then it is called
multi threaded application.
* When JVM starts execution of java application, it also starts execution
of "main thread" and "Garbage collector". Hence every java application is
multithreaded.
```

Main Thread

```
1. It is Non Deamon / User Thread
2. It is responsible for invoking main() method.
3. It's priority is 5.
```

Garbage Collector

```
1. It is also called as finalizer.
2. It is Daemon / Background Thread.
3. It is responsible for releasing / reclaiming / deallocating memory of
unused objects.
4. It's priority is 8.

* Thread is non java resource ( unmanaged resource ). With the help of
SUN/ORACLE's thread framework, we can use OS thread in java application.
In other words, java has given built in support to thread hence java is
considered as multithreaded.
```

Java is secure programming language.

```
* If we want to develop secure application then we use types declared in
java.security package. It means that SUN/ORACLE has given support to
achive security hence it is considered as secure.
```

Java is dynamic programming language

```
* In java, all the methods are by default virtual.
* We can run java application on any evolving hardware as well as
operating system hence it is truly dynamic
* We can add new types inside existing libraries(.jar) at runtime time.
```

Java is High Performance programming language.

```
* Performance of java is slower than C/C++.
* During conversion of bytecode into native cpu code, JIT compiler use
optimization technique which improves Performance of application.
* With the help of JNI, we can invoke code written in C/C++ into java
which help us to improve Performance of application
```

Java is distributed.

```
* RMI : Remote method invocation.
* It is java language feature which supports Service Oriented
Architecture( SOA).
* Java is distributed because it supports RMI.
```

- If we want to access any type outside package then we should use either F.Q.TypeName or import statement.
- java.lang package is by default imported in every .java file.

Stream

```
* It is an abstraction( object ) which is used to produce( write ) and
consume ( read ) information from source to destination.
* Console = Keyboard + Monitor
* Standard Stream objects of java which is associated with Console:
    1. System.in    :   Keyborad
    2. System.out   :   Monitor
    3. System.err   :   Error Stream
```

Console IO

```
* Scanner is final class declared in java.util package
* If we want to read tokens from Console, File etc, then we can use
Scanner class.
* Syntax:
    import java.util.Scanner;
    Scanner sc = new Scanner(System.in);
* Methods of Scanner class:
    1. public String nextLine()
    2. public int nextInt()
    3. public float nextFloat()
    4. public double nextDouble()
```

Class

```
* It is collection of fields and methods
    class ClassName
    {
        //Fields
        //Methods
    }
* Structure and behavior of object/instance depends of fields and methods
declared inside class hence class is considered as a
template/model/blueprint for instance.
* Class represent collection of instances which is having commong
structure and common behavior.
* e.g Car, Laptop
* If we want to achive Encapsulation then we should define class.
```

Instance

```
* In java, object is called as instance.
* An entity which is having physical existance is called instance.
* If entity has state, behavior and identity then it is called instance.
* e.g Maruti 800, MacBook Air etc.
* If we want to achive abstraction then we should create instance and
perform operations on it.
```

Access modifiers(4)

```
1. private( - )
2. package level private( ~ )
3. protected( # )
4. public( + )
```