Object Oriented Programming Using Java – UNIT 1

Fundamentals of Object Oriented Programming

- Object Oriented programming is programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
- OOP is a new way of organizing and developing programs & has nothing to do with any particular language. It attempts to eliminate some of the pitfalls of conventional programming methods.
- In OOP, computer programs are designed by making them out of objects that interact with one another. Several object-oriented languages are Java, C++, C#, Python, PHP< Ruby, Perl, Pascal, Objective-C, Common Lisp, and Smalltalk.
- Fundamentals of OOP are Abstraction, Encapsulation, Inheritance & Polymorphism.
- Since the invention of computer, many programming approaches have been tried. These include modular programming, top-down programming, bottom-up programming & structured programming.
- Primary motivation in each case has been to handle increasing complexity of programs that are reliable and maintainable.

Difference between Procedure & Object Orinetend Language

	Procedure or Function Oriented	**Object Oriented**
1	Programs are divide into functions	Programs are divided into objects
2	Importance is not given to data but to functions as sequence of action to be done	Importance is given to data rather than function because it works on real world.
3	Most of the functions share global data	Data structure characterizes objects
4	Data moves from function to function	Functions that operate data bind to form classes
5	It is top down approach	It is bottom up approach
6	It needs less memory	It needs more memory
7	It do not have access specifiers	It has access specifiers
8	It is less secure	It is more secure
9	It follows no overloading	It follows both operator & method overloading
10	Example: C, Fortran,	Example: C++, Java, .Net

Basic OOP Concepts

Classes- Class is a user-defined data type. It is template/blueprint for an object.

- It is a logical entity.
- It can also be defined as collection of objects.
- Class doesn't store any space.
- Once a class is defined we can create any number of objects belonging to that class.
- The behaviour of class is similar like any other built-in data type.

Object- Object is instance of class. It is an entity which has state & behaviour.

- It can be both physical & logical in nature.
- An object contains an address and takes up some space in memory.
- Objects can communicate without knowing details of each others data or code.
- Objects represents a person, place, data or any item that the program may handle.
- Example: A dog is an object because it has states i.e. colour, name, breed etc. as well as behaviors i.e. wagging the tail, barking, eating etc.

Abstraction- Hiding internal details and showing functionality.

- In java, we use abstract class and interface to achieve abstraction.
- It refers to act of representing essential features without including background details.
- Classes are defined as list of abstract attributes.

Encapsulation- Binding (or wrapping) code and data together into a single unit.

- Example: A capsule, it is wrapped with different medicines.
- Data Encapsulation is striking feature of a class.
- Here, the data is not accessible to the outside world & only those methods, which are wrapped in class can access it.
- These methods provide interface between objects data & program. This insulation of data from direct access by program is called **data hiding.**

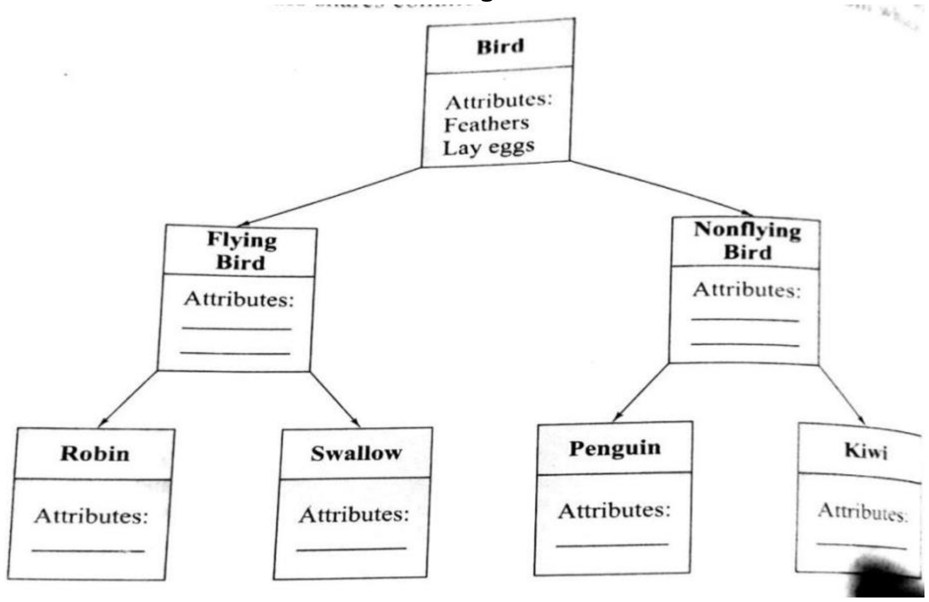
Example for understanding Abstraction & Encapsulation

The concept of data encapsulation and abstraction can be understood by considering simple example of a mobile phone. A mobile is electronic device, used for text & voice communication. How the internal circuitry of mobile works is irrelevant from user view point. He is only concerned about usage. Thus user is isolated from all the complex circuitry which is well **encapsulated** within the mobile phone.

Now, standard operations performed by mobile phone are- dialing/receiving a call, sending & receiving messages etc. Also mobile has address book, inbox, sent items etc. We can refer to these sets of data & operation as **abstract** of a generic mobile phone. All mobile phones of different make will adhere to this abstraction. So we can say all mobile phones from different manufactures are nothing but different instances of this common abstraction of generic mobile.

Inheritance- When one object acquires all the properties and behaviors of parent object.

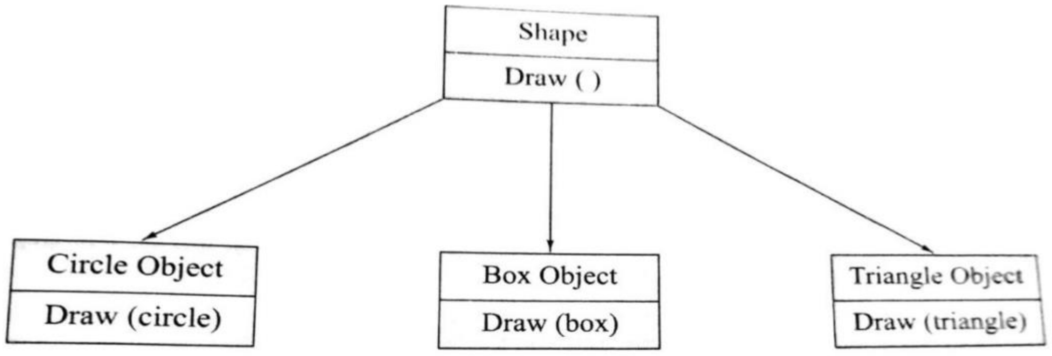
- It provides code reusability.
- It is also used to achieve runtime polymorphism.
- It supports hierarchical classification.
- It provides idea, by which we can add additional features to existing class without modifying it. This is possible by **deriving** new class from existing one.
- In Java, derived class is known as **subclass**. Each subclass defines only those features that are unique to it.
- Example: Consider a bird robin which is part of class flying bird, which is again part of class bird as shown in the below figure.



Polymorphism- Poly means ability to take more than one form. When one task is performed by different ways i.e. known as polymorphism.

- An operation may exhibit different behaviour in different instances.
- Example, consider operation of addition. For numbers, operation will generate a sum & for strings it will produce a third string by concatenation.
- In java, we use method overloading and method overriding to achieve polymorphism.
- It is extensively used in implementing **inheritance**.

Consider a example of Draw operation for various shape instances



History of Java

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of java starts from Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions etc. But, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted and Dynamic". Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. We can call Java as a revolutionary technology because it has brought fundamental shift in hoe we develop & use programs. Nothing like this had happened to the software industry before.

Following are the major points that describe the history of java.

1) James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan initiated the Java language project in June 1991. The small team of sun engineers called Green Team.

2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called "Greentalk" by James Gosling

4) After that, it was called Oak and was developed as a part of the Green project.

5) Why Oak? Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

6) In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.

7) Why had they chosen java name for java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

8) Java is an island of Indonesia where first coffee was produced (called java coffee). Since java was so unique, most of the team members preferred java.

9) Notice that Java is just a name not an acronym.

10) Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995. **James Gosling** is called **Father of Java**

11) JDK 1.0 released in (January 23, 1996). There are many java versions that have been released. Current stable release is Java SE 10(March 2018)

Features of Java

Java is defined as general purpose high level computer programming language that is concurrent, class-based, object-oriented & has platform independent architecture.

The features of Java are also known as java buzzwords. They can also be called as advantages of java.

1. **Simple** - Java is very easy to learn and its syntax is simple, clean and easy to understand. The syntax is similar to C, C++ & has removed
2. **Object Oriented** - Everything in Java is an object. It means we organize our software as a combination of different types of objects that incorporates both data and behavior.
3. **Platform Independent** – Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, and Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere (WORA).
4. **Secured** – With Java we can develop virus-free systems. Java is secured because: it has no explicit pointers, it runs inside JVM, class loader, byte code verifier & security manager.
5. **Robust** – because of strong memory management, automatic garbage collection, exception handling & type checking mechanism. It is resistant to failures.
6. **Architectural Neutral** – because there is no implementation dependent feature: size of primitive type is fixed. In C programming, int data type occupies 2 bytes for 32-bit architecture and 4 bytes for 64-bit architecture. But in java, it occupies 4 bytes for both 32 and 64 bit architectures.
7. **Portable** – because it facilitates you to carry the java bytecode to any platform. It doesn't require any type of implementation.
8. **High Performance** – Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.
9. **Interpreted** - Java interpreter is high quality, high performance & less space occupying, making it one of the fastest in current scenario.
10. **Distributed** - because it facilitates users to create distributed applications in java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.
11. **Multithreaded** - A thread is like a separate program. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area.
12. **Dynamic** - Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages i.e. C and C++.

Java Development Kit (JDK) Environment

JDK is a software development environment which is used to develop java applications and applets. It physically exists. It contains JRE + development tools.

JDK includes tools such as

- appletviewer - enables us to run java applets
- javac (compiler)- translates java source code into byte code files
- java (interpreter) - which runs applet & apps by reading bytecode files
- javap (Java disassembler) - converts byte code files into program description.
- javah - produces header files for use with native methods
- javadoc - creates HTML format documentation from java source code files.
- jdb (debugger) - helps to find errors in our program.

Java Virtual Machine (JVM)

JVM (Java Virtual Machine) is an abstract machine.

- It is called virtual machine because it doesn't physically exist.
- It provides runtime environment in which java bytecode can be executed.
- **Byte code** is highly optimized set of instructions.
- JVMs are available for many hardware and software platforms.
- The JVM performs following main tasks: Loads code, Verifies code, Executes code and Provides runtime environment

Java Runtime Environment (JRE)

JRE facilitates the execution of programs developed in java

- It is the implementation of JVM. It physically exists.
- It contains set of libraries + other files that JVM uses at runtime.
- It primary comprises of the following

  A) JVM

  B) Runtime class libraries - required for execution of java program

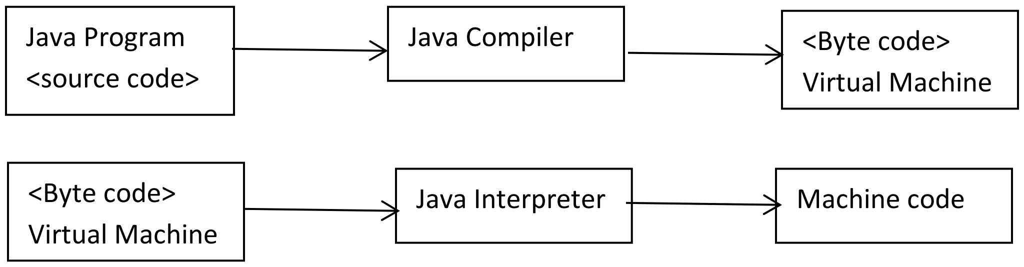
  C) User interface toolkits - AWT & Swings used to interact with application program.

  D) Deployment Technologies:

    I) Java plug-in - enables execution of java applet

    II) Java Web start - enables remote deployment of application.

Following Diagram gives representation of java compiler & interpreter functionality


Java Program <source code>
Java Compiler
<Byte code> Virtual Machine
<Byte code> Virtual Machine
Java Interpreter
Machine code

Identifiers and keywords

Keywords are predefined; reserved words used in Java programming that have special meanings to the compiler.

- int score;
- Here, int is a keyword. It indicates that the variable score is of integer type .
- You cannot use keywords like int, for, class etc as variable name (or identifiers) as they are part of the Java programming language syntax.

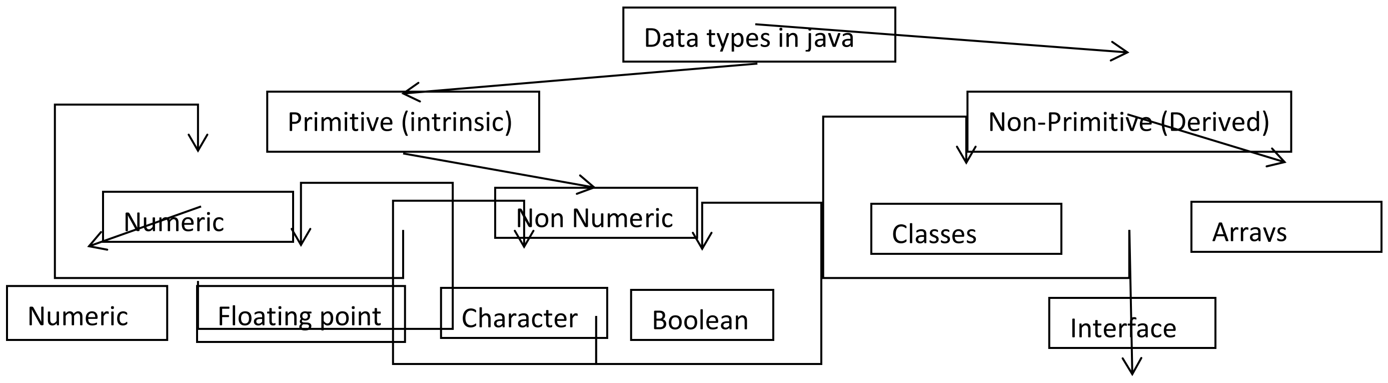
Identifiers are the name given to variables, classes, methods etc.

- int score;
- Here, score is a variable (an identifier).

- Rules for Naming an Identifier
  - Identifier cannot be a keyword.
  - Identifiers are case-sensitive.
  - It can have a sequence of letters and digits. However, it must begin with a letter, $ or _.
  - The first letter of an identifier cannot be a digit.
  - It's convention to start an identifier with a letter rather and $ or _.
  - Whitespaces are not allowed.
  - Similarly, you cannot use symbols such as @, #, and so on.
- Example
  - valid Identifiers - score, marks, total,
  - Invalid Identifiers - @pk, String, oc ta.

Datatypes

It specifies the size & type of values that can be stored. Following diagram shows the classification of datatypes



- Integer datatype can further have Byte, Short, Int and Long values
- Floating point can have Float & Double value.

Variables

A variable is name of memory or storage location.

- It specifies what type of data variable will hold
- Place of declaration decides scope of variable.
- There are three types of variables in java: local, instance and static.

Type Casting

The process of converting one datatype into another is called casting.

- It is used when there is need to store value of one type into variable of another type.
- Four integer types can be cast to any other type except boolean.
- Casting into smaller tpe may result in loss of data
- Casting float to integer may result in loss of fraction part.
- **Automatic conversion** - For some type we can assign values without a cast
  byte b = 75;
  Int a = b;
- The process of assigning smaller type to large is called **widening or promotion**
- Assigning larger types to smaller is called **narrowing**.

Java coding convention

It is set of rules to follow while naming your identifiers (be class, interface, methods etc). but it is not forced to follow. So, it is known as convention not rule. It increasing code readability.

NAME	**Convention**
Class	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
Interface	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
Method	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc
Variable	should start with lowercase letter e.g. firstName, orderNumber etc.
package	should start with lowercase letter e.g. firstName, orderNumber etc.
Constants	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Expressions

Operators are used in program to manipulate data & variables forming part of mathematical & logical expressions. Java operators are divided into following categories

Arithmetic Operators	+ - * /
Relation Operators	<<= >>= == !=
Logical Operators	&& || !
Assignment Operators	= (op=expr)
Increment & Decrement Operators	++ --
Conditional Operators	?: (exp1 ? exp2 : exp3)
Bitwise Operators	& ! ^ ~ <<>>>>>
Special Operators	Instanceof dot(.)

Decision Making: Control Structures

Java languages poses decision making capabilities & supports following statements known as control or decision making statements. There are primarily three decision making statements:

- if statement
- switch statement
- conditional operator

1. if statement

It is powerful decision making statement and is used to control flow of execution. It is used in conjunction with an expression. It can be implemented in 4 ways

a. Simple if statement

```
if(test expression) {
        Statement block
}
```

b. if… else statement

```
if(test expression) {
        True statement block
}
else {
        False statement block
}
```

c. Nested if.. else statement

```
if(test condition1) {
        If(test condition2) {
                Statement1;
        }
        else{
                Statement2;
        }
}
else {
        Statement3;
 }
```

d. else if ladder

```
if(test condition1)
        Statement1;
else if(condition2)
        Statement2;
else if(condition3)
        Statement3;
else
        Statement4;
```

2. switch statement

It tests the value of given variable against a list of case values & when a match is found, block of statements associated with that case is executed.

```
switch(expression) {
        case vale1:
                Block1;
                break;
        case vale2:
                Block2;
                break;
        ……..
        default:
                Default block;
                break;
}
```

Expression can be integer or character. The break signifies end of particular case & causes exit. Default is optional case, it will execute if expression doesn't match any of the other cases.

3. conditional operator statement

Java has unusual operator, which is combination of ? and : & takes three operands. Popularly called as conditional operator.

```
condition expression ? expression 1 : expression 2
```

Example: (num > 0) ? Positive : negative

<u>**Decision Making : Looping Statements**</u>

used to perform a task repetitively. A loop consist of control statements & body of loop. Every loop will have initialization of counter, execution of statement, testing condition & increment counter.

1. while statement - entry controlled loop

```
Initialization;
while(test condition) {
        Body of loop
}
```

2. do statement - exit controlled loop

```
Initialization;
do {
        Body of loop
} while (test condition;
```

3. for statement

Type 1
for(initialization; test condition; increment) {
        Body of loop;
}
	Type 2 : enhanced
	for(type identifier : expression) {
	        Body of loop;
	}

Arrays & its methods

An array is homogeneous group of contiguous or related data items that share common name. Creation of arrays involves three types.

1. Declaring array

```
type arrayname[];
        or
type [] arrayname;
```

Example: int number[];
float[] marks;

Remember we do not enter the size of array in declaration.

2. Creating memory location

```
arrayname = new type[size];
```

Example: number = new int[5];
marks = new float[8];

Combining declaration & creation is also possible -> int number [] = new int[5];

3. Putting values into memory location

These can be achieved through initialization or accepting values from users.

Initialization: int a[] = {1,2,3};

Array Length

In Java, arrays store the allocated size in property named length.

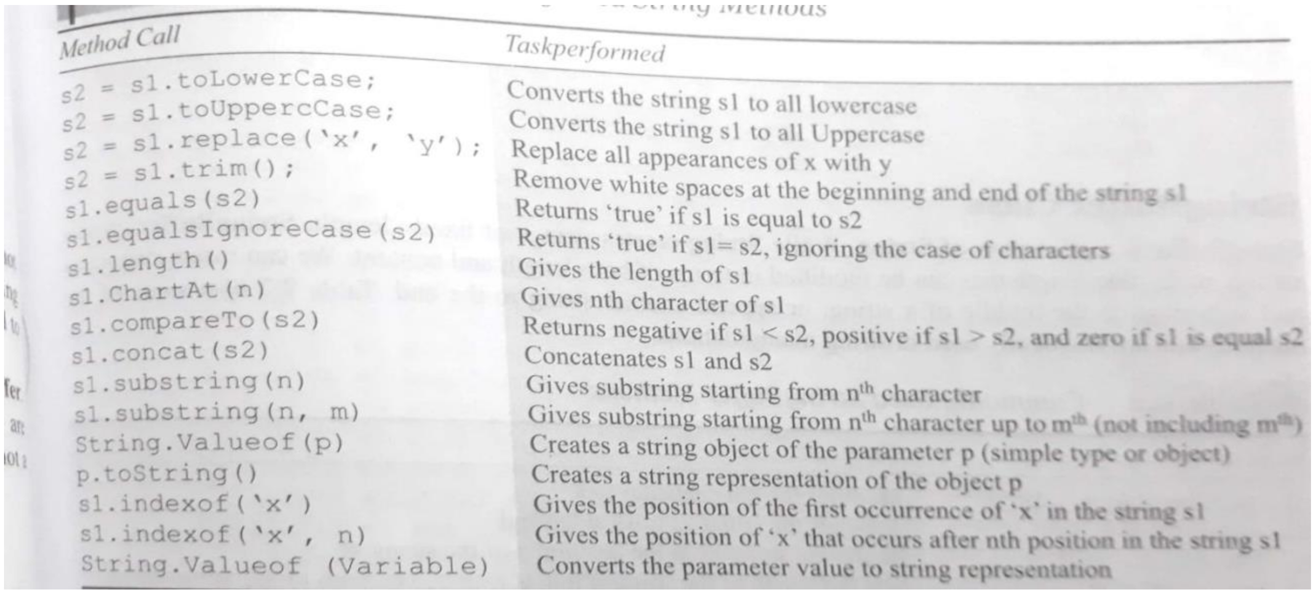
```
int number [] = new int[5];
Int arraysize = number.length //assigns value 5
```

Types of array

1. Single dimension array
2. Multi-dimension array

String Arrays

String class defines number of methods that allow us to accomplish variety of string manipulation tasks as shown below:



Method Call	*Taskperformed*
`s2 = s1.toLowerCase;`	Converts the string s1 to all lowercase
`s2 = s1.toUppercCase;`	Converts the string s1 to all Uppercase
`s2 = s1.replace('x', 'y');`	Replace all appearances of x with y
`s2 = s1.trim();`	Remove white spaces at the beginning and end of the string s1
`s1.equals(s2)`	Returns 'true' if s1 is equal to s2
`s1.equalsIgnoreCase(s2)`	Returns 'true' if s1= s2, ignoring the case of characters
`s1.length()`	Gives the length of s1
`s1.ChartAt(n)`	Gives nth character of s1
`s1.compareTo(s2)`	Returns negative if s1 < s2, positive if s1 > s2, and zero if s1 is equal s2
`s1.concat(s2)`	Concatenates s1 and s2
`s1.substring(n)`	Gives substring starting from nth character
`s1.substring(n, m)`	Gives substring starting from nth character up to mth (not including mth)
`String.Valueof(p)`	Creates a string object of the parameter p (simple type or object)
`p.toString()`	Creates a string representation of the object p
`s1.indexof('x')`	Gives the position of the first occurrence of 'x' in the string s1
`s1.indexof('x', n)`	Gives the position of 'x' that occurs after nth position in the string s1
`String.Valueof (Variable)`	Converts the parameter value to string representation

Command Line Arguments

The parameters that are supplied to the application program at the time of invoking it for execution are command line arguments.

- Consider the following example
  ```
  java Test Hello Hi Hey World
  ```
- Here, Test is name of class & it has been passed with 4 arguments
- main() method is defined to accept array of strings
  ```
  public static void main (String args[])
  ```
- So, args[0] will hold value Test & args[3] will hold value World
- Individual elements of an array are accessed using index or subscript like args[I]
- args.length determines the total number of arguments passed, in our case it will be 3 as indexing starts from 0.