**Java Programming**

**6/6/2022**

**Java**

- is general purpose,

- object oriented,

- high level,

- programming language

**History**

* Developed by : James Goasling & his team members
* 1991 @ Sun micro system
* Initially it was named as “Oak” ( greenish)
* 1995 it was renamed as “Java”
* 2009, oracle corporation acquired java

**Features of java**

* Platform independent language (winodws, macos , linux …)
* Object oriented
* Uses both compiler & interpreter
* Portable
* Scalable
* Slogon : WORA – Write Once Run Anywhere
* freeware

JDK

* Java Development Kit
* It contains bundle software tools
* To write java code
* Ie development environment

JRE

Java Runtime Environment

* Comes along with JDK
* Runtime Environment
* It executes the java code ( byte code)

JVM

Java Virtual Machine

* Abstract machine
* Interpreter , will executes the bytecode – line by line

JVM - interpretor

(filename.class)

Source Code

(filename.java)

Java Compiler

(filename.class)

Byte Code

IDE

Integrated Development Environment

* code editors
* helps to program to write the efficiently & effectively
  + syntax highlighting
  + code alignment
  + code suggestions (auto list features
  + single click – debug, compile & execute
  + code snippets
  + etc
* Netbeans \*
* Eclipse \*
* IntelliJIDEA \*
* VS code
* JCreator

….

Version history of jdk

Jdk 1.1

Jdk 1.2

…..

J2SE 5(jdk 1.5) (Java 2 Standard Edition)

…

…

JSE (Java Standard Edition – jdk 1.8)

….

JSE 18 ( jdk1.18)

Download & Install

* [www.oracle.com](http://www.oracle.com)
* Download & install the appropriate version

Download the Netbeans from – apache.org

Netbeans 8.2 , or netbeans 13

**Basic Programming Concepts**

**Identifier**

* User defined name, given to a variable, class, method, interface
* Identification purpose
* Should be meaningful

Ex : firstname, lastname, email, address, mobile 🡪 Valid names

x, y, ab 🡪 invalid names

**Keywords**

* Also called reserved words
* It has pre-defined meaning
* Cannot change that meaning

Example

break to stop the execution

continue continue the execution

int whole number

float decimal number

….

Variables

* Identifier ( user defined name given to the memory space)
* To store the values into the memory
* For easy access

Rules to be followed:

* Names must starts with letter or underscore, and followed by alphabets, numbers
* Max length – 32 chars
* Don’t use keywords
* Don’t use spaces
* Case sensitive ( AGE, age – 2 different names)
* Don’t use special characters except underscore

Examples:

first\_name, \_lastname, emai\_id etc

**Basic syntax to declare the variable**

**Datatype variablename1, variablename2 …. Variable n;**

Example:

String firstname,lastname,email, address;

int postal\_code;

float mark;

char grade;

**Data Type**

* Specifies the type of data to be stored with the variables

\* Primitive data types

\* non primitive data types

**Primitive types**

int 4 bytes – whole number , +ve / -ve ex: 12345678 , -456656

short 2 bytes “ “ “” 1234

long 8 bytes “ “ “ “ 9894610563

float 4 bytes - decimal number 96.35

double 8 bytes – decimal 651237.89

char 1byte , any character denoted within single quote ‘A’ ‘\*’, ‘5’

byte binary/byte (unreadable) 0 /1

boolean true / false

**non primitive types**

String group of characters denoted withing double quotes “rahul”, “info123@gmail.com”

Object student, person, employee, product

Product: productid – int, productname – string, price – double

**Comments**

* Additional information about your code
* This is only for the user’s reference
* It will not be executed by the java
* // Single line comments
* /\* multiple

Lines of

Comments

\*/

**Basic output statement**

**System.out.println(“msg to be displayed”);**

**System.out.println(variablenames);**

Example:

System.out.println(“Hi Rahul, Welcome to Cispro”);

System.out.println(“Happy Coding “);

**Get Input From user**

1. Import the package : import java.util.Scanner;
2. Create an object for the scanner class : Scanner scan = new Scanner(System.in);
3. Read the inputs
   1. next() - reads the string value
   2. nextInt() - reads the integer value
   3. nextFloat() - reads float
   4. nextDoulbe(), nextBoolean(), nextByte() …..

**Escape Sequences**

-set of characters used to format the output

\n new line

\t tab position

\\ \

\’ ‘

\” “

**Operators**

* operator is a symbol which perform a particular task

assignment operator =

arithmetic operators + - \* / %

relational/comparison < <= > >= == !=

logical operators && || !

increment ++

decrement --

conditional operator ?:

dot operator .

**Type Casting**

* type conversion
* convert the variables from one type to another
* implicit conversion ( Java compiler converts automatically from one type to another)
* explicit conversion ( have to convert from one type to another manually, using appropriate conversion methods or cast with the type)

**Conditional Statements**

* the statements which is used to take decisions based on the condition(s)

\* if statement

\* switch-case

**If statement**

* if the specified condition is true/valid then do some task else then do some other task
  + simple if (true part only)
  + if-else ( true & false – two way branching)
  + else if ladder ( multi branching – multiple conditions)
  + nested if ( if statement within another if statement)

**if** (condition)

{

True part;

}

**else**

{

False part;

}

**switch-case**

-choices based decision making

**switch**(expression)

{

**case** val1:

statement;

break;

**case** val2:

statement;

break;

…..

……

**default**:

Statement;

}

Practice:

. get the number from user, and print the relevant month name

. get the number from user, and print the relevant weekday name

**Control structure/statement**

To control the iteration/looping based on the given condition

\*break ( stops the current iteration and terminates the loop)

\* continue ( stops the current iteration, and it will continue from the next iteration)

Practice:

* print the multiplication table
  + Enter the table number : 5
  + Enter the range to print: 50

1 \* 5 = 5

2 \* 5 = 10

3 \* 5 = 15

4 \* 5 = 20

5 \* 5 = 25

….

50 \* 5 = 250

Mark Statement

----------------------

Name : xxxxx Class : xxxxx

Regno : 0000 Specialization: xxxx

---------------------------------------------------------------------------------------------------------

Subject marks result

---------------------------------------------------------------------------------------------------------

Python 35 Fail

Java 89 Pass

Maths 66 Pass

---------------------------------------------------------------------------------------------------------

Total 220

---------------------------------------------------------------------------------------------------------

**Arrays**

* Is a derived type
* It is a variable used to store group of values in the same name
* The values must be similar type
* Ie group of similar values which shares the common name
* The values in an array is called – elements
* Elements can be accessed using its position / index
* Index always starts with 0
* Elements will be stored in the form of rows & columns

Types:

* + One dimensional array
  + Multi-dimensional array

**One dimensional array**

* An array contains only one dimension
* Ie array with one row and many columns

Syntax:

**Datatype [ ]arrayname = new datatype[column-size];**

Initializing the array elements

|  |  |  |
| --- | --- | --- |
| **50** | **78** | **45** |
| **Index 0** | **1** | **2** |

**Multi-dimensional array**

* An array contains more than one dimensions
* Ie array with more than one rows & columns
* ex : matrix

**Datatype [ ][ ] arrayname = new datatype[row-size][column-size];**

example:

|  |  |
| --- | --- |
| **10** | **20** |
| **30** | **40** |

Practice:

Matrix addition

1. 2 + 5 6 = 6 8

3 4 7 8 10 12

**Strings in Java**

* String is a group of characters denoted within double quotes
* It is immutable ( once the string is created, it cannot be changed/modified)
* String can be created in 2 ways
  + String str = “string value”; ( String literal)
  + String str2 = new String(“String value); (Object type)

**Naming Convention**

**Ex: Welcome to java**

Variable (lowercase) welcome\_to\_java / welcometojava / \_welcometojava

Package(lowercase) welcometojava / welcome.to.java

Class (TitleCase) WelcomeToJava

Interface (titleCase) WelcomeToJava

Method(camelCase) welcomeToJava

Final variable(Uppercase) WELCOMETOJAVA/\_WELCOMETOJAVA/WELCOME\_TO\_JAVA

**StringBuffer & StringBuilder**

import java.lang.\* ( no need import)

* Both classes are used to create the mutable string

|  |  |
| --- | --- |
| StringBuffer | StringBuilder |
| Synchronized | Non-synchronized |
| Thread Safe  Multiple threads can’t access a string buffer | Thread unsafe  Multiple threads can access the value |
| Slower | Faster |
| Mutable | mutable |
| Introduced in 1.0 | Introduced in java 5 (jdk1.5) |

**String methods:**

* Built-in methods to manipulate the strings
  + Length() - length of the string
  + Concat() - join 2 strings
  + startsWith() – check whether the string is starts with the specified characters
  + endsWith() – check whether the string is ends with the specified characters
  + equls() - compare 2 strings which are equal or not ( with case sensitive)
  + equalsIgnoreCase() – compare 2 strings which are equal or not (without case)
  + toUpperCase() – converts the string into uppercase
  + toLowerCase() – converts the string into lowercase
  + split() - split and extract the string values based on the delimeter
  + contains() – check whether string contains the given string
  + charAt() - returns the character in a string at the specified index
  + indexOf() - returns the index of the given character in a string
  + lastIndexOf() – returns the last index of the given char in a string
  + trim() - removes the extra spaces in leading & ending of the string

User Defined Method

* small unit of code, which performs a particular task and returns a value
* ie large program is divided into small programs
* it may or may not take arguments
* arguments – value which is passed to the function as input

**Benefits of methods:**

* code reusability ( no need to write the same code again and again. We can call it whenever required)
* program become structured one, so that we can easily understand the flow of logic
* debugging is easy

**Syntax:**

Access\_specifier return\_type method\_name ( [args ])

{

// body of method

}

**Access Specifier:**

* also called access modifier
* specifies the rights to accessing the data , ie scope of the method
  + private
  + public
  + protected
* return type : type of data to be returned by the method ( primitive, non-primitive)

example:

public void display()

{

System.out.println(“Welcome to cispro “);

}

public int add(int x, int y)

{

return x+y;

}

**Method calling:**

* just use the method name followed by ( ), and pass arguments if required inside ( )
* display()
* int res = add(3,4);

practice:

* calculate the simple interest ( pnr/100)

**Object Oriented Programming Concepts ( OOPs)**

* java is a pure object oriented programming language,
* each and everything is treated as object

benefits:

* + code reuse
  + data safe
  + logically group the data

oops concepts

* object
* class
* data abstraction
* data encapsulation
* inheritance
* polymorphism

**object:**

* real world entity, that may be a person, place, bird, animal …. ( living / non-living things)
* each object has its own behavior/functionality

example:

peacock - dancing

dog - barking

lion - roaring

person - doctor, engineer, farmer, programmer….

**Class:**

* collection / group of similar objects
* a class can have n number of objects
* ie objects and its functionalities which are tied together
* class is a type , blue print

example:

Birds --- class

Peacock

Dove

Parrot --- objects

**Syntax to defined the class :**

Access\_specifier class ClassName

{

// constructor

// fields ( variables, constants)

//methods

}

Example

public class Products

{

int product\_code;

String product\_name;

flaot price;

public void display()

{

System.out.println(“User defined method”);

}

}

**Create object / Instantiation**

* to access the properties/members of the class, we have to create object. By using object only we can able to access the class members
* object is also called as : instance

**Syntax:**

Classname instance/object name = new constructor();

|

Class name

Example:

Products prod1= new Products();

**Accessing class members:**

Prod1.product\_code = 101;

Prod1.product\_name = “Cinthol”;

Prod1.price = 45.00;

Prod1.display();

Practice:

Create an employee class with the fields : eid, ename, designation, salary

* get & print the details using methods

Data Abstraction

* specifies the essential feature of an object not the full details
* example : chair
* abstract, interface keywords are used

Data Encapsulation:

* capsule
* hiding the data into the class ( wrapping the data into the class)
* using the access specifier
* getter & setter : set : used to assign the value, get – used to get the value

this

* Is a keyword
* Denotes the current instance/ object