Java Basics

Comments in Java:

-Explain the code & make it more readable

-prevent execution of code

two types:

1. Single-line Comments- two forward slashes

Ex: // single line comment

2.Multi-line Comments - /* ..... */

Ex: /* multi

line

comments */

Data Types in java:

Data types represents the different types of values to be variable.

two types:

1. Primitive data types - built-in datatypes

8 - primitive datatypes

- byte

byte num = 123;

byte num = 334; // gives me error

- short

short num = 22345;

short num = 10000000; // gives an error

- int

int num = 123456;

- long

long num = 123456789L;

- float

float num = 12.45f;

float num = 14.123456789f; // stores 14.1234567

- double

double num = 34.678d;

double num = 14.12345678912d; // stores 14.12345678912

- boolean

boolean status = true;

boolean isComplete = false;

- char

char grade = 'A';

2.Non-Primitive data types - user defined datatypes(Except String)

String

Arrays

Classes

Ex: String name = "NNRG";

Variables:

variable is a name of a temporary memory location.

Datatype variable-name = value;

(or)

Datatype Variable_name;

Variable_name = value;

Examples:

int myNum = 521;

float salary = 1234.56f;

or

int myNum;

myNum = 123;

Print / display variable values:

println() - used to display variable value on the screen.

println(myNum)

+ used to combine text and variable value.

Rules for naming Variables:

- Contains Letters, digits, _ , $
- Must begin with letters
- Names should begin with lower case letter.
- Names are case sensitive (i.e myNum != mynum)
- don't use keywords(class, int , float, ...) as variable names

- use Descriptive variable names

Exmaple:

```
int a = 521;

int rollNo = 521;
int minutePerHour = 60;
```

Java supports 3 type of variables:

1. Local Variable
   -variable which is declared inside the methods
2. Instance Variable
   - variable which is declared inside the class but outside the method.
3. Static(or Class) Variable
   - variable which is declared using static keyword.

- which cannot be local(i.e declared inside the class but outside the method.)

Type Casting and Type Conversion:

used to convert data from one data type to another data type.

Type Casting:

-When a data type is converted into another data type by the programmer.

- type casting () operator

- Higher data type to lower data type.

Example :

```
float x = 3.6f; // 4bytes
short  y = x; // gives an error // 2bytes
short y = (short)x; // it stores 3
```

Type Conversion:

- if a data type is automatically converted into another data type.

- Compiler

- Lower data type to higher data type

Example:

short x = 4;

float y = x; // stores as 4.0

Operators in JAVA:

It is Symbol which is used to perform an opertion on variables and values.

Java Supprots following Operators:

-Arithmetic Operators:

+ , - , * , / , %, ++ , --

- Assignment Operators:

= , += , -= , *= , /=, %=

Ex:

a+=10;

i.e a = a + 10;

- Comparison/ Relational Operators:

== , != , > , >= , < , <=

- Logical Operators:

&& , || , !

- Bitwise Operators:

& , | , ^ , << , >>

- Conditional Operator:

(Expression)? value1 : value2;

Expressions:

An expression is consists of variables , operators , literals and method calls.

- all the expressions written based on the syntax of programming language.

Example:

```
int intMarks = 21;
int extMarks = 56;
int totalMarks = intMarks + extMarks;
```

Example:

```
float x =10, y = 20;
float z;
 z = x + y / 100; // z value 10.2
 z = (x+y) / 100; // z value 0.3 // recommended expression
```

Control Statements in java:

used to control the flow of execution of the program.

Java Control Statements:

-Conditional Statements

-Loop Statements

-Jump Statements

Conditional Statements:

used to execute a block of code based the condition.

if True : one block of code

False : another block code

The conditional Statements are:

- if statement
- if - else statement
- if - else - if statement
- nested - if statement
- switch - case statement

- if statement:

Syntax:

```
if ( condition )
{
// statements
}
```

- if - else Statement:

Syntax:

```
if ( condition )
{
// statements -1
}
```

```
else
{
// statements -2
}
```

- if - else - if statement:

Syntax:

```
if ( condition1 )
{
// statement -1;
}
else if ( condition2)
{
// statements -2;
}
. . .
else
{
// statements -n;
}
```

- nested - if statement:

Syntax:

```
if ( condition1 )
{
if ( condition2 )
```

```
{
// statements
}
else
{
// statements
}
}
else
{
// Statements
}
```

-Switch - case statement:

Syntax:

```
switch(expression)
{
case value1: Statements;
break;
case value2: Statements;
break;
. . .
default: Statements;
}
```

Loop Statements :

execute block of code multiple times.

Java supports following loop statements:

- while loop
- do - while loop
- for loop

while loop:

- if the number of iterations are not fixed.

Syntax:

```
while ( condition )
{
// statements;
}
```

do - while loop:

- if the number of iterations are not fixed.
- execute the statements atleast once.

Syntax:

```
do
```

{

// statements;

} while(condition);

for loop:

- if the number of iterations are fixed.

Syntax:

for (initialization ; condition ; inc / dec)

{

// statements;

}

for - each loop:

-used to read/ access the values from an array.

Syntax:

for(datatype var_name : array_name)

{

// statements

}

Jump Statements:

- break - immediately jumps to end of the loop(stops the execution of the loop)

Syntax:

break;

- continue - immediately jumps to next interation

Syntax:

contiue;

Arrays:

int a = 10;

Special Data Structure called Arrays

Array is collection of similar data items.

index - starts from 0 to n-1.

where n is number of elements in an array.

Example:

Array Elements:	12	34	56	67	89
index:	0	1	2	3	4

Two types of Arrays:

1. Single Dimensional Array

- which is an array with one dimension or index.

- look like single row or column.

Syntax:

// Declaration and allocating memory

Datatype array_name[] = new DataType[size];

(or)

Datatype array_name[];

array_name = new Datatype[size];

// Declaration and Initilization

Datatype array_name[] = {val1,val2,val3,.....valn};

2.Multi Dimensional Array

- with two or more diamensions or indexes.
- look like matrix of rows and cloumns

Lets take a two - dimensional array:

rows and columns

Syntax:

Datatype arr_name[][] = new Datatype[rows][cols];

(or)

Datatype arr_name[][];

arr_name = new Datatype[rows][cols];

(or)

Datatype arr_name[][] = { {1,2,3} ,{4,5,6} }

example:

// declaring and initializing values to an array

int marks[][] = {{45,56,25,78},{89,57,45,78}};

Classes and Objects:

- building blocks of object - oriented programming language.

Class:

A Class is a blueprint or prototype that defines the variables(attributes) and methods(functionalities) common to all objects.

- use keyword called "class"

format :

class class_name

{

// variables declaration

```
// methods declaration
}
```

Object:

- An Object is software bundle of variables and methods.

- An object is instance of a class. by using this instance you are able to access varaibles and methos in the class.

format:

```
class_name obj_name = new class_name();

type var_name;
class_name obj_name;
```

Methods:

-Method is a collection of statements that are grouped together to perform an action.

- None of the methods declared outside the class

- All methods name must be starts with lower-case letter.

- make code reusable

- simplicity the code

Syntax:

```
[modifiers]  return_type method_name(parameters)
```

{

// statements;

}

Modifiers(optional) : public, protected,default or private

return_type : void (it doesn't returns any thing) or any datatype(int,float, String...)

method_name: valid identifier as method name(lower case letter).

parameters : pass one or more values to method

method_body: used perform an action.

String Handling:

What is String:

-In general, A string is a sequence of characters.

- But in Java, String is an Object that represents a sequence of characters.

- using java.lang.String class.

- An array of characters works same as Java String

How to Create Strings:

two ways to strings in java

-

String var_name = "abc"; // by string literal

(or)

String obj_name = new String("abc"); // by using new keyword

Example:

String s = "Welcome";

String s1 = "Welcome"; // will not create new instance

s1 = "Bye";

(or)

String s = new String("Welcome");

String s1 = new String("Welcome");

// both s ans s1 have separate memory

Methods to handle strings in java:

- charAt():

-returns a character value at the given index.

- index starts from 0 upto n-1(where n is length of string)

Syntax:

charAt(int index)

Example:

String str = "Java Program";

char ch = str.charAt(6);

s.o.p(ch)

o/p: r

-indexOf():

-returns an index value of the given character.

- if char was not found, returns -1.

Syntax:

indexOf(char ch)

or

indexOf(char ch , int fromindex)

Example:

String str = "Java Program";

int index = str.indexOf('a');

int index1 = str.indexOf('a',5);

s.o.p(index)

s.o.p(index1)

o/p: 1

10

-concat():

-used to combine two strings

Syntax:

str1.concat(str2)

Example:

String s1 = "Java";

String s2 = "Programming";

String s3 = s1.concat(s2);

S.o.p(s3);

O/p:

JavaProgramming

-contains():

- used to search for the sequence of characters in the given string
- True : Found in the string
- False : Not Found in string

Syntax:

str.contains(chars)

Example:

String rollno = "197Z1A0557";

S.o.p(rollno.contains("7Z"));

S.o.p(rollno.contains("7P"));

O/p:

true

false

- equals():

Used to compare two strings

-True : Both Strings are equal

-False : Both are not equal

Syntax:

str1.equals(str2)

Example:

```
String s1 = "Java";
String s2 = "JAVA";
String s3 = "Java";

S.o.p(s1.equals(s2));
S.o.p(s1.equals(s3));
```

O/p: false

true

-length():

-Used to find length of the given string.

-It returns number of characters in the given string.

Syntax:

str.length()

Example:

```
String s1 = "Java";
String s2 = "Pro gram";
S.o.p(s1.length())
```

S.o.p(s2.length())

O/p: 4

8

-substring():

- Used to extract sub string from the given string.

Syntax:

str.substring(start_index)

(or)

str.substring(start_index,end_index)

Example:

String s = "Java Program";

S.o.p(s.substring(5));

S.o.p(s.substring(5,8));

S.o.p(s.substring(0,4));

O/p:

Program

Prog

Java

- replace():

-Used to replace old set of characters with a new set of characters

in the given string.

Syntax:

str.replace(old_char , new_char)

Example:

```
String s = "Java Program";
S.o.p(s.replace('a','s'));
S.o.p(s.replace('Java','Python'));
S.o.p(s);
```

O/p:

```
Jsvs Progrsm
Python Program
Java Program
```

-toLowerCase():

- used to convert given string into lower case letters

Syntax:

str.toLowerCase()

Example:

```
String s1 = "Java";
String s2 = "PROGRAM";
S.o.p(s1.toLowerCase());
S.o.p(s2.toLowerCase());
```

O/p: java

program

-toUpperCase():

- used to convert given string into upper case letters

Syntax:

str.toUpperCase()

Example:

```
String s1 = "Java";
String s2 = "program";
S.o.p(s1.toUpperCase());
S.o.p(s2.toUpperCase());
```

O/p: JAVA

PROGRAM

---

super Keyword:

Used to access properties(Variables / methods) of parent class from child class.

i.e

- used to access the variables in parent class

- used to access methods in parent class.

---

final Keyword:

Used to restrict the user.

Can be appiled on variables, methods and classes.

- The final keyword can be appiled on variables to declare constants.

- The final keyword can be appiled on methods to disallow method overridding

- The final keyword can be appiled on classes to prevent inheritance

Final Varibale(Constant):

You can cahnge the value final variable.

Final Method:

You cannot override that method.

Final Class:

You cannot extend it.

In simple words:

- Stop value change
- Stop method overriding
- Stop Inheritance

Polymorphism:

perform single action in different ways

poly: many

morphs: forms

two types of polymorphisms:

1. Compile time - method overloading(single class)
2. RunTime - method Overriding(parent and child classes)

Method Overloading:

if a class have multiple methods with same name but different parameters

is known as method overloading.

There are two different ways of method overloading

- By changing datatypes of parameters
- By changing number of paremeters

Method Overriding:

If a child class has the same method as declared in the parent class.

Rules for method overriding:

- Methods must have same names
- Methods must have same method parameters
- IS-A relation

Binding:

It is the process of connecting a method call to its body.

2 types of binding:

-Static(early) Binding - at compile time when we overload methods

-Dynamic(late) - at runtime when we override methods

Abstract Class:

What is Abstraction?

-Abstraction is the process of hiding the implemenation details and showing simple functionalities to the user.

-In other words, it shows imporant things to the user and hides the internal details.

Def: A class that is declared with abstract keyword is known as an abstract class. and it contains one or more abstract methods.

An abstract class can have abstract and non abstract methods.

What is Abstract Method?

A method that is declared as abstract and it doesn't have implementation is known as abstract method.

(or)

An abstract method is a method that is declared with no body or implementation.

-Example for abstract methods:

```
abstract void run();

abstract void display();
```

-Example for Abstract Class:

```
abstract class Bike
{
        abstract void run();
}
```

Package:

A package is a group of classes,interfaces and sub-packages.

- Packages are used in java, in-order to avoid naming conflicts and to control access of classes.

- Packages are categorized into 2 types

  - Built-in Packages
  - User-defined package

-Built-in Packages:

There are many built-in packages such as java,lang,out,javax, swing,net,io,util,sql and etc.

-User-defined packages:

In Java, User able to create own packages by using the keyword "package".

format:

package package-name;

Example:

package mypack;

- To compile

javac -d directory filename

In the above syntax, -d specifies "directory" where to place generated class files.

Ex: D:/madhu

If you want to keep class files in the same directory you need to use (.) dot

- To Run / execute

java packagename.classname

You need to use fully qualified name.

Note:

The package declaration must be first statement in the program.

Importing packages:

The "import" keyword is used to import built-in and user-defined packages into your java source file.

By importing packages,you are able to use classes from one package to another package.

i.e, the "import" statement is used to make classes and interfaces of another package to the current package.

There are 3 ways to import packages

- using fully qualified name
- using packagename.classname(import only class you want to use)
- using packagename.* (import all classes from the package)

- using fully qualified name:

Fully qualified name is nothing but "packagename.classname".

If you want to use "fully qualified name" no need to import the package.

But you need use "fully qualified name" every time when you are accessing the class.

- using packagename.classname(import only class you want to use)

If you import "packagename.classname" then only specified class of this package will be accessible.

i.e, In this, we can import only class that you want to use.

- using packagename.* (import all classes from the package)

If you use "packagename.* " then all the classes and interfaces of this package will be accessible but not sub packages.

Note:

If you import package, sub packages will not be imported.

i.e. if you import a package, all the classes and interfaces of that package will be importe excluding the classes and interfaces of sub-packages.

Note: sequence of a program

- package statement
- import statement
- class statement

.

.

Rule:

There can be only one public class in java program and it must be saved by the public class name.

Subpackage:

- package inside the package is called as sub package.

The standard of defining subpackage is "packagename.subpackagename"

Example:

pack.subpack

Setting CLASSPATH:

-If you want to save java source files and class files in different directories then we need to set "CLASSPATH" to run (or) execute those class files.

- To compile

javac -d c:\classes s.java

- To Run

To run you need to set CLASSPATH of directory where the class file resider.

set classpath = c:\classes;.;

java pack.s

Access Modifiers (OR) Member Access Rules:

The Access modifiers in java specifies the accessibility(Scope) of data members, methods, constructors or classes.

Java supports following access modifiers:

those are:

1. private - With in the class only
2. default - Inside the package only
3. protected - Inside the package and outside the package but through Inheritance
   - Not applied on the Classes
4. public - Everywhere

Enumerated type:

An Enumerated (enum) type is a special data type that contains

fixed set of contstants.

In Java, "enum" keyword is used to create/define Enumerated type.

- create / define enumerated types any time when you need to represent a fixed set of constants.

Example:

```
public enum Directions {
        EAST,WEST,SOUTH,NORTH
        }
```

Scanner Class:

-The Scanner class is used to get / read input from user(keyboard).

-This class comes under "java.util" package.

-This Class uses "System.in" object to get input from keyboard.

-This class have following methods

- nextLine() --- reads input as string
- nextInt() --- reads input as Integer

- nextFloat() --- reads input as Float

- nextLong() --- reads input as Long

- nextShort() --- reads input as Short

- nextDouble() --- reads input as Double

Console input and output (or) Console Class:

The Console Class provides convenient methods for reading input and writing output.

- This class is comes under "java.io" package.

- This class uses "System" Class to create an object.

- This class provides the following methods

    - printf() --- writes / prints something on screen.

    - readLine() --- Reads input from keyboard as string.

Constructors in Java:

Constructor is a special type of method that is used to initialize the object.

i.e, it provides the data for the object.

- Constructors are the methods which are used to initialize objects.

- Constructors are invokes at the time of object creation.

- There are Two rules defined for the constructor

    - Constructor name must be same as it's Class name.
    - Constructor must have no explict return type.

Java Supports two types of constructors

    - Default Constructor
    - Parameterized Constructor

Default Constructor :

A constructor that have no parameters is known as default constructor. This is used to provide default values to an object.

Parameterized Constructor :

A constructor that have parameters is known as Parameterized Constructor. This is used to provide different values to the distinct objects.

Constructor Overloading:

Constructor Overloading is a technique in which a class can have any number of constructors with different parameter lists.

The compiler differentiates those constructors by number and type of parameters.

Interface:

An interface is a collection of abstract methods which are in public scope.

(or)

It is a collection of methods,which are public and abstract by default.

What is Abstract Method?

A method that is declared as abstract and it doesn't have implementation is known as abstract method.

(or)

An abstract method is a method that is declared with no body or implementation.

-Example for abstract methods:

```
abstract void run();

abstract void display();
```

- Java doesn't supports multiple inheritance among the classes but

interfaces allow java to support this feature.

- i.e, In Java a class can inherits any number of interfaces

(Multiple Inheritance).

- Note: In interfaces none of the methods have body.

- Interfaces are declared with the help of keyword "interface".

Syntax:

```
interface interface_name
{
        return_type methodname(arguments);
        -
        -
}
```

- In simple words, interface is a blue print of a class, it has static

constants and abstract methods.

- The Interface in java

    - to achieve abstraction

    - to achieve multiple inheritance

- In java,

    - a class extends another class

    - an interface extends another interface

    but

    - a class implements an interface.

Extending interfaces (or) Inheritance in interfaces :

- Like inheritance in classes, the interfaces can also be extended.

- An interface can inherit another interface using the keyword "extends".

Multiple inheritance by Interface :

If a class implements multiple interfaces (or) an interface extends multiple interfaces is known as Multiple inheritance.