• Datatypes

• Flow controls

• Arrays

• Class

• Methods

• Methods arguments

• Methods arguments and return types

• Static

• Static block

• Instance

• Instance block

• Public

• Default

• Private

• Protected

• Final

• Object (tostring)

• Hashcode

• Getclass

• Encapsulation

• Setter initialization

• Constructor

• Constructor initialization

• Default constructor and argument constructor

• String

• String methods

• Package

• New

• This

• Super

• Inheritance

• Single level

• Multilevel

• Hierarchical

• Multiple

• Hybrid

• Overloading : (Access modifiers---private,Static,Final,Return Arguments .)

• Overriding : (Access modifiers---private,Static,Final,Return Arguments .)

1. Data Types:

Collection of information is called data

Data types are two types

1. Primitive data types
2. Non- primitive data types

Primitive data types: it is already predefined in java. Byte, short, int, long, float,double,char,boolean

Non-primitive data types : non- primitive data types are created by the programmer.

It includes String,classes and arrays.

Byte: byte is a java keyword is used to declare a variable that can hold 8 bit data values.

Short: short is a java keyword is used to decalre a variable that can hold a 16 bit integer.

Int: int is a java keyword is used to declare a variable that can hold a 32-bit signed integer.

Long: long is a java keyword is used to declare a variable that can hold a 64-bit integer.

Float: float is a java keyword is used to declare a variable that can hold a 32-bit floating point

number.

Double : double is a java keyword is used to declare a variable that can hold 64-bit floating point

Number

Byte //range is 128 to -128

Short //range 32000 to -32000

Int // medium range values

Long // numeric values will be larger

Float //when cal don’t required more than 6 to 7 digits

Double // exact value

Char: char is used to store single character, any single character(alphabet/digit/special character) we

can use char data type.

2.Flow controls :

Control flow statements are

Decising naking statements

1. If statement
2. If-else statement
3. If-else-if statement
4. Nested if statement

Switch statement

Looping statements

1. For loop
2. While loop
3. Do-while loop
4. For-each loop

Jumping statements

1. Break statement
2. Continue statement

If statement: it is simple decision-making statement, if certain condition is true then a block of

statements executed otherwise not.

Example:

public class Student {

public static void main(String[] args){

int age = 18;

if(age >= 12 && age < 20){

System.out.println(“Eligible to vote”);

}

}

}

If-else statement: if the condition is true, if block of statements will be executed.if the condition is

false else block of statements will be executed.

Example :

Public class Student {

Public static void main(String[] args) {

int age = 18;

if(age >= 15){

System.out.println(“person is eligible to vote”);

} else {

System.out.println(“not eligible to vote”);

}

}

}

If-else-if statement: it executes one condition from multiple statements.

Example:

public class Student {

public static void main(String[] args) {

Int marks = 98;

If(marks >= 35 && marks <= 45) {

System.out.println(“pass”);

}else if(marks >= 55 && marks <= 65) {

System.out.println(“c grade”);

}else if(marks >= 75 && marks <= 85 ) {

System.out.println(“b grade”);

}else {

System.out.println(“Topper”);

}

}

}

Nested if statement:

It is always legal to nest if-else statements which means we can use one if or else if statement inside

another if or else if statement.

Example:

**if**(**true**) {

System.***out***.println("first name");

**if**(**true**) {

System.***out***.println("enter second name");

}

**else** {

System.***out***.println("enter third name");

}

} **else** {

System.***out***.println("enter last name");

}

Switch statement: The java switch statement executes one statement from multiple conditions. It is

like if-else-if ladder statement. switch is not applicable for float, Boolean, double and long data types.

Mostly int data type used in switch statement

Example:

Public class Student {

Public static void main(String[] args) {

Int day = 4;

switch(day){

case 1:

System.out.println(“Monday”);

Break;

case 2:

System.out.println(“Tuesday”);

Break;

case 3:

System.out.println(“wednesday”);

Break;

case 4:

System.out.println(“Thursday”);

Break;

case 5:

System.out.println(“Friday”);

Break;

case 6:

System.out.println(“Saturday”);

Break;

default:

System.out.println(“invalid code”);

}

}

}

Looping statements:

For loop:

the java for loop is used to iterate a part of several times, if the number of iterations is fixed, it is

recommended to use for loop

for loop consists of

1. Initialization
2. Condition
3. Increment/decrement

Example:

public class Student {

public static void main (String[] args) {

int[] a = {12, 15, 17, 25, 45};

for(int i = 0; i < a.length; i++) {

System.out.println(a[i]);

}

}

}

While loop: the java while loop is used to iterate a part of program repeatedly until the specific

Boolean condition is true. If the number of iterations is not fixed it is recommended to use while loop

Example :

public class Student {

public static void main(String[] args) {

int num = 1;

while (num <= 10) {

System.out.println(num);

num = num + 1;

}

}

}

Do while loop:

It is used to execute block of statements continuously until the given condition is true

Or

Do while loop in java executes the statements at least once even the condition is not satisfied.

Example:

Int x = 1;

do {

System.out.println(x);

x = x + 1;

}while(x <= 10);

For-each :

In Java, the for-each loop is used to iterate through elements of arrays and collections (like ArrayList).

It is also known as the enhanced for loop.

Example:

public class Student {

public static void main(String[] args) {

int[] marks = {45, 49, 58, 78, 98, 98};

for(int i : marks ){

System.out.println(“marks “ + i);

}

}

}

Break:

Java break keyword is used to break the loop or switch statement. It breaks the current flow of the

program at specific conditions.

3.Array:

Array is an object which contains elements of similar data type.

Array index starts from zero.

Arrays are initialized in two ways.

String[] name = {“raju”, “mahesh”, “manu”, “manoj”, “karna”, “mohith”};

int[] a = new int[4];

a[0] = 15;

a[1] = 25;

a[2] = 35;

a[3] = 55;

Example:

String[] name = {“raju”, “mahesh”, “manu”, “manoj”, “karna”, “mohith”};

for(int i = 0; i < name.length; i++) {

System.out.println(name[i]);

}

4.Class :

class is a keyword. A class is nothing but a blueprint, and a collection of objects is called a class.

Example:

public class Student {

public static void main(String[] args) {

}

}

5.Method:

A **method** is a block of code which only runs when it is called. You can pass data, known as

parameters, into a method.

Example:

public class Calc{

int a;

int b;

public void add() {

System.out.println(“addition” + (a + b));

}

Public static void main (String[] args){

Calc c = new Calc();

c.a = 12;

c.b = 15;

c.add();

}

}

6. Method Arguments:

Example - 1:

public class Student {

public void add(int a, int b, String n){

System.out.println(“add : ” + (a + b));

System.out.println(n);

}

public static void main(String[] args) {

Student s = new Student();

s.add(25, 45, “Raju”);

}

}

Example - 2:

public class Student {

public void add(int[] a, int[] b, String[] n){

for(int i = 0; i < a.length; i++) {

System.out.println(a[i]);

}

for(int i = 0; i < b.length; i++) {

System.out.println(b[i]);

}

for(int i = 0; i < n.length; i++) {

System.out.println(n[i]);

}

}

public static void main(String[] args) {

Student s = new Student();

int[] i = {12, 25, 27,28, 35};

int[] j = {25, 29, 35, 38, 57,58};

String[] k = {“ram”, “Mahesh”, “manu”};

s.add(I, j, k);

}

}

7. Methods arguments and return types

Return keyword:

the return keyword stops the execution of a function and returns the desired output. Even if there’s

no value to return, it will use void as a return type.

Example - 1:

public class Student {

int show(int a, int b){

return a + b;

}

public static void main(String[] args){

Student s = new Student();

Int c = s.show(25, 45);

System.out.println(c);

}

}

Example -2:

public class Student {

int[] get(int[] a){

return a;

}

public static void main(String[] args) {

Student s = new Student();

Int[] c = {25, 35, 45, 75, 95};

Int[] d = s.get(c);

for (int i = 0; i < d.length; i++){

System.out.println(d[i]);

}

}

}

Example – 3 user defined class

public class Student {

public int age;

public String name;

}

public class Teacher {

Student getdata(Student x){

return x;

}

public static void main(String[] args) {

Teacher t = new Teacher();

Student s = new Student();

s.age = 35;

s.name = “Raju”;

Student m = t.getdata(s);

System.out.println(m.age + “ “ + m.name);

}

}

Example – 4 user defined Array

public class Student {

public int age;

public String name;

}

Public class Teacher {

Student[] get(Student[] x){

return x;

}

Public static void main(String[] args) {

Teacher t = new Teacher();

Student s = new Student();

s.age = 25;

s.name = “manu”;

Student s1 = new Student();

S1.age = 29;

S1.name = “Mahesh”;

Student[] d = {s, s1};

Student[] m = t.get(d);

for(int i = 0; i < m.length; i++) {

System.out.println(m[i].age );

System.out.println(m[i].name);

}

}

}

8. Static keyword:

Static keyword:

we can apply static keywords with variables, methods and blocks.

No need to create an object for static method calling or static variable

1. For method calling we can use reference of class name or object reference also.
2. The static keyword gets memory only once in the class area at the time of class loading.

Ex :

public class Student{

public static int age = 20; //static variable

public static void show(){ // static method

System.out.println(“age “ + age);

}

public static void main (String[] args){

Student s = new Student();

System.out.println(Student.age);

// System.out.println(s.age);

Student.show(); // method calling

}

}

static variable:

if you declare any variable as static it is known as static variable.

Static method:

if you declare any method as static, it is known as static method.

Static block:

if you declare any block as static, it is known as static block.

What is the use of static keyword in java main method:

The main() method is declared static so that JVM can call it without creating an instance of the class

containing the main() method.

Heap memory :

after creating an object instance variables, instance methods and instance blocks are stored in

heap memory.

Static memory :

when we load the class static variables, static methods and static blocks are stored in Static area it is

allocated separate memory.

Static to static: both are loaded at the time when the class is loading.

Instance to instance : Both will come in to the picture after the object creation only.

Instace to static : by the instance created, object is already loaded.

Static to instance : at the time static loading, it cannot view instance variable or method.

Example: for above definations

package com.cisco;

public class Teacher {

// static to static with variables

public static int age = 25;

public static int id = 25;

//static to static

public static int num = 45;

public static void name() {

System.out.println("static to static " + num);

}

//instance to instance with variables

public int number = 15;

public int number2 = 25;

//instance to instance

public int num1 = 25;

public void show() {

System.out.println("num1 : " + num1);

}

//static to instance

public int num2 = 35;

public static void data() {

Teacher t = new Teacher();

System.out.println(t.num2);

}

//instance to static

public static int num3 = 55;

public void getdata() {

System.out.println("num3 : " + num3);

}

public static void main(String[] args) {

Teacher t = new Teacher();

// static to static with variables

System.out.println("age : " +Teacher.age);

System.out.println("id : " + Teacher.id);

// static to static

Teacher.name();

//instance to instance with variables

System.out.println(t.number);

System.out.println(t.number2);

//instance to instance

t.show();

//static to instance

Teacher.data();

//instance to static

t.getdata();

}

}

9. Access modifier:

Two types of modifiers in java

1. Access modifiers
2. Non- access modifiers

There are four types of access modifiers.

1. Public
2. Default
3. Private
4. Protected

Public:

It can be accessed from within the class, outside the class, within the package and outside the

package.

Default:

The access level of a default modifier is only within the package and within the class.

Private:

The access level of a private modifier is only within the class.

Protected:

The access level of a protected modifier is within the package and outside the package through child

class.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile,

Transient.

10. Final:

The final keyword in java is used to restrict the user.

The java final keyword can be used in many context. Final can be

1. Variable
2. Method
3. Class

Blank final variable:

A final variable that have no value it is called blank final variable or uninitialized final variable.

It can be initialized in the constructor only.

Example :

public class Student {

public final int data;

public Student(){

data = 70;

}

}

Final variable:

If you make any variable as final, you cannot change the value of final variable ( it will be constant).

Example:

public class Student {

public final int data = 90;

public Student(){

data = 150; // cannot change the value of final variable.

}

}

Final method:

If you make any method as final, you cannot override it.

Ex :

public class Student {

public final int data;

public Student(){

data = 70;

}

final void run(){

System.out.println(“ running”);

}

}

public class Teacher extends Student {

final void run(){

System.out.println(“cannot override it”);

}

}

Final class:

If you make any class as final, you cannot extend it.

Static blank final variable:

a static final variable that is not initialized at the time of declaration is known as static blank final

variable. It can be initialized in static block.

Ex :

public class Student {

static final int data;

static {

data = 85;

}

Public static void main (String[] args) {

System.out.println(Student.data)

}

}

Is final method inherited ?

Yes, final method is inherited but you cannot override it.

Example: Ex :

public class Student {

final void run(){

System.out.println(“running”);

}

}

public class Teacher extends Student {

public static void main (String[] args) {

Teacher t = new Teacher();

t.run();

}

}

What is final parameter:

If you declare any parameter as final, you cannot change the value of it.