Variables:-

To store the data

Rules to create a variables:-

1) The variable name should be meaningful

String s = "Raju"
String username = "Raju"

2) The variable name should always start with _ or $ or a alphabet

String *username = "Raju"(Invalid)
String $username = "Raju"(Invalid)
String 100username = "Raju"(Invalid)

Start the variable names with alphabet for better understanding.

C, C++(_, $) ==> 10 years
James Gosling and his team ==> JAVA(1990's)

JAVA ==> $

```
class First
{
    ..........
    ..........
    class Second
    {
        ..........
        ..........
    }
}
```

First.java ——> Java Compiler ——> **First$Second.class**

3) If we create a variable with multiple words, never seperate them space, use underscore symbol

String city_name = "Pune"
String cityName = "Pune"

4) If we create a variable with capital letters, then it is a special variable, we call as CONSTANT.

String NAME = "Raju"

NAME = "Mary"(Not Recommended)

Symbols(=, ==, <, >, +, -, ...) ==> OPERATORS

+, -, *, /, % ========> Arithmetic opertators

System.out.println(10 / 2); 5
System.out.println(10 % 2); 0

<, >, ==, <=, >= ========> Comparision Operators
These oepratros will always give output as true OR false

s.o.p(10 > 5)
s.o.p(10 < 5)
s.o.p(10 == 5) ==> compare L.H.S with R.H.S

=, +=, -=, *=, /=, %= ===========> Assignment Operator

int age = 30;
age = age + 1;
age *= 10;

System.out.println(age);

Increment and Decrement oeprators

++ ==> increment
age++ ==> Post Increment
++age ==> Pre Increment

-- ==> decrement
age-- ==> Post Decrement
--age ==> Pre Decrement

Logical Operator

&&(AND), ||(OR), !(Not)

System.out.println((10 > 20) && (4 == 5));;
System.out.println((10 > 5) || (4 == 5));
System.out.println(!(10 > 5) || (4 == 5));

Ternary Operator

Conditional Statements ==> Statements will get executed based on some condition

marks = 87

s.o.p("A Grade")
s.o.p("B Grade")
s.o.p("C Grade")
s.o.p("Fail")

1) if statement
2) if else statement
3) if elseif else statement

marks = 87

s.o.p("A Grade")
s.o.p("B Grade")
s.o.p("C Grade")
s.o.p("Fail")

```
int marks = 87;

if(marks > 85)
{
    System.out.println("A Grade");
}
```

OR

```
int marks = 87;

if(marks > 85)
    System.out.println("A Grade");
```

```
int marks = 50;

if(marks > 85)
    System.out.println("A Grade");
else if(marks > 60)
    System.out.println("B Grade");
else if(marks > 35)
    System.out.println("C Grade");
else
    System.out.println("Fail");
```

Scanner class

In java there are so many built in classess, already avaiable.

WAP to reverse a string
Raju ==> ujar

reverse()
....
....
....

WAP to count the character spresne tin a string
Raju ==> 4

length()
....
....
....

1000 + functions / methods ===> divide across many java classes

class String { reverse(), count(), ... }
....
....
....

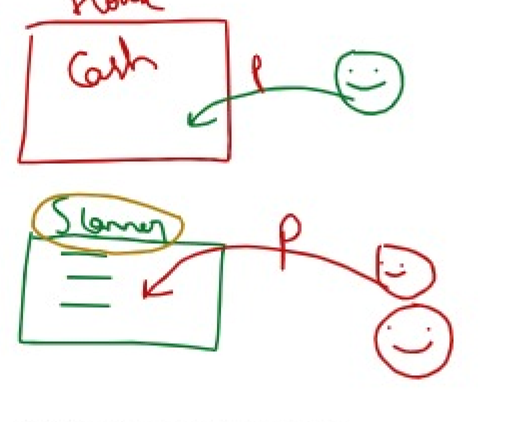
```
class Scanner
{
    1)
    2)
    3)
    .....................
}
```

——> Help to collect the informatioin entered by the user

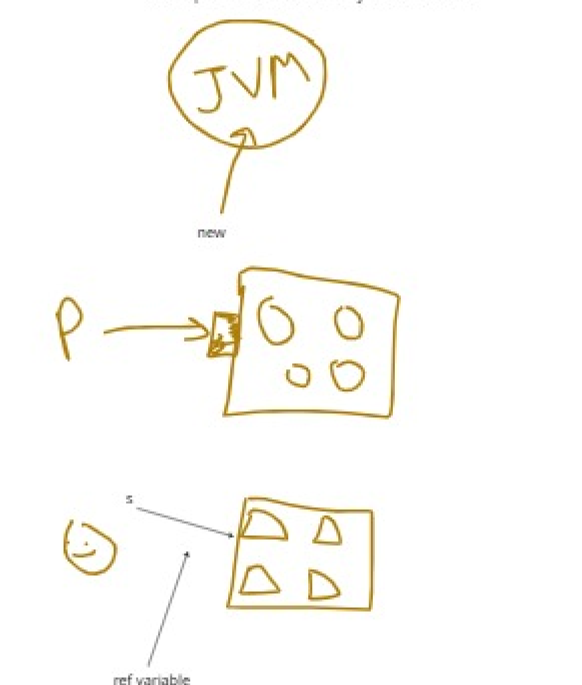
Marks [85]
Submit

```
int marks = 85;

if(marks > 85)
    System.out.println("A Grade");
else if(marks > 60)
    System.out.println("B Grade");
else if(marks > 35)
    System.out.println("C Grade");
else
    System.out.println("Fail");
```



Developer should create an Object of that class



1) for loop
2) while loop
3) do while loop

When should we use a loop?

If we want run certain logic again and again

1) for loop
Use for loop if we want how many time we should run a logic.

EX: Print "Hello" 10 times

While loop

lotterny_no = 7000

We can use While loop, if we want to execute certain statements again and again till the condition becomes true.

```
System.out.println("Enter your lottery no:");
    int lotteryNo = scan.nextInt();

    while(lotteryNo != 7000)
    {
        System.out.println("Enter your lottery no:");
        lotteryNo = scan.nextInt();//700
    }
    System.out.println("Congratulation, "
        + "you have won the lottery!");
```

Class and a Object

Class ==> Container ==> Used to store metho's and variables

```
class SavingsAccount
{
    int accountNo = 0920938823
    int accountHolderName = "Raju"

    dispklayDetails() { s.o.p(accountNo) }
    withdraw() { ---- }
    depoist() { ---- }
    ..............................
    ..............................
}
```

A variable created inside a class can be a LOCAL VARIABLE OR Instance Variable OR static Variable

```
class Student
String name;
int rollno;
String schoolName;
..................
..................
```

s1 ==> (String name; int rollno; String schoolName;)
s2 ==> (String name; int rollno; String schoolName;)
s3 ==> (String name; int rollno; String schoolName;)

s1.name = "Mary"
s1.rollno = 1
s1.schoolName = "ABC High School"

s2.name = "John"
s2.rollno = 2
s2.schoolName = "ABC High School"

s3.name = "Sam"
s3.rollno = 3
s3.schoolName = "ABC High School"

static String schoolName = "ABC ...";

s1 ==> (String name = "mary"; int rollno = 1;)
s2 ==> (String name = "John"; int rollno = 2;)
s3 ==> (String name = "Sam"; int rollno = 3;)

float ==> 0.0
int ==> 0
String ==> null

```
class Student
{
    String sname;
    int srollno;
    static String sschoolname = "ABC High School";
}

public class Demo {

    public static void main(String[] args) {

        Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();
    }

}
```

s1 ==> (sname = "", srollno = 0)
s2 ==> (sname = "", srollno = 0)
s3 ==> (sname = "", srollno = 0)

Constructor ==> Method ==> We can use thi method to provide the value to the instance variable of various objects

Instance variable names and local variable snames ===> SAME

this keyword is used on the instance variable to avoid conflict with local avriable

Giving the values to the constructor is like giving the student details during admission process.

constructo will give the initial OR starting value to the instance variable. Once the values are given to I.V, then we can update those values using setter.

```
class Student
{
    String sname;
    int srollno;
    static String sschoolname = "ABC High School";

    Student(int srollno, String sname)
    {
        this.srollno = srollno;
        this.sname = sname;
    }

    public void setSname(String sname) //sname = "Mark"
        this.sname = sname;
    }

    public void displayStudentDetails()
    {
        System.out.println(this.srollno);
        System.out.println(this.sname);
    }
}

public class Demo {

    public static void main(String[] args) {

        Student s1 = new Student(1, "Raju");
        Student s2 = new Student(2, "Mak");

        s2.setSname("Mark");

        s1.displayStudentDetails();
        s2.displayStudentDetails();

    }

}
```

1) this keyword can be used to differenticate instance and local variables.
2) this keyword can be used in chaining setter method

s1 ——> (sname, srollno, setSname(), setSRollno(), displayStudentDetails())

3) we can use this keyword in constructor chaining to reduce the length of the code and to make the code more readable

Constructor chaining is teh process of calling one constructor from another constructor. We can achieve constructir chaining using this keyword.

4) this keyword can be used to call a method from another method.

ternary opertaor is used to reduce the length of the code that is written using if else statement

```
int age = 40;
String result =
age > 18 ? "D.L can be given" : "D.L cannot be given";
System.out.println(result);
```

Encapsulation:-

Providing security to the most important component of an object is only called as Encapsulation.

A.C ==> coolant ==> shield
Bulb ==> filament ==> glass
....
....
....
....

Coding ==> important component of the object ==> security

setters and getters

setter ==> method ==> provide the value to instance variable
getter ==> method ==> get the value from the instance variable

Method Overloading:-

Single method doing multiple tasks is only called as Method Overloading.

```
System.out.println("welcome")
System.out.println(10)
System.out.println(true)
System.out.println(10.56f)
```

Constructor Overloading