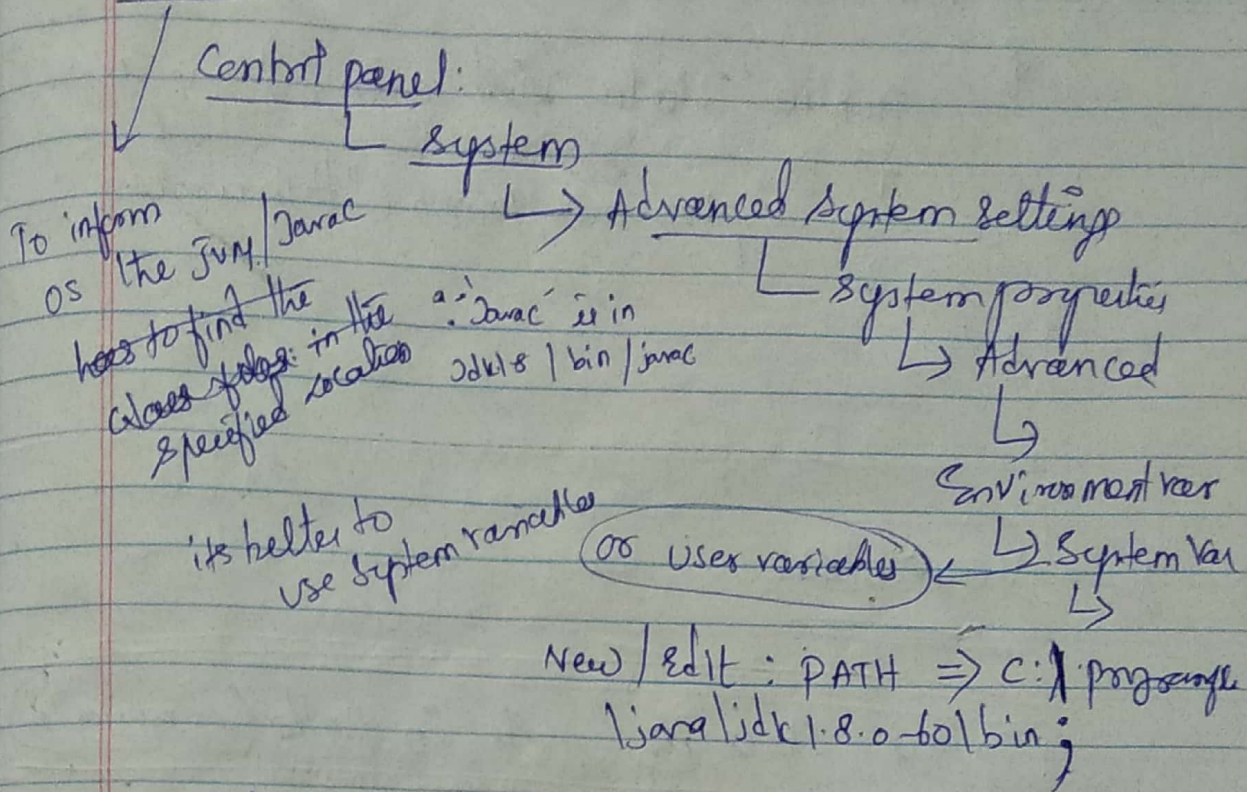


# Java Programs

## class path



## Compile:

> javac FirstProgram.java

## Execution:

> java FirstProgram

(OS)

## Cmd prompt

### Windows

set path = C:\Program Files\Java\jdk\bin

### Mac OS

export JAVA\_HOME = /Library/Java/Home

echo \$JAVA\_HOME



helloworld.java

```

public class helloworld
{
    public static void main (String args[])
    {
        System.out.println ("helloworld");
    }
}
    
```

## Variables:

Object stores its state in fields (or variables)

Kind of Variables:

(i) instance variables:

(non-static fields)   
 local class [static] instance [non-static]   
 ~~Global~~ not ~~shared~~ shared but can be improved using static is public

⇒ objects are declared as ~~instance~~ / stored

⇒ associated with as non-static fields

⇒ the class ~~shared~~ ⇒ Because their values are unique to each instance of a class

⇒ declared in a class but not in a method   
 only method is allowed to create objects when an object is created

Note: These packages are imported by default.

Java compiler ⇒ java.lang.\*;

(ii) class variables (or static fields)

Static variable   
 is in the instance variable   
 is static   
 is shared between all instances of the class   
 is used to store data which is common to all objects

(iii) local variable:

⇒ Method store its temporary state in local variables

⇒ an object stores its state in fields   
 eg: int count = 0;

⇒ no local keyword. Variables declared in open brace & close braces indicate local & not accessible from rest of the class

(iv) ⇒ Parameters:

⇒ Parameters are always variables not fields

Fields ⇒ instance of class (not local & parameter)   
 variables ⇒ all the above as local & parameter equally

## Naming:

- (i) Variable names are case sensitive
- (ii) use mixed letters & digits (VTFF-16)



- begins with letter, \$, or underscore
- Subsequent characters may be letters, digits or underscore or \$
- no keyword or reserved word used as variable name
- One word variable
  - ⇒ spell in lowercase letters
  - eg: gear
- two word or more
  - ⇒ Capitalize the first letter of each subsequent word
  - eg: currentGear
- Constant variable:
  - ⇒ Capitalize each letter
  - eg: final int ~~MAX~~ = 50;
  - (or)
  - final int MAX\_SUBSECT = 5;

Capitalize even for subsequent word & use underscore.

## Quiz:

1) Which type of variable requires a keyword to determine its scope.

Local (ii) static (iii) instance (iv) none

(i) variable is in scope only with in its own method no other code in the class can see.

When we call second time, it recreates the local variable & reinitializes

local variables must be initialized.

Local variable is on the stack (highest frequency heap)

→ Local variable cannot use instance variables

modifiers such as

public, transient, volatile,

abstract, static

but can use final

Class variable

(ii) Every instance of the class shares a class variable.

(iii) Any object can change the value of a class variable.

(iv) can be manipulated without creating an instance of a class



## Class variable Example:

Public class ClassVariable {  
 p.s.v.m (String args[])

{  
 private static int count = 10;

Public class ClassVariable {  
 private static int count = 10;

Public static void main (String args[])

{  
 ClassVariable c1 = new ClassVariable();

ClassVariable c2 = new ClassVariable();  
 ClassVariable c3 = new ClassVariable();

S.o.pn ("c1.count" + c1.count +  
 + c2.count + c3.count);

c1.count = 50;

S.o.pn ("c1.count" + c1.count +  
 "c2.count" + c2.count + "c3.count"

c3.count);

Sig.o.pn ("class.count" + ClassVariable.  
 count);

3  
 3

o/p:

c1.count 10 10 10

c1.count 50 c2.count 50 c3.count 50

class.count 50

Public class Jagra

{  
 private static int count = 10;

3

Public class ClassVariable

{  
 p.s.v.m (String args[])

{  
 Jagra j = new Jagra();

j.count = 50;  
 S.o.pn ("count" + j.count);

error  
 cannot access  
 count  
 private to class  
 Jagra

A static method cannot access a nonstatic

(instance) variable, because there is no instance

eg: Public class Jagra

{  
 int count = 10;  
 p.s.v.m (String args[])

{  
 S.o.pn ("count" + count);

3

non static  
 variable cannot  
 be referenced  
 from a static method

static  
 variable

ie static = class  
 non static = instance

Public class Jagra  
 static int count = 10;  
 p.s.v.m (String args[])  
 {  
 S.o.pn ("count" + count);

3

Static method cannot  
 directly invoke a  
 non static method



## instance variable

→ associated with object

→ defined inside the class but outside of any method

→ initialized when the class is instantiated

→ values are unique to each instance of the class

→ lives as long as the object does

Use four access levels

(i) Final (ii) transient (iii) volatile, public, protected & default

cannot be

abstract, static

if used static then because class variable

public class ~~Person~~

{  
private int height = 0;  
}

public Person()

{  
this.height = 0;  
}

void setHeight(~~set~~ int height)

{  
this.height = height;  
}

int getHeight() {  
return height;  
}

public class int instance variable {

P.S.V.M (Shape) {

Person Song = new Person();

Person Song = new Person();

Song.setHeight(150);

Song.setHeight(150);

Song.setHeight(150);

Song.setHeight(150);

Song.setHeight(150);

}



## Global Variable:

no direct concept of global variable in java but can implement using

(i) static keyword & public access modifier

public class GlobalVariable {

public static int MaxSize = 10;  
public static int MinSize = 0;

Access it as from anywhere.

GlobalVariable.MaxSize  
GlobalVariable.MinSize

with the help of interface.

public interface GlobalVariable {

// variables are implicitly public, static & final  
int MaxSize = 100;  
int MinSize = 0;

public class Sample implements GlobalVariable {

public static void main (String args[]) {  
System.out.println ("MaxSize" + MaxSize + "MinSize" + MinSize);  
}

## Local Variables:

→ only within the method

public class LocalVariable {

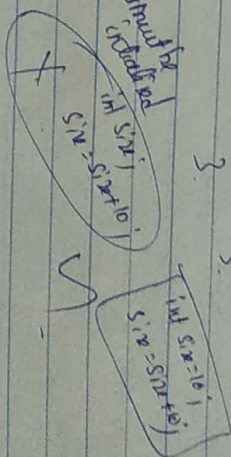
public void main (String args[]) {

int size = 10;  
System.out.println ("size" + size);

}

int size = 10;  
System.out.println ("size" + size);

Local Variable must be declared inside the method



Local Variable

Local Variable must be declared inside the method