# Follow controll

Follow control describe the order in which statements will be executed at runtime

Follow-control

Selection statements Iterative statements Transfer statement

If-else (group of statements execute raptly) (transfer the control one to another)

(only 2 options) while() break

switch do-while() continue

(More options select for() return

one and execute it) foreach() 1.5v try-cutch-finally

assert 1.4v

## Selection statement

### If-else

if(boolean b){

// action if b is true

}

else{

// action if b is false

}

boole x = true;

if(x = false){

sop(“hello”)

}else{

sop(“hi”);

}

int x = 2;

if(x == 5){

sop(“hello”)

}else{

sop(“hi”);

}

int x = 10;

if(x = 20){

sop(“hello”)

}else{

sop(“hi”);

}

int x = 0;

if(x){

sop(“hello”)

}else{

sop(“hi”);

}

o/p: “hi” o/p: “hi”

compileTimeError:

incompatible type (found: int, required: boolean)

else part and curly braces are optional.

Without curly braces only one statement is allowed under if which should not be declarative statement.

* if(true)

sop(“hello”); correct

* if(true); correct (; is empty statement)
* if(true)

int x=10; wrong

(int x is available within this if block, but if is finished then why)

* if(true){

int x=20; correct

}

Every else is mapped to the neatest if.

If(true)

If(false)

else{ sop(“printing”);

### Switch

switch(x)

{

case 1:

action-1;

break;

…

case n:

action-n;

break;

default:

default action;

}

If several options are available then not recommended to go

For **if-else**

if(x==1){ sop(“jan”) switch(x){

}else if(x==2){ sop(“feb”); case 1:

}else if(x==3){ sop(“mar”); sop(“jan”);

}else if(x==4){ sop(“apr”); break;

… case 2:

else if(x==12){ sop(“des”);} sop(“feb”);

break;

#### Allowed argument type of switch

Switch(x)

|  |  |  |
| --- | --- | --- |
| 1.4 | 1.5 | 1.7 |
| byte | Corresponding wrapper classes  Byte | + String |
| short | Short |  |
| char | Character |  |
| int | Integer |  |
|  | + enum |  |

Not allowed

boolean : only 2 cases will be available so no need

long : int range cases is more than enough

float

double 0 … 1 there are infinity cases. Not possible

Curly braces are mandatory in switch statement only. Otherwise curly braces are optional.

Int x =10;

Switch(x){

Is valid. Because both case and default are

} optional

But

Int x =10;

Switch(x){

Sop(“hi”); is wrong complieTimeError:

} Case, default, or } is expected

Within switch every statement should be under case or default.

Independent statements are not allowed

Every cased should be compile-time constant. (Constant expression)

int x =10;

int y = 12; but :: final int y = 12;

switch(x){

case 10:

sop(“ten”);

case y:

sop(“nop”);

}

compileTimeError:

constant expression required

But if y is final then we won’t get any error

Both switch level and case level argument can be expressions. But case level should be constant expression

int x = 10;

switch(x + 12){

case 10:

sop(“ten”);

case 12+13: is valid

sop(“hi”);

}

byte x = 10;

switch(x){

case 10:

sop(“ten”);

case 12+13:

sop(“nop”); invalid

case 1002: byte range is -128 to 127

sop(“hi”);

}

compileTimeError:

incompatible types: possible lossy conversion from int to byte

byte x = 10;

switch(x + 1){ now byte come int

case 10:

sop(“ten”);

case 1003: is valid

sop(“nop”);

}

byte x = 10;

switch(x){

case ‘a’: will be converted to byte(97)

sop(“ten”);

case 97: but duplicate case level (2 time 97)

sop(“nop”); so not valid

}

#### Switch summary

1. Should be constant expression (2+3, not x,y)
2. Value should be in the range of switch

argument type (byte -128 to 127)

1. Duplicate case levels are not allowed

#### Fall through inside switch

Within the switch if any case is matched, from that case onward all statements will be executed until break or end of the switch.

Advantage:

We can define common action for multiple cases (code reusability)

switch(x){ switch(x){

case 1: case o:

case 2: sop(0);

Sop(“q1”); case 1:

break; sop(1)

case 3: break;

case 4: case 2:

case 5: sop(2);

sop(“q2”); default:

break; sop(“def”);

.... }

x = 0 output 0 1

x = 1 output 1

x = 2 output 2 def

x = 3 output def

#### Default case

1. Within the switch we can write default case at most once
2. Default case will be executed if and only if there is no case matched
3. Within the switch we can write anywhere but recommended to write as last case

switch(x){

default :

sop(“def”); x=0 output 0

case 0: x=1 output 1 2

sop(0); x=2 output 2

break; x=3 output def 0

case 1:

sop(1);

case 2:

sop(2);

}

## Iterative statement

### While

If we don’t know number of iterations in advanced then we should go for while loop.

while(rs.next()){}

while(e.hasMoreElements()){}

while(itr.hasNext()){}

while(b){ argument type should be boolean

action

}

while(1){

sop(“hi”); not valid (incompatible type)

}

Curly braces are optional.

Without curly braces only one statement is allowed under if which should not be declarative statement.

* while(true)

sop(“hello”); correct

* while(true); correct (; is empty statement)
* while(true)

int x=10; not valid

(int x is available within this if block, but if is finished then why)

* while(true){

int x=20; correct

}

int x = 2, y = 3;

while(false){

sop(“hello”);

}

sop(“hi”);

int x = 2, y = 3;

while(false){

sop(“hello”);

}

sop(“hi”);

while(true){

sop(“hello”);

}

sop(“hi”);

while(false){

sop(“hello”);

}

sop(“hi”);

final int x = 2;

final int y = 3;

while(false){

sop(“hello”);

}

sop(“hi”);

final int x = 2;

final int y = 3;

while(false){

sop(“hello”);

}

sop(“hi”);

compileTimeError:

**unreachable statement** (sop(“hi”))

also unreachable statement is there but compiler don’t know because there is chance x, y change in runtime

Every final variable will be replaced by the value at compile-time only

final int a =10,b =20

int c =30

sop(a) sop(10)

sop(c) sop(c)

After compilation

sop(a+b) sop(30)

sop(a+c) sop(10+c)

sop(a<b) sop(true)

sop(a<c) sop(10<c)

if every argument is a final variable then that operation should be performed at compile-time only

### do-while

do{

body

}while(b); b should be boolean

If we want to execute