

Operators

[] ()	
x++ x--	post unary
++x --x	pre unary
~ + - !	! only for logic
* / %	
+ -	
<<>>>>	shift
<> <= >=	compare
instanceof	
== !=	for primitives, obj type compare addr
& - ^ ->	logic
&& ->	shortcut logic
(boolean)?x : y	ternary (can be nested)
=	assignment
+= -=...	compound assignment, auto casting

casting (type)

== and != are logic; = is assignment
. operator precedent of casting, so be
careful with (B a).m() vs (B)a.m()

Numeric promotion

promote the larger operand type

byte,short,char ->int->float->double

assign result to promoted type

1)unary exclusive short ++ -> still short

2) by default literal is int or double

float f=1; ok, 1 is a definite number

float f=0.1; X 0.1 is not definite number

long l=powe- X Integer.MAX_VALUE=-
r(2,32) power(2,32)-1

Numeric promotion (cont)

short a=2,b=3,z;z=a*b; a*b -> int

x++ and ++x

int x=3,y;

y=x++; assign x to y(1), then
increment x(2)

y=++x; increment x(2), then assign
x to y(2)

y=++x*5/x-- x=2,y=7

+ --x;

No effect in stand for loop

Using enum

1. dot notation, Days.FRIDAY
2. name(), toString() ->capital label
3. ordinal() -> index from 0
4. valueOf(String) ->parse string to enum
5. compare: == or equals()
- 6.switch (enum_var) {
 case enum_value:
 ..
 default:

}

enum

Why enum?

more readable allow compiler
constant declaration time check

avoid unexpected document upfront
behavior

What's enum start from java5

extends java.lang.enum

enum constant internally implemented as
public static final variables

enum (cont)

overridden equals(), toString(), hashCode()

may define variable,constructor, methods
just like in a class

name(),toString() return ordinal() return
string value index

could has constructor and methods, just like
a class

Assignments

assignment overflow (int i=1.0;byte
too high to hold) b=128;float f=0.1;

explicit cast required when overflow

underflow (float number too small to
represent)

Compound auto cast back to
assignment left operand

assignment chaining int i,j,k; i=j=k=9;
allowed

if (a=9) {} //exception if (a=true){} //ok

enum example

```
public enum Days {
    SUNDAY, MONDAY, TUESDAY,
    WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY;
}

// syntax like class, but with
enum keyword

// name=capital label;ordina-
l=index value from 0
// could have fields/construc-
tor/methods inside
// extends java.lang.enum
// implemented internally as
class,
// enum value are public static
final object constant
public enum MusicType{
    CLASSIC(1), JAZZ(4), ROCK(6), -
    METAL(11);
    private int earDamageFacor;
```

enum example (cont)

```
private MusicType(int earDamageFactor)
{ this earDamageFactor=earDamageFactor; }

public int calcHearingLoss(int huors)
{ return huors*earDamageFactor;}

public String toString ()
{ return this.name()+"-"+this.ordinal()+"_"
+ this.earDamageFactor;
}
```

Java Statement

if (**boolean**) {} multiple statement need {}
 else if (...) {} one statement may miss {}
 else {}
 (...) ? x : y x,y: same data type
 ... ? ...?x :y:z nested
 if (x=5) {} //won't compile

Switch

```
switch (literal--value) {
case 1: Character,char,enum,String
....;break;
case 2: 8 types, after case is literal
....;break;
case 0: default: position not matters
....;break;
default: breaks; avoid follow through,
need in every case, exclude last on
}
```

the case value must be compiling time literal

While

while (...) { .. increate conunter ...}
 do {...} wjile (...) execute >=1 times
 infinity loop while (true) {}

for loop

for (int i;i<arr.length-1;i++)

standard for loop: (4 steps init ->eval ->body exec ->increment, ++i,i++ no different)

multiple declaration initializer allowed, separated by ',', should be same type

initializer scope: inside for loop

for (Class instance:collection){}

for each: iterate only, unable to access i

java.iterable <-java.lamg.collection

Array, ArrayList, List

for(;;){} infinite loop

for(int i=0;i<10;) {i=i++;} infinite loop

for (int value, values) {} -> for (java.util.iterator(Integer) i=value.iterator(); i.hasNext()){}

Advance Flow Control

Nested loops

Optional label

OUTER_LOOP: all flow control allowed, but bad practice
 INNER_LOOP

breaks optional_label; not for if..then..else

breakout current block breakout current block and go to optional label

Continue optional_label; not for if..then..else and switch

stop and Continue next iteration stop and go to optional label

Scope

if, while,for loop define local scope

vaiable declared inside control block could not used out of {}



By **Jianmin Feng** (taotao)
cheatography.com/taotao/

Not published yet.
 Last updated 8th May, 2019.
 Page 2 of 2.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>