Cheatography

Operator an statement Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19373/

Operators

[] ()	
x++ x--	post unary
++x --x	pre unary
~ + - !	! only for logic
* / %	
+ -	
<<>>>>>	shift
<> <= >= instanceof	compare
== !=	for primitives, obj type compare addr
&- ^ ->|	logic
&& -> ||	shortcut logic
(boolean)?x :y	tenary (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()

Numberic 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=power(2,32)	X Integer.MAX_VALUE=-power(2,32)-1

Numberic 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;	x=2,y=7

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 constant declaration	allow compiler time check
avoid unexpected behavior	document upfront
What's enum	start from java5

extends java.lang.enum

enum constant internally implemented as public static final variables

enum (cont)

overriden equals(), toString, hasCode()

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

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

could has constuctor and methods, just like a class

Assignments

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

explicit cast required when overflow

underflow (float number too small to represent)

Compound assignment	auto cast back to left operand
assignment chaining allowed	int i,j,k; i=j=k=9;
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;ordinal=index value from 0
// could have fields/constructor/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 earDam-
ageFactor)
 { this earDamageFactor=earDa-
mageFactor; }

  public int calcHearingLoss(int
huors)
  { return huors*earDamageFac-
tor;}
  public String toString ()
  { return this.name()+"-"+t-
his.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) {	Byte,byte,Short,short,Integer,int
case 1: ....;break;	Character,char,enum,String
case 2: ....;break;	8 types, after case is literal
case 0: ....;break;	default: position not matters
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 ->increament, ++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.uitl.iterator(Integer) i=value.iterator(); i.hasNext-())){}

Advance Flow Control

Nested loops

Optional label

OUTER_LOOP:.... INNER_LOOP	all flow control allowed, but bad practice
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 {}