package my.package;

#if js

import js.Browser;

#elseif sys

import Sys;

#else

import Date;

#end

import Lambda;

using Main.IntExtender;

extern class Math {

static var PI(default,null) : Float;

static function floor(v:Float):Int;

}

/\*\*

\* Abstract forwarding

\*/

abstract MyAbstract(Int) from Int to Int {

inline function new(i:Int) {

this = i;

}

@:op(A \* B)

public function multiply(rhs:MyAbstract) {

return this \* rhs;

}

}

// an enum

enum Color {

Red;

Green;

Blue;

Rgb(r:Int, g:Int, b:Int);

}

@:generic

class Gen<T> {

var v:T;

public function new(v:T) {

this.v = v;

}

public var x(get, set):T;

private inline function get\_x():T

return v;

private inline function set\_x(x:T):T

return v = x;

}

class Main extends BaseClass implements SomeFunctionality {

var callback:Void->Void = null;

var myArray:Array<Float> = new Array<Float>();

var arr = [4,8,0,3,9,1,5,2,6,7];

public function new(x) {

super(x);

}

public static function main() {

trace('What\'s up?');

trace('Hi, ${name}!');

// switch statements!

var c:Color = Color.Green;

var x:Int = switch(c) {

case Red: 0;

case Green: 1;

case Blue: 2;

case Rgb(r, g, b): 3;

case \_: -1;

}

for(i in 0...3) {

trace(i);

continue;

break;

}

do {

trace("Hey-o!");

} while(false);

var done:Bool = false;

while(!done) {

done = true;

}

var H:Int = cast new MyAbstract(42);

var h:Int = cast(new MyAbstract(31), Int);

try {

throw "error";

}

catch(err:String) {

trace(err);

}

var map = new haxe.ds.IntMap<String>();

var f = map.set.bind(\_, "12");

}

function nothing():Void

trace("nothing!");

private inline function func(a:Int, b:Float, ?c:String, d:Bool=false):Dynamic {

return {

x: 0,

y: true,

z: false,

a: 1.53,

b: 5e10,

c: -12,

h: null

};

}

override function quicksort( lo : Int, hi : Int ) : Void {

var i = lo;

var j = hi;

var buf = arr;

var p = buf[(lo+hi)>>1];

while( i <= j ) {

while( arr[i] > p ) i++;

while( arr[j] < p ) j--;

if( i <= j ) {

var t = buf[i];

buf[i++] = buf[j];

buf[j--] = t;

}

}

if( lo < j ) quicksort( lo, j );

if( i < hi ) quicksort( i, hi );

}

}