# Class 1

## Primitive Types

Bite: 0/.1

给刘洁解释什么是0/1

计算机 进行数值计算，进行逻辑计算， 存储记忆功能

人类历史第一台计算机于1946年诞生，如今的硬件技术为第四代

第一代：电子管计算机 （1946-1958）

军事科学计算为主

体积大，功耗高，可靠性差，速度慢（每秒数千次至数万次），价格昂贵

第二代：晶体管数字机 （（1958-1964）

第三代：集成电路数字机 （1964-1970）

应用领域开始进入文字处理和图形图像处理领域

第四代：大规模集成电路机 （1970-now）

Byte: 8 bits

|  |  |  |
| --- | --- | --- |
| Primitive Type | What It Store | Range |
| Byte | 8-bit integer | -128 to 127 |
| Short | 16-bit integer | -32,768 to 32,767 |
| Int | 32-bit integer | -2,147,483,648 to 2,147,483,647 |
| Long | 64-bit integer | -263 to 263 – 1 |
| Float | 32-bit floating-point | 6 significant digits |
| double | 64-bit floating-point | 15 significant digits |
| char | Unicode character |  |
| boolean | Boolean variable | false and true |

2’s complement

+2 -> 取反 -> +1

byte, Boolean | short, char | float, int | long, double

取整：

In Java, when there is decimals, down cast to the closet integer  
全部向零取整， 特别注意负数向零取整

java中只存在向零取整

Char

Single quote is a char; Double quote is string

## 进位制

123| =>. Decimal

0123|=> Oct

0x123|=> Hex

0b101 => bin

String Concatenation

|  |
| --- |
| **优先级** |
| postfix | *expr*++ *expr*-- |
| unary | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| multiplicative | \* / % |
| additive | + - |
| shift | << >> >>> |
| relational | < > <= >= instanceof |
| equality | == != |
| bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| logical AND | && |
| logical OR | || |
| ternary | ? : |
| assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

">>" is arithmetic shift right

">>>" is logical shift right

“<<”，算术或符号左移位 16<<3相当于16\*2^3，高位移出(舍弃)，低位的空位补0。

“>>”，算术或符号右移位 16>>3相当于16/2^3，低位移出(舍弃)，高位的空位补符号位，即正数补0，负数补1

(左乘右除)

上面两个会保留符号（符号位不变）

“>>>” 逻辑右移位，不保存符号，低位移出（舍弃），高位的空位补0，仅对int，long有效 如16>>>2 10000(16)右移两位变为00100(4)

没有“<<<”运算符

## Type Conversion

Implicit conversion: no (double) or (int)

Explicit conversion: with(double) or (int)

Byte -> short -> int -> long -> float -> double

🡪 Widening

It is okay to cast lower precision type to higher precision type implicitly

Not from high precision type to lower precision type.

If you are not sure, always use explicit conversion.

# Class 02

## Conditional

If..else.

For while

if 里面只能有boolean或返回值为boolean的function

Incrementation does not have to be ++.

## Loop.

While.

While is a simplified version of for

为什么for循环没有死循环？

没有跳过i++

for（i=0; I <n; i++) 当statement里面有continue的时候，while loo 和 for loop不等价 ，其他时候都等价！

问： 什么时候 for 循环和 while 循环不等价？

出现 continue 不等价， continue 跳过当前循环节， for 的条件加一在程序首，while 会直接跳过，进入死循环。

for loop只跳过statement；

while loop 跳过statement和increment

## Break and Continue

break跳过整个循环

Continue 跳过当前循环节（只跳最里面)

## Function

Method 《》function

\* modifier （修饰符 = visibility修饰符 + 其他类型修饰符（synchronize））

\* return type

\* name of the method

\* list of parameters

class name: starts with capital letter

method name, variable name: starts with lower-case letter camelCasing.

Note1: what uniquely identifies a function?

Signature: function name + the list of parameter types.

The parameter name does not really matter

Note2: Order of parameter type list matters.

Method overloading. 函数名字形同，但parameter type list 不同的function

1. Different types
2. Different num of parameters
3. Different order

Note3: Return type cannot be used to identify which function to use.

## Return Type

* A primitive type: int, double, char, Boolean
* A class: TreeNode, ListNode
* Void: no return value

如果是non-void类型，一定要保证又一个return会被执行

void 不一定要求写return，但有的时候会有逻辑性作用（比如 return 有提前返回的作用）

返回值不一定要assign给一个variable

Don’t assume the compiler is too smart.

Visibility : public, protected, private;

Protected : package 和 subclass可见

Default : package可见

## Function Quiz

程序 Check Goldbach’s conjecture

**Code Review：**

int i：x （enhance for loop) 枚举，每个值打印一遍

Font: Courier New

找质数： iterate 到根号A就可以 （复杂度降到根号A）

i\*i<a（避免开根号， Math.sqrt(2))

注意corner case

# Class 03

## Array

### 1-D Array

1 byte: byte Boolean

2 bytes: short, char

4 bytes: int float.

8 bytes: long double

The length cannot be changed after creation

初始都为0

如果想用可以改变长度的，应该用list

### 2-D Array

matrix.length 行数

matrix[0].length 列数 （为什么用0？用其他的结果也可能相同； 有可能matrix很特殊，只有一行）

**Code Review：**

for (int i : arr) {

max = (max > i) ? max: i;

}

### Main Function

整个程序的入口

### The Object-Oriented Paradigm in Java

#### Basic concept

Object – 具体 ； Class – 抽象 ； reference 名片地址

成员变量没有初始赋值，int 为0，boolean为false

成员变量没有赋值的话，且为class的类型，默认为null

class里面有两个构造函数可以吗？

可以，可以为overload函数 （前面有讲过）

构造函数和method怎么进行区分？

call一个method必须要加括号，field是不加括号的；

为什么我们需要一个get函数？

大部分为了安全性都是private，大多成员变量为private。我们为了得到需要用一个get函数来得到

构造函数帮助我们创造object，一般都跟着一个new

构造函数一般都跟着new

什么是package： 同一个功能模块的java class 都会放到一个package

#### Object memory layout

操作系统中（Operating System）

Stack: stack variable has the lifetime of the function scope.

Heap: Java objects

-There is no copy of object itself

-Only the reference value of rose is copied and assigned to jack

### Java “Hello World”2: The more you should know

（OP 操作系统 operating system）（not 数据结构，数据结构中的stack是一个桶，heap是一个堆）

Operating中和 数据结构中有什么关系吗？

一半一样，一般不一样， stack是一样的，heap是完全不同的

Stack 栈 后进先出 LIFO

Heap 动态分配内存空间

Static属于整个累，没有属于具体的object

静态函数用类名调用

Parent Class 和 son class 关系“is a”

Stack 存 local variable

Heap 存 object

Stack 和 heap 的区别是什么？

Stack 和 heap都是内存区域 存数据的，不存code；具体的区别，stack 存局部变量，heap存instance 和 object

New 的都是存在heap上面

Stack 后进先出，垃圾桶 Heap GC

Input variable 是存在stack上面的

从main函数进入别的函数之后，他的变量并不会消失，只有伴随着return之后才会消失

Stack 是宝塔形的形状

Return value 是存在stack上，因为只有他上一层的函数需要

有三角形的代表存在heap上

### Primitive types vs. class types

Class types (objects) : stored in heap, operated by references

Primitive data types: basic data types, no reference, data directly stored in memory

### Constructor, this, and Null pointers

Constructor: the same name as the class. Not return type

The current(self) instance. Mandatory if argument name is the same as field name.

Null: An empty reference.

Java Parameter passing

Java function call is always pass by value(copy)

primitive type: copy value itself

objects: copy of the object reference

null 为empty reference

int 是primitive

integer是 objects类型

## Class 4

### Static

Member(field, methods, classes) belong to class, not object

关键词，既可以修饰成员变量，也可以修饰成员函数。

有static是属于整个类，没有的话属于对象

不可以用一个类名调用一个nonstatic method，因为它是属于整个对象的。

Static method can access only static variables

Non-static method can access both static and non-static variables

### Access modifier

Public - everyone can access

Private - only myself can access (but only at class level, other objects of the same class can access as well)

Protected – only my children and same package can access

Default – only the same package can access

### Stacks and Heaps

Stack is smaller than heap; storing data to stack is faster than heap

### Java Parameter Passing

Java function call is always pass by value (copy)

Primitive type: copy of the value itself

Objects: copy of the object reference

## Variables and their scopes

### Local Variables

Local variables mush be initialized before use

Local variable has the lifetime of their own scope

In practice: use possible the MINUMUM scope for each variable. By doing this you can avoid polluting your program/s namespace

### Instance Variables

Field == instance variable == member variable

Non-static variables defined within a class, but not within any methods.

Instance variables have the same life cycle as the instance.

Instance variables’ visibility is defined by their access modifier

Instance variables have default values if they’re not initialized by a constructor

Rule: zero, false and null

### Class Variables (static)

Similar to instance variables, just belong to class but not an instance

## Objects, references and null

Compile Error: In Java, each local variable must be initialized before use. Note: local variables do not have default values!

Runtime Error:

NullPointerException: happens when we try to dereference a reference with null value.

#### Array vs. objects

Arrays are Objects.

The elements in the array still occupy consecutive memory space on the heap

# Class 5 Sorting Algorithms

Data structure is a particular way of organizing data in a computer so that it can be used efficiently

Common data structure

* Array
* Stack
* Queue
* Linked List
* Tree
* Heap
* Graph
* Hash table
* UnionFind
* Trie

Time Complexity O(n)

1<log(n)<sqrt(n)<n<n\*log(n)<n^n<n^3<2^n<10^n<n!<n^n

## Sorting Algorithms

具体情境：

如果你有一摞批改过的卷子，怎么按照分数从小到大排列？

### Selection Sort

第一个想法：每次挑一个最低分，拿出来，放在最下边，然后剩下的再挑一个最低分。

Time: O(n)

Space: O(1)

Code Review : 没什么问题

### Merge Sort

第二个想法：

1. 把一摞卷子平均分成两摞
2. 每一摞排好序
3. 把排好序的两摞合并起来-从两摞卷子上边拿，哪边小哪边

Time : Total Sum = O(nlogn + n) = O(nlogn)

Space: O(n)

### Quick Sort

第三个想法：

1. 随机选一张卷子， 把分数比这张低的放在上边，分数比这张高的放在下边
2. 对上边的进行同样的操作
3. 对下边的进行同样的操作

Time:

Worstcase: O (n^2)

Average case: O(nlogn)

Space:

Worst case: O(n) average O(logn)

# Class 7 Recursion, Queue, Stack

知识点

1. 表象上: function call itself
2. 实质上: Boil down a big problem to smaller ones
3. Implementation上：
4. Base case: smallest problem to solve
5. Recursive rule. How to make the problem smaller

三部曲：

1. Define subproblem
2. Find recursion rule
3. Define base case

**Example1: Fibonacci sequence**

Base case : F(0) = 0; F(1) = 1;

Recursive rule: F(n) = F(n-1) + F(n-2)

Time: O(2^n)

Space: O(n)

**Example 2: Power**

**How to calculate a^b?**

**Method 1**

1. **Define subproblem: f(a, b-1)**
2. **Recursive rule: f(a,b) = f(a, b-1)\*a**
3. **Base case: f(a, 0) = 1**

**Time: O(b)**

**Space O(b)**

**Method 2**

1. **Define subproblem: f(a, b-1)**
2. **Recursive rule: f(a,b) = a^(b/2) \*a(b/2) if b is even number**

**a^(b/2) \*a(b/2)\*a if b is odd number**

1. **Base case: f(a, 0) = 1**

**Time: O(log(b))**

**Space: O(log(b))**

## Queue and Stack

### Queue

**FIFO**

### Stack

# Class 08 Queue & Linked List

Stack: push, pop, size, peek, top, isEmpty

Queue:

**Popular interview questions:**

**Question 1.1 : How could we implement a queue by using two stacks?**

**Approach 1:**

**Time complexity:**

**Push O(n), pop O(1)**

**Approach 2:**

**Push O(1) per operation,, pop Amortized O(1) per operation**

**此处有疑问**

**Question 1.2: How could we implement a stack by using two queues?**

**Approach 1:**

**Two Queues, push O(1), pop O(n)**

**Approach 2:**

**Two Queues, push O(n), pop O(1)**

***Question: 为什么创造出来的新的辅助queue space只是1，不应该是n吗？***

**前两个都是一个做临时的相互导，没有什么区别**

**Approach 3:**

**One Queue, push O(n), pop O(1)**

**只用一个队列，一面加，一面倒**

**Question 2: How to reverse a linked list**

**Time: O(n)**

**Space: O(1)**

# Class 09 LinkedList

Q1: How to find the middle node of a linked list?

Time = O(n)

Space = O(1)

Q2 判定一个linkedlist是否有环？

快慢指针

Time = O(n)

Space = O(1)

Q3 Insert a node in a sorted linked list

Q4 How to merge two sorted LinkedList into one long sorted linked list?

谁小移谁

Dummy head