# Linux

android 的内核：linux；ios 的内核：unix

linux 桌面到终端的切换：ctrl + alt + F2，从终端到桌面的切换：ctrl + alt + F1

ctrl + alt : 本地鼠标与虚拟机鼠标之间的切换

**一. linux常用命令：**

万事万物皆**文件**

linux文件结构：

|  |
| --- |
| 根目录 (/) |
| /bin 放置可以执行的命令。系统有很多放置执行档的目录，但/bin比较特殊。因为/bin放置的是在单人维护模式下还能够被操作的指令。在/bin底下的指令可以被root与一般帐号所使用，主要有：cat,chmod(修改权限), chown, date, mv, mkdir, cp, bash等等常用的指令。 |
| /boot放置引导程序。主要放置开机会使用到的档案，包括Linux核心档案以及开机选单与开机所需设定档等等。 |
| /dev 放置设备驱动。在Linux系统上，任何装置与周边设备都是以档案的型态存在于这个目录当中。只要通过存取这个目录下的某个档案，就等于存取某个装置。 |
| /etc 放置系统配置。系统主要的设定档几乎都放置在这个目录内，例如人员的帐号密码档、各种服务的启始档等等。一般来说，这个目录下的各档案属性是可以让一般使用者查阅的，但是只有root有权力修改。 |
| /home 这是系统预设的使用者家目录(home directory)。 在你新增一个一般使用者帐号时，预设的使用者家目录都会规范到这里来。 |
| /lib 系统的函式库非常的多，而/lib放置的则是在开机时会用到的函式库，以及在/bin或/sbin底下的指令会呼叫的函式库。 |
| /media 底下放置的就是可移除的装置。 包括软碟、光碟、DVD等等装置都暂时挂载于此。 |
| /root 系统管理员(root)的家目录。 |

常用命令（一）：

1. su 普通账号与超级账号之间的切换。su：切换到超级用户；su xxx 切换到普通账户xxx

su gzy 回到普通账户。 超级账号后面是 #，普通账户后面是 $。

2. cd 切换到某一个制定的目录下，例如 cd /home, 切换到home 目录下。注意区别绝对路径和相对路径。tab 键可以对文件名进行自动补全。 cd .. 返回上一级文件夹

3. clear 清屏。

4. pwd 查看当前所在目录。print work directory

5. ls 查看当前目录下的所有文件

6. mkdir 创建文件夹，不是创建文件。mkdir 文件夹名（创建一级文件夹）

mkdir -p 文件夹1/文件夹2/…（创建多个文件夹）

7. man 查看命令的帮助文档，类似于help。举例：man mkdir

8. touch 创建普通文本文件，类似于window里面的“新建文本”。注意创建的不是文件夹。

9. file 用来查看文件类型，如file home。注意，所查看的内容必须在当前目录下。

10. rm 删除文件，如rm xxx (只能删除一个文件，不能是文件夹) rm –rf xxx (可以删除文件夹，但是非常危险，是强制删除)

11. top 动态查看任务管理器。进入top后按q退出。

12. ps 静态查看任务管理器，一般加上参数：ps –au

13. shutdown 关机或重启，只有超级用户才有关机的权限，一般加上参数：

shutdown –h now立刻关机；shutdown -r now 立刻重启

14. reboot 重启，只有超级用户才有该权限。

常用命令（二）：

1. history 查看历史命令，会看到带有编号的历史命令。如需再次调用历史命令，输入! +编号，例如：!43

2. find 可以在制定目录下查找文件，如find /目录名/xx –name 要查找的文件名。 管道命令( | )：把多个命令串联起来，举例：find / -name he\* | grep ab 在根目录（/）下查找以he 开头且包含ab的文件

3. tar 压缩与解压文件

压缩文件语法：tar –czvf 文件名1.tar.gz 文件名2 （与一般的习惯不同，文件1是压缩后的名字，文件名2是想被压缩的名字，压缩后的文件后缀一般为.tar.gz）

解压文件语法：tar –zxvf 文件名.tar.gz

tar -C DesDirName -zxvf 文件名.tar.gz 解压到目标路径

4. ifconfig 查看linux下的ip地址

5. service防火墙，具体为：service iptables start 开启防火墙 service iptables stop 关闭防火墙

6. rpm –qa | grep 查看linux下是否安装了指定的软件。例如rpm –qa | grep java

7. rpm -e --nodeps xxxxxx 卸载软件

（1）强制删除已安装python及其关联

# rpm -qa|grep python|xargs rpm -ev --allmatches --nodeps

（2）删除残余文件

# whereis python|xargs rm -frv

8. vi 打开文件。举例：vi xxx（xxx为普通**文本文件**名，打开后并不能直接编辑，默认为只读）打开后输入i，即可进入编辑模式，插入完成后要从编辑状态切换到命令状态（先按esc, 再按冒号:），然后可以输入q!(强制退出，不保存)，wq(保存并退出)

9. cat 打开一个文件，但是不进入到编辑框

10. kill 关闭进程，结合top命令使用。先top命令查询进程的PID，然后kill -9 xxx

11. chmod 赋予文件夹权限，如 chmod 777 /home/gzy

12. echo -e "\033[?25h" 显示光标

13. pip install -i https://pypi.tuna.tsinghua.edu.cn/simple xxxx

14. lsb\_release –a 查看linux是否为centos

15. PYTHONIOENCODING=utf-8 python your\_script.py

16. vim ~/.vimrc 在其中写入 set ts = 4 (使tab键等同于4个空格) ,set nu（显示行号）

# SQL

1. create table xxx as select: 创建表并实现表内容的拷贝

2. NVL(表达式1，表达式2):如果表达式1为空值，NVL返回值为表达式2的值，否则返回表达式1的值。

3. TRUNCATE 函数：删除表中的记录，保留表

e.g. TRUNCATE TABLE 表名

4. cast(expression AS data\_type)：用于数据转换

5. row\_number() over(partition by v1 order by v2 desc) rank: 根据v1分组，根据v2排序, 生成一个新变量命名为rank，内容为排序的数字。

rank() over over(partition by v1 order by v2 desc)

dense\_rank() over(partition by v1 order by v2 desc)

三者之间的区别是：row\_number() 排序后不考虑并列，直接是1，2，3，4，5

rank() 考虑并列，为1，2，2，4，5

dense\_rank()考虑并列，为1，2，2，3，4

6. select ‘1’ flag : 生成一个变量flag，内容为’1’。

7. replace(s1,t1,t2) 查询s1, 如果s1有t1则替换为t2。如果是字符串记得加括号。

8. wm\_concat('{x:'||mp\_age\_range||',n:'||cnt||'}')

9. decode(col,k1,v1,k2,v2,k3,v3,x) 判断col中的内容，如果是k1则返回v1, 是k2则返回k2……都不是则返回x, x可以为col本身。

10. TRUNC() 函数：截断函数

TRUNC(number)截断数字，TRUNC(date)截断日期

TRUNC(SYSDATE,’year’) 返回当年第一天

# Oracle

单词字母开头大写代表为命令。

1. 创建数据库 Create Database databasename

2. 删除数据库 Drop Database dbname

3. 创建表

Create Table tabname(col1 type1 [not null] [primary key],col2 type2 [not null],..)

根据已有的表创建新表：

A：Select \* Into table\_new From table\_old (使用旧表创建新表)

B：Create Table tab\_new As Select col1,col2… from tab\_old definition only<仅适用于Oracle>

4. 删除表：Drop Table tabname

5. 重命名表 Alter Table 表名 Rename To 新表名

6. 增加字段 Alter Table 表名 Add (字段名 字段类型 默认值 是否为空);

例：Alter Table tablename Add (ID int);

eg：Alter Table tablename Add (ID Varchar2(30) Default '空' Not Null);

7. 修改字段

说明：Alter Table 表名 Modify (字段名 字段类型 默认值 是否为空);

eg：Alter Table tablename Modify (ID number(4));

8. 重名字段

说明：Alter Table 表名 Rename Column 列名 To 新列名 （其中：column是关键字）

eg：Alter Table tablename rename column ID to newID;

9. 删除字段

说明：Alter Table 表名 Drop Column 字段名;

eg：Alter Table tablename Drop Column ID;

10. 排序：ORDER BY name DESC 注：DESC放在末尾

SELECT name, credit\_limit,address

FROM customers

ORDER BY 2 DESC, 1;

在这个例子中，name列的位置是1，credit\_limit列的位置是2。

ORDER BY state ASC NULLS FIRST/LAST 首先将NULL值放置在前面/后面。

ORDER BY子句可在一列上应用一个函数，如ORDER BY UPPER( name )。

11. SELECT DISTINCT column\_1, column\_2, …, column\_n : 在此语法中，column\_1，column\_2和column\_n中的值的组合用于确定数据的唯一性。

DISTINCT将NULL值视为重复值。如果使用SELECT DISTINCT语句从具有多个NULL值的列中查询数据，则结果集只包含一个NULL值。

12. Oracle外键

在Oracle数据库中，外键是强制实施参照完整性的一种方式，使用外键就意味着一个表中的值在另一个表中也必须出现。

被引用的表称为父表，而带有外键的表称为子表。子表中的外键通常会引用父表中的主键。

（1）使用CREATE TABLE语句创建外键的语法是：

CREATE TABLE table\_name

(

column1 datatype null/not null,

column2 datatype null/not null,

...

CONSTRAINT fk\_column

FOREIGN KEY (column1, column2, ... column\_n)

REFERENCES parent\_table (column1, column2, ... column\_n)

);

示例：

CREATE TABLE supplier

( supplier\_id numeric(10) not null,

supplier\_name varchar2(50) not null,

contact\_name varchar2(50),

CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id)

);

CREATE TABLE products

( product\_id numeric(10) not null,

supplier\_id numeric(10) not null,

CONSTRAINT fk\_supplier

FOREIGN KEY (supplier\_id)

REFERENCES supplier(supplier\_id)

);

在这个例子中，我们在supplier表上创建了一个名为supplier\_pk的主键。 它只包含一个字段 - supplier\_id字段。 然后，在products表上创建了一个名为fk\_supplier的外键，该表根据supplier\_id字段引用supplier表。

（2）在ALTER TABLE语句中创建外键的语法是：

ALTER TABLE products

ADD CONSTRAINT fk\_supplier

FOREIGN KEY (supplier\_id)

REFERENCES supplier(supplier\_id);

在这个例子中，我们创建了一个名为fk\_supplier的外键，它根据supplier\_id字段引用supplier表的supplier\_id字段。

13. Like 语句后面跟着字段的表示方法：LIKE '%'||table1.col1||'%'

因为oracle中没有+号，字符串的连接为 ||

14. like 'ab\%%' ESCAPE '\' 此处代表匹配‘ab%‘的字符串。

ESCAPE '\' 表示以 \ 为转义符

15. Oracle中不支持limit方法。

# Neo4j

1. 查询所有节点和关系：

MATCH (n)

OPTIONAL MATCH (n)-[r]-()

RETURN n,r

2. 删除所有的节点和关系：

MATCH (n)

OPTIONAL MATCH (n)-[r]-()

DELETE n,r

3. MATCH (accountHolder:AccountHolder)-[]->(contactInformation)

WITH contactInformation, count(accountHolder) AS RingSize

WITH的作用是对上面的MATCH进行处理，比如，给标签重命名，使用聚合函数等。

WITH 可以将上一条查询语句的结果连接起来，用于下一条查询语句之中，即起到一种管道的功能。

4. collect() 函数：把得到的结果以list的形式返回。

5. MATCH (p1:Person {name:'Michael Sherman'})-[r1:RATED]->(m:Movie)<-[r2:RATED]-(p2:Person {name:'Michael Hunger'})

RETURN m.name AS Movie, r1.rating AS `M. Sherman's Rating`, r2.rating AS `M. Hunger's Rating`

(p1)-[r1]->(m)<-[r2]-(p2) 查找p1和p2通过关系关注的相同的m.

6. MERGE可以看作是CREATE和MATCH的合体，即查找属性、节点、关系，不存在则创建，存在则返回。

7. Neo4j导入数据的方法：

1) LOAD CSV 方法：

a. 把csv文件放到import文件夹中；

b. 打开console，进入浏览器；

c. 输入命令

USING periodic commit 1000

LOAD CSV WITH HEADERS FROM "file:///test.csv" AS line

CREATE (p:person{id:line.id,name:line.name,age:line.age})

// 也可以采用列表的形式表示 id:line[1]

// 有多少行，就生成多少个节点

8. 查询最短路

MATCH paths = allShortestPaths((a:Page { title:'Angela Merkel' })-[:Link\*]->(b:Page { title:'Apple Inc.' }))

RETURN paths

// [Link\*]可以更改为[\*..n]，表示路径深度n以内(n为数字)查找所有存在的关系中的最短路径关系。

MATCH path = shortestPath((a:Page { title:'Doctor Who' })-[:Link\*]->(b:Page { title:'Microsoft' }))

RETURN path, length(path) AS Steps

9. OPTIONAL MATCH ()-[]->(m)

查找关系，当关系不存在时，会保留节点m，使用MATCH找不到关系时不会保留节点m。

OPTIONAL MATCH path = (a:Page { title:'ASP.NET' })-[:Link]->(b:Page { title:'.NET' })

RETURN

CASE

WHEN path IS NULL THEN FALSE

ELSE TRUE

END AS ValidPath

10. =~ ‘regexp’ 用于匹配正则表达式

11. RETURN \* 返回所有节点和关系。

12. 使用一个参数设置所有属性：

{

"props" :{ name: "Andres", "position":"Developer"

}

}

MATCH (n{name: ‘Andres’})

SET n = $props

RETURN n

13. 创建和使用索引：

CREATE INDEX ON :Person(name) //给节点的name属性添加节点

索引创建完成后，在使用时自动调用。或者，指定使用的索引：

MATCH (n:Person) USING INDEX n:Person(name)

DROP INDEX ON :Person(name) //删除索引

14. 利用现有节点创建关系时，直接使用create和merge会创建新的节点，即使添加了属性限制。应当先match, 然后再create.

15. 创建约束：

Create Constraint On (p:Person) Assert p.name Is Unique

删除约束：Drop Constraint On (p:Person) Assert p.name Is Unique

16. 使用CASE 子句：

MATCH (n:Test) RETURN

CASE n.name

WHEN “a” THEN 1

WHEN “b” THEN 2

ELSE 3

END AS test

17. CALL db.labels() YIELD label RETURN label //查询数据库中可用的标签

# Java

**1**. java 的强制转换：int a = (int) (c+b)

2. s=s+1 与s +=1不同。+=隐含了一个强制转换。

3. 赋值规则：

包：全部小写，多级包用 . 隔开，如com.china.gzy

类或接口：一个单词的，单词首字母大写，多个单词拼接的，每个单词首字母大写

方法或者变量：一个单词的，小写，多个单词拼接的，除第一个单词外，其他单词首字母大写，如getName

常量：一个单词的，全部大写，如PI，多个单词拼接的，全部大写，用 \_ 隔开，如MAX\_AGE

**4.** &&相比较&具有短路的效果，即左边false，右边不执行，因此效率高，开发中常用&&. || 也是同理。

**5.** ^ 在逻辑运算中表示相同则0，不同则1，类比于男女关系。

**6.** 键盘录入：共三步

import java.util.Scanner;

Scanner sc = new Scanner(System.in);

String x = sc.next();

**7.** 一般来说，有{}就没有; 有;就没有{}

**8.** for 循环中，break 表示退出当前循环，即直接退出本层循环。可以结合标签选择跳出的循环层，例如：

f1: for(;;){

f2: for(;;){

if(ture){

break f1;

}

}

}

continue 是退出本次循环，还是会继续下一次循环的。

return 是不仅退出循环，还退出了整个方法。

**9.** 方法：在java中称之为方法，在其他编程语言中称为函数。注意：方法不调用则不会执行；方法与方法是平级关系，不能嵌套定义，例如，不能在mian方法中定义方法；定义方法时，方法中的参数用逗号隔开；方法调用的时候不用再次传递参数类型；当参数不同时，方法名可以相同，称为“方法重载”

void 返回值方法只能单独调用，不可赋值给其他值。

格式：

修饰符 返回值类型 方法名(参数类型1 参数名1,…..,) {

方法体语句；

return 返回值；

}

修饰符: public static

返回值类型: 就是结果的数据类型

方法名：符合命名规则即可

参数：分为实际参数和形式参数

**10.** 数组

定义： int[] a; 或者 int a[]; 建议采用第一种

数组必须初始化后才可以使用，初始化方法：

i. 数据类型[] 数组名=new 数据类型[数组长度] //只指定长度

int[] arr = new int[3]; //数组默认值全是0

ii. 数据类型[] 数组名=new 数据类型[]{元素1, 元素2,….} //指定内容，不指定长度

iii. 数据类型[] 数组名={元素1, 元素2,….} 是第2种的简化方式

数组的常用操作：

arr.length //获取数组长度

二维数组：

int[][] arr = new int[m][n] //m表示有几个一维数组，n表示每个一维数组有n个元素。此时，arr.length = m

int[][] arr = new int[m][] //定义了二维数组，数组元素个数可变

数据类型[][] 变量名={{元素…}{元素….}{元素…}[{…}}

**11.** 面向对象的编程思想：把多个方法封装到一个类中。

**12.** 类的定义包括：变量和方法，方法包括成员方法和构造方法。

类名 对象名 = new 类名();

成员变量：在类中方法外；随着对象存在或消失；有默认初始化值；

局部变量：在方法定义中或者方法声明上；随着方法存在或消失；没有默认初始化值，必须定义赋值后才能使用；

**13.** 匿名对象：没有名字的对象，如 new Student();

匿名对象可以调用方法，new Student().method(); 调用完被回收。匿名对象可以作为实际参数传递。

**14.** 封装：隐藏对象的属性和使用细节，仅对外提供公共访问方式。使用private修饰，private修饰的变量和方法只能在本类中进行访问。

**15.** this 代表所在类的对象引用，如this.age等价于Student.age。方法被哪个对象调用，this就代表哪个对象。

**16.** 构造方法：给对象的数据进行初始化。如：

class Student{

public Student(){}

}

如果没有定义构造方法，系统会自动提供一个无参构造方法。如果定义了构造方法，系统将不再提供默认的无参构造方法，必须自己给出才可调用。

构造方法可以重载。

**17.** static 修饰的变量为所有成员所共享，随着类的加载而加载，可以被类调用，也可以被方法调用；静态方法中没有关键字this，静态方法只能访问静态的成员变量和静态的成员方法。

**18.** 测试类的作用：创建其他类的对象，调用其他类的功能。

**19.** java.lang包下的类不需要导入，其他的全部需要导入。

**20.** java中，用{}括起来的代码称为代码块。

构造代码块：在类中的成员位置，每次调用构造方法执行前，都会先执行构造代码块。

静态代码块：在类中的成员位置，用static修饰，一般是对类进行初始化，只执行一次，意思是，多次实例化一个类也只执行一次静态代码块。

一个类中，最先执行静态代码块（甚至优先于main方法），然后构造代码块，最后是构造方法。

class Code{

static {// 这是静态代码块

int a =100;

}

{ //这是构造代码块

}

}

**21.** 继承：extends

class 子类名 extends 父类名{}

java中继承的特点：java只支持单继承，不支持多继承；java支持多层继承；子类只能继承父类所有非私有的成员（成员方法和成员变量）；子类不能继承父类的构造方法，可以通过super关键字来访问父类构造方法；子类的所有构造方法默认都会访问父类中空参数的构造方法；子类每一个构造方法的第一条语句默认都是super();

**22.** this和super的区别：

this 代表本类对应的引用，super代表父类存储空间的标识

**23.** 子类调用方法规则：先看子类中有没有，再看父类。

**24.** 方法重写：子类中出现了和父类中一模一样的方法声明。方法重写即沿袭了父类的功能，又定义了子类特有的功能。在重写的方法中，要想保留父类方法的功能，需要在子类重写的方法中添加super.xxx()

**25.** final 可以修饰类，方法，变量。如 final class Hello{} public final void show()

final 修饰的类无法被继承，final修饰的方法无法被重写，final修饰的变量无法被重新赋值，即只能被赋值一次如同常量，如 final in num; num = 100;

**26.** 多态：同一对象，在不同时刻体现的不同状态。多态的前提：要有继承关系；要有方法重写；要有父类引用指向子类对象，如 父类 f = new 子类，此时f只能访问父类的变量，因为存在方法重写，f访问的是子类的方法，

**27.**多态中父类不能使用子类的特有方法，解决方法：向下转型。向上转型：

父类 f = new 子类()，把一个子类类型安全地变为父类类型的赋值; 向下转型：父类 f = new 父类(); 子类 z =(子类) f，把一个父类类型强制转型为子类类型

**28.** 抽象类：如果父类的方法本身不需要实现任何功能(没有{})，仅仅是为了定义方法签名，目的是让子类去覆写它，那么，可以把父类的方法声明为抽象方法。一个类如果有抽象方法，这个类也必须是抽象类，用abstract修饰，如abstract class A{ public abstract void method();}

抽象类无法被实例化；抽象的子类要么也是抽象类，要么全部重写父类的全部抽象方法。

abstract不能与private, final, static 共存。

**29.** 接口：如果一个抽象类没有字段，所有方法全部都是抽象方法，这个类就可以定义为接口。方法：interface Person{}

当一个具体的class去实现一个interface时，需要使用implements关键字，如：

class Student implements Person {}

一个类可以实现多个interface, 例如class Student implements Person, Hello { // 实现了两个interface ... }

注意事项：接口仍是抽象的，不能实例化；接口内可以定义变量，但是全部为常量；接口可以继承接口，即可以单继承，也可以多继承；抽象类被继承，体现的是“is a”的关系，接口被继承，体现的是“like a”的关系。

**30.** 包的概念：包就是文件夹，对类就行分类管理，把相同的类名放到不同的包中，定义方法：package 包名; 多级包用 , 分开package放到程序的最上。

**31.** 不同包下的类之间的访问：使用import 包名.类名;

注意：虽然import 包名.\*; 可以导入所有的类，但是不推荐，应当用哪个类就导入哪个类。

**32.** java的权限修饰符：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **本类中** | **子类中** | **同包类中** | **其他类中** |
| **public** | 可以 | 可以 | 可以 | 可以 |
| **protected** | 可以 | 可以 | 可以 | 不可以 |
| **默认** | 可以 | 同包子类可以 | 可以 | 不可以 |
| **private** | 可以 | 不可以 | 不可以 | 不可以 |

类：只能public或者为默认，不能用protected和private修饰，内部类可以加private修饰。

构造方法：4中权限修饰符都可以修饰构造方法，且构造方法只能使用权限修饰符修饰，不能用static，final，abstract修饰。

**33.** 内部类：把一个类定义在其他类的内部，这个类就是内部类。内部类的访问特点：内部类可以访问外部类的所有成员，包括私有；内部类可以加private修饰；内部类和外部类不是继承关系。内部类可以分为成员内部类（定义在成员变量处）和局部内部类（定义在类的方法中）

**成员内部类的访问：**

class Person{

int num = 10;

class Student{

public void show() {

System.out.println(num);

}

}

}

要想访问内部类Student的show方法，代码如下：在main函数中，

// 外部类.内部类 对象名 = new 外部类().new 内部类();

Person.Student ps = new Person().new Student();

ps.show();

如果内部类用static作限定，访问方法为：

Person.Student ps = new Person().Student(); //访问静态方法

ps.show2();

**局部内部类的访问：**

局部内部类可以访问局部变量，这个局部变量必须用final修饰。

**34.** 匿名内部类：内部类的简化写法，前提是存在一个类或者接口。格式：

new 类或接口名(){ 重写方法(); }

**35.** Eclipse重要快捷键：Ctrl + Shift + F : 代码格式化

Ctrl + Shift + O : 自动导包

Ctrl + / : 选中行的添加、取消注释，生成的是//

Ctrl +Shift + / : 选中行的添加注释，生成的是/\* \*/

Ctrl +Shift + \ : 选中行的取消注释

F3 或者 按住Ctrl单机：查看选中类的源码

**36.** Eclipse 可以把一个项目导出为jar包。项目中添加jar包的方法：复制jar包到项目中，然后右键这个jar包，选择Build Path🡪Add….

**37.** Scanner中的常见方法：sc.hasNextXxx() 返回布尔类型，判断下面键盘输入的格式是否为Xxx。

Scanner sc = new Scanner(System.***in***);

if(sc.hasNextInt()) { //判断等下输入的是不是int类型

int x = sc.nextInt();

System.***out***.println(x);

}else {

System.***out***.println("格式不对");

}

**38.** 使用Scanner获取多个对象时，如果先获取int再获取String会出错，应当先把所有的对象设为String，然后再转换为相应的类型。

**39.** String 生成的变量一经赋值便无法修改

String s = "Hello";

s += "World";

这里，并非s的值发生了改变，而是生成了一个新地址指向s.

40. ==与equals的区别：

==比较的是地址值，equals比较的是内容。

String s1 = new String("abc");

String s2 = "abc";

System.***out***.println(s1==s2); //false

System.***out***.println(s1.equals(s2));//true

41. String 的几个功能：

.equals() 返回布尔值，判断是否相等

.equalsIgnoreCase 判断是否相等，忽略大小写

.contains() 是否包含

.isEmpty() 是否为空

42. String的获取功能：

substring(int start, int end) //截取子串，从指定位置开始，指定位置结束，包括start不包括end，而不是取几个。int end 可以省略，则意味着取到最后。

charAt(int index) 获取字符串指定索引的字符。

43. 字符串的遍历操作：

String s = "abcdefghijklmnopqrstuvwsyz";

for(int i=0;i<s.length();i++) {

System.***out***.println(s.charAt(i));

}

44. 字符串的转换：

返回值 方法

char[] toCharArray(); 把字符串转换为字符数组

String concat(String str); 字符串拼接

static String ValueOf(char[] chs); 把字符数组转换为字符串

String toLowerCase(); 把字符串转为小写；

String toUpperCase(); 把字符串转为大写；

String replace(old, new); 替换

String trim(); 只去除字符串两端的空格，中间的空格不去除

int compareTo(String str); 比较两个字符串是否相当，如果相等返回0

String[] split(String regex); 字符串分割，返回字符串数组，如String[] arr=str.split(“ ”); 按空格分隔

45. StringBuffer: 线程安全的可变字符串，与String相比，前者内容可变使用StringBuffer 的字符拼接功能节约空间，使用前需要生成对象：

StringBuffer sb = new StringBuffer(); //构建了一个缓存区

46. StringBuffer sb = new StringBuffer();

sb.append(“hello”);// 向后添加字符

sb.insert(int x," "); // 指定位置插入

sb.deleteCharAt(int index); //删除指定位置的字符

sb.delete(int start, int end); //删除起始位置的字符，不包括end，必须填入两个参数，不可省略

sb.toString(); //转成字符串

sb.replace(start, end, str); 把指定区域的字符进行替换

sb.reverse(); 反转

47. Arrays类的操作：

int[] arr = {……};

Arrays.sort(arr); //从小到大排序

Arrays.binarySearch(arr, key) //查找数组arr中数值key的索引

String s = Arrays.toString(arr); //数组转成字符串

48. int与String 的转换：

int num = 10;

String s = String.*valueOf*(num);// int\_to\_str

String str = "100";

int n = Integer.*parseInt*(str); //str\_to\_int

49. String 和StringBuffer之间的转换

String\_to\_StringBuffer: 使用StringBuffer的append方法

StringBuffer\_to\_String: 使用StringBuffer的toString方法

50. 正则表达式：

|  |  |
| --- | --- |
| **形式** | **含义** |
| [abc] | a,b,c任一个 |
| [^abc] | 除了a |
| [a-zA-Z] | 所有字母中的任一个 |
| \w | 任意字母，注意，要想使用，需要\\w, 其他的类似 |
| \W | 非字母 |
| [0-9] | 0到9 |
| \d | 任意一个数字 |
| \D | 非数字 |
| {n,m} | {}前面的重复n到m次 |
| . | 任意字符。想要表示.需要\\. |
| ^ | 行的开头 |
| $ | 行的结尾 |
| \b | 单词边界，即不是单词的地方 |
| ? | 0-1次 |
| \* | 0次或多次 |
| + | 一次以上 |
|  |  |

matches(String regex)，返回布尔类型, 判断字符串是否符合regex的规则

split(String regex) 返回类型为String[], 按照regex规则进行字符串拆分

replaceAll(String regex, String replacement) 替换功能

51. 正则表达式的获取功能：

import java.util.regex.Pattern;

Pattern p = Pattern.*compile*(String regex);

Matcher m = p.matcher(String str); //通过m调用相关的匹配函数

52. Random 类的常见方法

Random r = new Random(int n); //n是设置的随机数种子，可以省略

int num = r.nextInt(100); r.nextInt(100) 返回一个0-100之间的随机数

53. System的一些常用类

System.exit(0); //终止当前的jvm，下面的程序将不再执行，类似于break;

System.currentTimeMillis() //返回以毫秒为单位的当前时间，类型为long

54. Date类

Date d = new Date();

System.out.println(d); //按标准日期输出当前时间

long time = d.get.time(); //以毫秒为单位获取时间

SimpleDateFormat sdf = new SimpleDateFormat();

String s = sdf.format(d); //这一步将Date得到的时间按照默认的格式进行格式化

d.setTime(long time); //设置时间

//以下代码按照便于理解的格式输出当时日期

Date d = new Date();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");

String s = sdf.format(d);

System.***out***.println(s);

55. Calendar 类

Calendar now = Calendar.*getInstance*();

int year = now.get(Calendar.***YEAR***);

int month = now.get(Calendar.***MONTH***); //获取月份，从0开始，加上1为正常的月份

int day = now.get(Calendar.***DATE***);

now.add(Calendar.***YEAR***,4); //设置时间为4年后

now.add(Calendar.***DATE***,-10); //设置时间为10天前

now.set(int year,int month,int dat) //设置日期为某年某月某日，注意，month是从0开始的，如果month为2，实际上是3月

56. 集合与数组的区别：数组的长度固定，集合的长度可变；数组的类型相同，集合的类型可以不同；集合只能存储引用类型

57. 集合的常用功能

// Collection c = new Collection(); //错误，接口不能实例化

Collection c = new ArrayList();

|  |  |
| --- | --- |
| **代码** | **作用** |
| c.add(String str) | 向集合c中添加元素，元素可以重复 |
| c.clear() | 清除集合c中的元素 |
| c.remove(String str) | 移除元素，如果c中不存在str则不移除，并返回false |
| c.contains(String str) | 判断是否包含 |
| c.isEmpty() | 判断是否为空 |
| c.size() | 求集合c的元素个数 |
| c.addAll(Collection c1) | 向集合c中添加另一个集合c1, 集合c中的元素可以重复 |
| c.containsAll(Collection c1) | 判断c是否包含c1, 只有c1是c的子集才会返回true |
|  |  |

58. 集合的遍历

//方式1

Object[] obj = c.toArray(); //把集合c转为Object数组

for(int i=0;i<obj.length;i++) {

String s = (String) obj[i]; //转为String，因为Object没有lenth()函数

System.***out***.println(s);

System.***out***.println(s.length());

}

//方式2 使用迭代器

Iterator it = c.iterator(); //生成迭代器

while(it.hasNext()) {

String s = (String) it.next(); //next()是获取一个元素，并指向下一个元素

System.***out***.println(s);

}

59. List: Lists 是有序的集合Collection, 特点是可重复。有序是指存储和取出的元素一致。

List list = new ArrayList();

list.add(String str); //添加元素

利用迭代器访问List的元素。

60. List的特有功能：

List list = new ArrayList();

|  |  |
| --- | --- |
| **代码** | **作用** |
| list.add(int index, Object element) | 在指定位置添加元素 |
| list.get(int index) | 获取指定位置的元素 |
| list.remove(int index) | 删除指定位置的元素 |
| list.set(int index, Object element) | 根据索引修改元素 |

遍历：

for(int i=0;i<list.size();i++) {

String s = (String) list.get(i);

System.***out***.println(s);

}

或者使用迭代器

61. 常见数据结构

栈：先进后出

队列：先进先出

链表：由一个链子把多个结点连起来组成的数据，结点包含数据和地址（数据域和指针域）。链表查找数据都是从头开始。

62. List的三个子类的特点ArrayList, Vector, LinkedList

ArrayList: 底层数据是数组，查询快，增删慢，线程不安全，效率高。

Vector: 底层数据是数组，查询快，增删慢，线程安全，效率低。

LinkedList: 底层数据是链表，查询慢，增删快，线程不安全，效率高。

63. 泛型：是一种把类型明确的工作推迟到创建对象或者调用方法的时候才去明确的特殊类型，也叫参数化类型。优点是把运行时期的问题提前到了编译时期，避免了强制类型转换。

ArrayList<String> array = new ArrayList<String>();

Iterator<String> it = array.iterator();

64. 泛型方法的使用：把泛型定义在方法上

public static class Student{

public <T> void show(T t) {

System.***out***.println(t);

}

}

然后在main方法中，

Student s = new Student();

s.show("Hello");

s.show(100);

65. 泛型的通配符：

? 表示任意的类型都可以

Collection <?> c = new ArrayList<Object>();

? extends E 向下限定，E及其子类

? super E 向上限定，E及其父类

66. 增强for

int[] arr = {1,2,3,4,5};

for(int s : arr) {

System.***out***.println(s);

// 也可以进行其他操作

}

可以用于集合的遍历

67. 可变参数：定义的时候未指定

参数的个数，格式为：

修饰符 返回值类型 方法名(类型.. 变量名){}

public static int sum(int... a){} //这里的a 实质是一个数组

68. Set: 一个不包含重复元素的Collection，特点：无序，存储和取出顺序不一致，有两个子类:HashSet 和 TreeSet

Set<String> set = new HashSet<String>();

69. TreeSet 可以对元素按照某种规则进行排序，底层是二叉树结构。排序方法：自然排序，比较器排序。

自然排序：

TreeSet<Integer> ts = new TreeSet<Integer>();

比较器排序：重写comepareTo

public int compareTo(Student s) {

int num = this.name.length()-s.name.length();

***return*** name;

}

70. Map集合的特点：可以存储键值对的元素；键是唯一的，值可重复；键的存储是无序的；

Map<String,String> map = new HashMap<String,String>();

//此map的k,v均为String

由于map有一个HashMap子类，所以也可以写成：

HashMap<String,String> map = new HashMap<String,String>();

|  |  |
| --- | --- |
| **代码** | **功能** |
| map.put(String key, String value) | 添加元素。如果键是第一次添加，就直接存储并返回null；如果键已存在，则把value覆盖，并返回以前的value。 |
| map.clear() | 清除所有元素 |
| map.remove(Object key) | 按照key的值进行删除 |
| map.containsKey(Object key) | 判断是否存在这个vkey |
| map.isEmpty() | 判断是否为空 |
| map.size() | 返回map的长度 |
| map.get(Object key) | 返回该key所对应的value |
|  |  |
|  |  |

//一下子获取所有的key

Set<String> set = map.keySet();

for(String s:set) {

System.***out***.println(s);

}

//一下子获取所有的value

Collection<String> con = map.values();

for(String values:con) {

System.***out***.println(values);

}

//同步获取key-value

Set<String> set = map.keySet();

for(String s:set) {

System.***out***.println(s+"--"+map.get(s));

}

71. TreeMap: 基于红黑树，可以保证排序和唯一性。

TreeMap<String,String> tm = new TreeMap<String,String>();

Set<String> set1 = tm.keySet();

for(String s:set1) {

String ss = tm.get(s);

System.***out***.println(s+"----"+ss);

}

72. Collections 常用工具类

List<Integer> list = new ArrayList<Integer>();

list.add(xxx) //省略多个添加语句

|  |  |
| --- | --- |
| **代码** | **功能** |
| Collections.sort(list) | 对list的内容进行排序 |
| Collections.binarySearch(list, key) | 查找list中key值的索引，没有找到返回-1 |
| Collections.max(list) | 返回list中的最大值 |
| Collections.shuffle(list); | 把list中元素进行随机打乱 |
|  |  |

73. 异常处理方案：

A: try{

可能出现问题的代码;

}catch(异常名 变量){

针对问题的处理;

}catch(Exception e){} //多个catch, Exception可以匹配所有的异常，要放到最后一个catch

finally{

}

注意：一旦try里面的内容出现错误，就会结束try中的内容去匹配catch，不会再返回try

B: throws

# End