1. 编程过程 .java->.class->jvm虚拟机(跨平台)
2. Jdk(j2ee,j2se,j2me)开发工具
   1. 打包
   2. 编译
3. 平台
   1. dir 列出目录和目录文件
   2. set方式临时设置(当前窗口有效)
      1. set path[=Java\_home;%path%] 编译 //先当前后指定
      2. set classpath[=Java\_home;%path%][;] 执行 //先指定后当前（加;表示可以执行当前）
   3. 系统环境变量设置
4. 数据
   1. 注释 /\*\*/ //
   2. 类型转换、强制类型转换
   3. 转义字符
   4. 整数byte,short,int,long
      1. 10,07,0x6f,1010 进制转换
   5. 小数float,double
      1. 1.1
   6. 布尔boolean
   7. 字符char
   8. 字符串
   9. Null
   10. 数组
       1. Int[] arr; int arr[]
       2. Int[] arr = new int[]{3,1,2};
       3. Int[] arr = {3,1,2};
       4. Int[] arr = new int[3];arr[0]=3;arr[1]=1;arr[2]=2;

五、字符

* 1. a+=1;a=a+1(带强转);
  2. Instanceof 是否为类的对象

1. 符号
   1. & ｜ ^
   2. ! && ||
2. 语句
   1. If
   2. W:for break W;
3. 函数
   1. 重载
4. 封装
   1. 私有化
   2. 构造函数 给对象初始化，能重载
   3. 构造代码块 给所有对象初始化，优先于构造函数
   4. this 当前操作的对象
   5. this() 函数 构造函数间调用
   6. static 类的加载而加载，优先于对象存在，所有对象共享，直接类调用，存在方法区中
      1. 静态只能访问静态
      2. 静态多用于工具类
   7. javadoc -d myhelp -author -version hah.jar 帮助文档制作
      1. public 类，构造函数就是public
      2. public 函数
   8. 静态代码块 类的加载而执行，只执行一次，给类初始化
   9. 对象初始化过程，方法区放对象的静态成员与方法体
      1. 找到class类文件加载到内存空间
      2. 静态代码块初始化class类文件
      3. 开辟空间，分配内存地址
      4. 在堆中建立属性并进行默认初始化
      5. 属性进行显示初始化
      6. 构造代码块初始化
      7. 构造函数初始化
      8. 内存地址会给栈内存中的变量
5. 继承
   1. 关系是 is a
   2. 只能单继承，可以多层继承
   3. 聚集 has a; 聚合 球员是球队的一个;组合 手是身体的一部分
   4. 父子成员变量的关系,变量名一样的时候 this.xxx super.xxx
   5. 函数覆盖 如果要调用父函数用super.function()
   6. 父子类中的构造函数
      1. 子类构造函数默认隐式第一行是super()
      2. 父类构造函数带参的子类要手动加super(xxx)
      3. 调用本类的构造函数用this(xxx)
   7. final
      1. 修饰类 以后就不可以被继承
      2. 修饰函数 不可覆写
      3. 修饰变量 成常量，只能赋值一次
      4. 本类的内部类定义在本类的局部位置上时，只能访问该局部被final修饰的局部变量。
   8. 抽象类 abstract
      1. 不可new
      2. 只知道要做这个事情，可是无法知道要什么实现主体
   9. 接口 implements
      1. 公开的
      2. 需要全都覆盖，才能new
      3. 类可以多实现
      4. 接口之间可以继承
   10. 多态
       1. 体现：父类的引用指向子类的对象
       2. 前提：类与类之间有关系，要么继承，要么实现；要存在覆盖
       3. 弊端：只能使用父类的引用访问父类中的成员。
       4. 转型：强制转成子类型 instanceof
       5. 特点：编译时看引用有没有该可以调用的方法，有就可以通过编译；运行时，看对象所属的类中有没有该调用的方法（编译看左，运行看右）
   11. equals()
   12. toString()
   13. 内部类 一般用在组合
       1. 可以访问外部类的所有成员，外部要访问内部要建立对象
       2. 直接访问内部类：Outer.Inner in = new Outer().new Inner();
       3. 内部的this; 要用外部的this就是Outer.this
       4. 静态内部类：只能访问外部的static成员 static是成员修饰符
       5. 静态内部类的动态方法：new Outer.Inner().function();
       6. 静态内部类的静态方法：Outer.Inner.function();
       7. 当内部类中定义了静态成员，该内部类必用static
       8. 外部类中静态方法访问内部类时，内部类必用static
       9. 匿名内部类（匿名子类对象）
          1. 是简写格式
          2. 必须继承一个类或实现接口
          3. 内部类可以直接继承外部类
          4. 只能调用一次方法
          5. 要是新增了方法就不可以为类起名了（只能调用父类），调用一次，复写方法才用
   14. 异常 Throwable
       1. 严重 Error
       2. 非严重 Exception
       3. try 需要检测的代码；catch 处理异常的代码；finally 一定会执行的代码
       4. throws 用在函数上，抛出异常不处理，可抛出多个异常
       5. 自定义异常
          1. throw 用于语句中，抛出自定义异常
       6. RuntimeException 为特殊异常子类（在函数内抛出异常为RuntimeException及其子类，在函数上不用声明）
       7. 编译时被检测的异常(try)和编译时不被检测的异常(不用try RuntimeException)
       8. 使用异常的描述
       9. finally 通常用于关闭资源
       10. 子类覆盖父类时，父类抛出异常，子类覆盖的只能抛出父类异常或父类异常的子类。
       11. 父类抛出多个异常，子类在覆盖时只能抛出父类异常的子集
       12. 父类或接口的方法中没有异常抛出，子类在覆盖方法时也不可以抛出异常。只能try
   15. 包 package
       1. 包与包之间的访问 必须public类，必须public函数；子类的可以是protected
       2. 同一个类中，同一个包中，子类，不同包中；public protected default private
   16. jar -cf haha.jar packa packb
   17. 多线程
       1. Thread
       2. Runnable 接口 更方便的使用拓展
       3. start(),sleep(time),wait(),notify(),stop();
       4. 安全问题：synchronized(对象)
       5. 两个线程以上，同一个锁
       6. 同步函数，this锁
       7. 静态同步函数，Class对象锁（类名.class）
       8. 死锁
       9. 多线程间通信
          1. 进出同时进行（注意安全问题）
          2. 在同步块中，使用wait()，notify(),notifyAll()要用锁来调用（object.wait()），用相同的锁来唤醒相同的线程
          3. Lock 代替了 synchronized
          4. Condition(唤醒对方线程) 代替了 Object的wait(),notify(),notifyAll()
          5. 守护线程setDaemon():后台，在start()前使用
          6. join():要CPU执行权，执行完再释放出来，主要把主线程执行权放出来
          7. 优先级自带有的，setPriority() 设置优先级1：MIN\_PRIORITY，5：NORM\_PRIORITY，10：MAX\_PRIORIYT
          8. yield():暂停自己执行权，放出来

this() 函数只能在构造函数中使用

{} 通用，以构造方法调用要早

Javadoc

Static{} 随着类的加载而执行一次，类初始化用得多

内部类有静态方法时，内部类必须是静态

finally : catch(){return;}，在返回时执行finally

在继承覆盖中，父类没抛出异常，子类要是发生异常，只能用try处理

十一、服务器安装

apache安装、启动、停止、删除

apache 安装 httpd.exe -k install

apache 启动 net start apache2.4

apache 停止 net stop apache2.4

apache 删除 sc delete apache2.4

tomcat安装、启动、停止、删除

tomcat 安装 service.bat install

tomcat 启动 net start tomcat9

tomcat 停止 net stop tomcat9

tomcat 删除 sc delete tomcat9

mysql安装、启动、停止、删除

mysql 安装 mysqld install

mysql 启动 net start mysql

进入bin目录新建my.ini，并执行

mysqld --initialize --user=mysql --console

得到临时密码：localhost: 0\_7M+w+cnRSI

设置新密码：

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpass');

或用mysql工具:mysqladmin -u USER -p password PASSWORD

mysql 停止 net stop mysql

mysql 删除 sc delete mysql

删除注册表数据，通过regedit，删除以下几个文件【也可以在注册表里搜索mysql】： HKEY\_LOCAL\_MACHINE/SYSTEM/ControlSet001/Services/Eventlog/Applications/MySQL

HKEY\_LOCAL\_MACHINE/SYSTEM/ControlSet002/Services/Eventlog/Applications/MySQL

HKEY\_LOCAL\_MACHINE/SYSTEM/CurrentControlSet/Services/Eventlog/Applications/MySQL

HKEY\_LOCAL\_MACHINE/SYSTEM/CurrentControlSet/Services一般服务会以相同的名字(名字通常是MySQL)在这里显

删除C盘下的“C:\ProgramData\MySQL ”所以文件

启动

net start 服务名

停止

net stop 服务名

卸载

sc delete 服务名

十二、反射

java.lang.reflet

Class

Constructor

Field

Method

Modifier

反射作用

1.反编译 .class -> .java

2.通过反射访问java对象的属性，方法等

----------------------------

获取class的三种方法

Class 代表 .class二进制文件

1.Class c1 = Class.forName("A");//指向堆中对象的A类型；注意A是类的全名：包名+类名

2.Class c2 = A.class;

3.Class c3 = new A().getClass();//对象都有getClass()方法

---------------------------

通过Class类对象创建java对象

Class.forName("A");//将A.class文件装载到JVM中的过程

Class c2 = A.class //不会将A.class文件装载到JVM中的过程

Object o = c2.netInstance();//调用无参构造方法创建新实例。条件：该类要有无参构造方法A(){}

---------------------------

可变长参数：只能放在参数最后面且只能出现一次

public static void m1(int... a){} //可以等同看成数组

m1(int a) //如果可以精确匹配的方法，则调用该方法，不会再去执行可变长参数的那个方法

--------------------------

与Properties和IO联合使用

dbfile.properties ->

username=abc

user=java.util.Date //联合反射机制

Properties p = new Properties();

FileInputStream fis = new FileInputStream("dbfile.properties");

p.load(fis);

fis.close();

String v = p.getProperty("username");

一般在程序中可变的东西不要写死，推荐写到配置文件中，也叫属性文件".properties"

可以使用“空格” “冒号” “等号”，如果都有，按最前面那个

如果是中文，请用工具类 native2ascii回车后输入中文

--------------------------

Field 属性

Field[] fs = Class.forName("a").getFields();//只能获取所有public修饰的方法

Field[] fs = Class.forName("a").getDeclaredFields();//获取所有的方法

String str = Modifier.toString(fs.getModifiers()); //获取修饰字段

class type = fs.getType();type.getName();/type.getSampleName(); //获取名字

--------------------------

[Class] getDeclaredField //获取特定属性

[Field] idF.set(c,12) //设置属性值

[Field] idF.setAccessible(true) //打破封装

--------------------------

[Class] getDeclaredMethods() //获取方法

[Method] getReturnType() //获取返回类型

[Method] getParameterTypes() //获取参数列表

--------------------------

通过反射执行

Class c = Class.forName("");

Method m = c.getDeclaredMethod("login",String.class,String class);

Object retO = m.invoke(c,"admin","123");