Java基础

1、注释

书写注释是一个非常好的习惯

2、标识符

所有的标识符 都应该以字母（A-Z或者a-z），美元符（$），或者下划线（_）开始

大小写敏感

3、数据类型

要求变量的使用要严格规定，所有变量必须先定义后才能使用

Java的数据类型分为两大类

基本类型：

数值类型：

整数类型：

byte占1个字节范围：-128-127

short占2个字节范围：-32768-32767

int占4个字节范围：-2147483648-2147483647

long占8个字节范围：-9223372036854775808-9223372036854775807

浮点类型：

float占4个字节范围

double占8个字节范围

字符类型：

char占2个字节

boolean类型：占1**位**，并且只有true和false两个值（1 Byte = 8 bit）

引用数据类型：

类、接口、数组

String不是关键字、他是一个类

二进制 0b　八进制 0　十六进制 0x （0-9 A-F 16）

银行业务用bigDecimal

类型转换：

Java是强类型语言，所以要进行有些运算的时候需要用到类型转换

低----------高

byte，short，char-->int-->long-->float-->double

运算中，不同类型的数据先转化成同一类型，然后进行运算

int i= 128 ;

//强制转换(高到低)

byte b = (byte) i ;//内存溢出

4、变量

可以变化的量

每个变量必须声明数据类型

类变量 static

常量final

//短路运算

```
int c = 5

boolean d=(c<4)&&(c++<4);//(前面执行错误就不会走后面)
```

5、用户交互scanner

```
//创建一个
Scanner scanner = new Scanner(System.in);
//凡是IO流的类如果不关闭会一直占用资源
scanner.close();
```

通过Scanner类的next()(不能有空格)与nextLine()(可以有空格)方法获取输入的字符串

6、顺序结构

从上到下依次执行

7、选择结构

if 单选泽结构，if。。else。。双选择结构，if。。else if。。多选择结构，switch多选择结构case。。break。。default

8、循环结构

while避免死循环

do。。while。。至少执行一次

for（初始化：布尔值：更新）是支持迭代的一种通用结构，是最有效，最灵活的循环结构

增强for循环

9、Java方法

方法的命名规则首字母小写，驼峰规则 是用来完成特定功能的代码片段

形式参数，用来定义作用

实际参数，实际调用传递给他的参数

return 0;(终止方法)

Java都是值传递

方法的重载：名字一样但是参数不同

可变参数（不定项参数）：可变参数必须在最后面

递归：自己调用自己 边界条件：边界

10、数组

数组是一种简单的数据结构

相同类型数据的**有序**集合

定义了什么类型的数组就new什么类型的数组

数组的大小都是固定的

11、Java内存分析

堆：存放new的对象和数组

可以被所有的线程共享，不会存放别的对象引用

数组的长度是确定的，数组一旦被创建，他的大小就是不可以改变的，其元素必须是**相同类型**，不允许出现混合类型

数组中的元素可以是热河数据类型，包括基本类型和**引用类型**

数组对象本身是在堆中的

数组的使用：

for-each循环

12、Arrays类

```
Arrays.toString()//打印数组元素
```

冒泡排序：算法的时间复杂度为O(n2)

13、对象

oop Java的核心思想就是面向对象编程

面向对象思想：物以类聚，**分类**的思维模式，思考问题首先会解决问题需要哪些分类，然后对这些分类进行单独思考。最后才对某个分类下的细节进行面向过程的思索，面向对象适合处理复杂的问题，适合处理需要多人协作的问题

面向对象编程的本质是：以类的方式组织代码，以对象的组织（封装）数据

三大特性：封装、继承、多态

14、方法回顾

```
修饰符  返回值类型  方法名（。。。）
{
    //方法体
    return 返回值;
}
```

15、类和对象的关系

类是一种抽象的数据类型 需要实例化 通过关键词 new

类实例化后会返回一个自己的对象

使用new关键字创建的时候，除了分配内存空间之外，还会赋予一个初始值 以及 对类中**构造器**的使用

16、构造器

也称构造方法，是在进行创建对象的时候必须要调用的，并且构造器有以下两个特点

1、必须和类的名字相同；

2、必须没有返回类型也不能写void

使用new关键字必须要有构造器，一旦定义了有参构造，无参构造必须显示定义，构造器用来初始化值

创建对象内存分析

17、封装

属性私有，get/set

18、继承

extends关键字

在java中所有的类都默认直接或者间接继承Object类

super

super调用父类的构造方法，必须在构造方法的第一个

super必须只能出现在子类 的方法或者构造器中

super和this不能同时调用方法

19、重写

需要有继承关系，子类重写父类的方法，方法名必须相同，参数列表必须相同，修饰符：范围可以扩大，抛出的异常范围可以被缩小但是不能被扩大

20、多态

是方法的多态

父类和子类有联系

instanceof

父类转子类需要强转，子类转父类可能会丢失方法

21、static关键字详解

static关键字一句话：方便在没有创建对象的情况下进行调用

被static关键字修饰的不需要创建对象去调用，直接根据类名就可以去访问

static随着类加载一起加载

main方法中每次创建对象都会先执行匿名代码块再执行构造器，而静态代码块始终只执行了一次

22、抽象类

abstract关键字、可修饰抽象方法、抽象类

抽象类：类的抽象，可没有抽象方法，但有抽象方法的类一定要声明为抽象类。

抽象类不能被实例化，**只有抽象类的非抽象子类**可以创建对象。

抽象类存在的意义：更利于代码的维护和重用，提高开发效率。

23、接口的定义与实现（interface）

在Java编程语言中是一个抽象类型，是抽象对象的集合，对象通常以interface关键字来声明。

- 普通类：只有具体实现
- 抽象类：具体实现和规范（抽象方法）共存
- 接口：只有规范，无法自己实现

  约束和实现分离->面向接口编程

接口就是规范，定义一组规则，它的本质是契约，制定好之后大家都要遵守。

特性

- 接口是隐式抽象的，当声明一个接口的时候，不必使用**abstract**关键字。
- 接口中每一个方法也是隐式抽象的，声明时同样不需要**abstract**关键字。
- 接口中的方法都是公有的。

类在实现接口的方法时，不能抛出强制性异常，只能在接口中，或者继承接口的抽象类中抛出该强制性异常。

在实现接口的时候，也要注意一些规则：

- 一个类只能继承一个类，但是能实现多个接口。
- 一个接口能继承另一个接口，这和类之间的继承比较相似。

继承

接口的继承使用extends关键字，子接口继承父接口的方法。

多继承

- 类**不允许**多继承
- **接口允许**多继承。

接口与类相似，一个接口可以有多个方法。

接口与类的区别：

- 接口不能用户实例化对象。
- 接口没有构造方法。
- 接口中所有的方法必须是抽象方法。
- 接口不能包含成员变量，除了static和final变量。

- 接口不是被类继承，而是被类实现。
- 接口支持多继承。

24、内部类

在一个类的内部再定义一个类。

静态内部类：

```
public class Outer {
    private int id ;
    public void out() {
        System.out.println("外部类的方法");
    }

    public static void outStatic() {
        System.out.println("外部类的静态方法");
    }

    // 静态内部类
    public static class Inner {
        public void inner() {
            System.out.println("内部类的方法");
        }

        // 可以直接使用外部类的 静态！！ 属性/方法
        public void getOuterId(){
            System.out.println("内部类调用外部类 静态！！ 属性和方法");
            outStatic();
        }
    }
}
```

局部内部类：局部内部类与局部变量类似，在方法中声明。

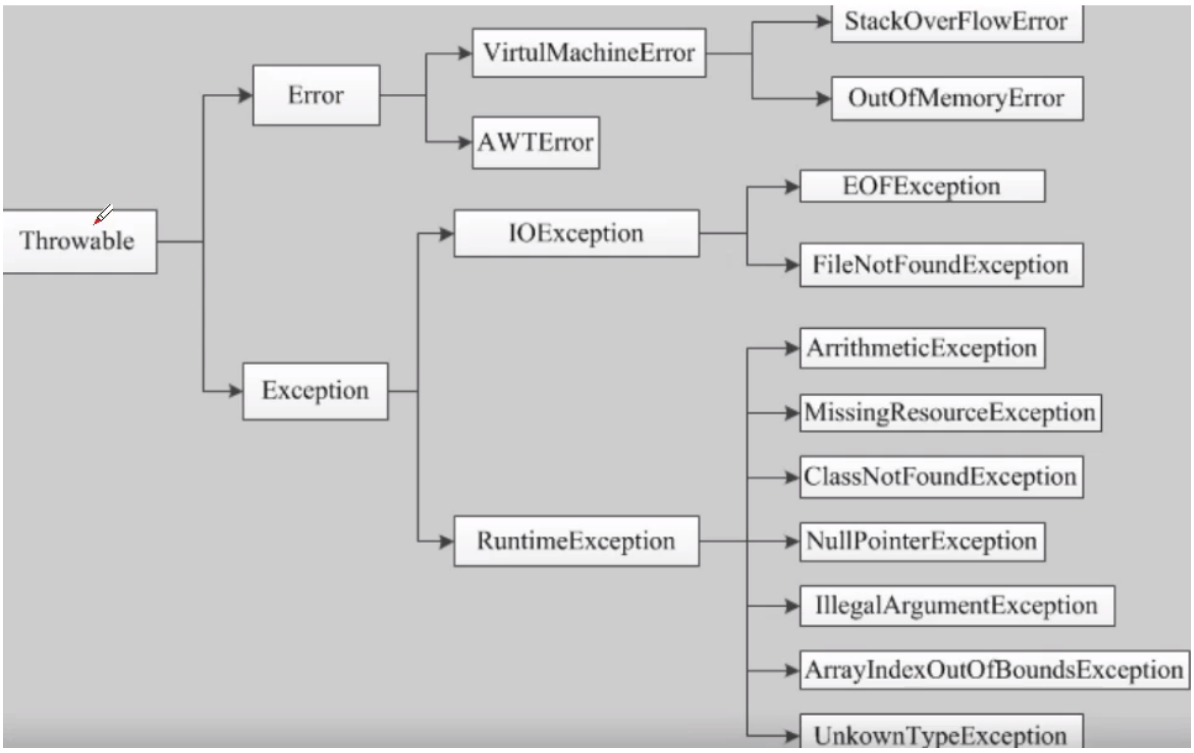
```
public class Outer {

    public void out() {
        System.out.println("进入外部类的方法");
        // 局部内部类
        class Inner {
            public void inner() {
                System.out.println("局部内部类与局部变量类似，在方法中声明");
            }
        }

        Inner inner = new Inner();
        inner.inner();
    }
}
```

25、异常处理机制

Java语言定义了许多异常类在Java.lang标准包中，主要分为Error和Exception两大类。



五个关键字try、catch、finally、throw、throws

使用 try 和 catch 关键字可以捕获异常。try/catch 代码块放在异常可能发生的地方。

try/catch代码块中的代码称为保护代码。

finally区可以不要，在IO流，资源关闭时使用。

捕获多个异常：从小到大！

IDEA快捷键：选中监控区域代码 --> Ctrl + Alt + T

EOFException文件结束异常;

抛出异常throws/throw

throws是用在方法名尾部，可以声明抛出多个异常，多个异常之间用逗号隔开。

throw是用在方法体内，主动抛出异常

自定义异常

extends Exception 自定义异常类

#####