1. Java基础
   1. Java内部类
      1. 常规内部类

常规内部类没有用static修饰且定义在在外部类类体中。

常规内部类中的方法可以直接使用外部类的实例变量和实例方法。  
 在常规内部类中可以直接用内部类创建对象

* + 1. 静态内部类

与类的其他成员相似，可以用static修饰内部类，这样的类称为静态内部类。静态内部类与静态内部方法相似，只能访问外部类的static成员，不能直接访问外部类的实例变量，与实例方法，只有通过对象引用才能访问。  
由于static内部类不具有任何对外部类实例的引用，因此static内部类中不能使用this关键字来访问外部类中的实例成员，但是可以访问外部类中的static成员。这与一般类的static方法想通

* + 1. 局部内部类

在方法体或语句块（包括方法、构造方法、局部块或静态初始化块）内部定义的类成为局部内部类。  
局部内部类不能加任何访问修饰符，因为它只对局部块有效。  
局部内部类只在方法体中有效，就像定义的局部变量一样，在定义的方法体外不能创建局部内部类的对象  
在方法内部定义类时，应注意以下问题：  
方法定义局部内部类同方法定义局部变量一样，不能使用private、protected、public等访问修饰说明符修饰，也不能使用static修饰，但可以使用final和   abstract修饰方法中的内部类可以访问外部类成员。对于方法的参数和局部变量，必须有final修饰才可以访问。  
static方法中定义的内部类可以访问外部类定义的static成员

* + 1. 匿名内部类

定义类的最终目的是创建一个类的实例，但是如果某个类的实例只是用一次，则可以将类的定义与类的创建，放到与一起完成，或者说在定义类的同时就创建一个类，以这种方法定义的没有名字的类成为匿名内部类。  
声明和构造匿名内部类的一般格式如下：  
   new ClassOrInterfaceName(){s

/\*类体\*/

}

 匿名内部类可以继承一个类或实现一个接口，这里的ClassOrInterfaceName是匿名内部类所继承的类名或实现的接口名。但匿名内部类不能同时实现一个接口和继承一个类，也不能实现多个接口。如果实现了一个接口，该类是Object类的直接子类，匿名类继承一个类或实现一个接口，不需要extends和implements关键字。  
由于匿名内部类没有名称，所以类体中不能定义构造方法，由于不知道类名也不能使用关键字来创建该类的实例。实际上匿名内部类的定义、构造、和第一次使用都发生在同样一个地方。此外，上式是一个表达式，返回的是一个对象的引用，所以可以直接使用或将其复制给一个对象变量。

* 1. final关键字
     1. final用法
     2. final的意义
  2. JAVA基类
     1. Java基类是？有哪些方法？

Java.lang.Object

对象相关{getClass()

protected clone()

toString()

equals()

hashCode()

private registerNatives()

}

线程相关{

notify()

notifyAll()

wait(空|long|long,int)

}

GC相关的{finalize()}。

* + 1. equals和==的区别？

"=="操作符的作用

1、用于基本数据类型的比较

2、判断引用是否指向堆内存的同一块地址。

equals的作用：

用于判断两个变量的内容是否相同，返回值为布尔类型

* + 1. hashCode()的作用？

hashcode方法返回该对象的哈希码值。支持该方法是为哈希表提供一些优点，例如，java.util.Hashtable 提供的哈希表。 Object.hashCode()实际上返回的是对象的引用地址，但子类可以覆盖此方法，实现不同的杂凑算法。

* + 1. hashCode()方法和equals()方法的关系

如果两个对象equals()相等，那么hashCode()必须相等；反之，则不一定。hashCode()相等，可能equals()不等，但这个概率不能太高，否则将增加HashMap冲突的可能性，而降低查找的效率。

* 1. JAVA基类
     1. 重载与重写

首先我们来讲讲：**重载(Overloading)**

（1） 方法重载是让类以统一的方式处理不同类型数据的一种手段。多个同名函数同时存在，具有不同的参数个数/类型。

重载Overloading是一个类中多态性的一种表现。

（2） [Java](http://java.chinaitlab.com/)的方法重载，就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义。

调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法, 这就是多态性。

（3） 重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回型别作为重载函数的区分标准。

然后我们再来谈谈 重写（Overriding）

（1） 父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在Java中，子类可继承父类中的方法，而不需要重新编写相同的方法。

但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。方法重写又称方法覆盖。

（2）若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类。

（3）子类函数的访问修饰权限不能少于父类的；