1. 类的定义：

类是相同或相似对象的一种抽象，是对象的一个模板，它描述一类对象的行为和状态。

类是具有相同属性和方法（行为）的对象的集合

属性是对象具有的特征。每个对象的每个属性都拥有特定值。

1. 类型变量：

局部变量：在方法、构造方法或者语句块中定义的变量被称为局部变量。变量声明和初始化都是在方法中，方法结束后，变量就会自动销毁。

成员变量：成员变量是定义在类中，方法体之外的变量。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。

类变量：也叫静态变量，类变量也声明在类中，方法体之外，但必须声明为 static 类型。

1. 定义类的时候不会为类开辟内存空间，但是一旦创建了对象，系统就会在内存中为对象开辟一块空间，用来存放对象的属性值和方法。
2. 作用域可以简单地理解为变量的生存期或者作用范围，也就是变量从定义开始到什么时候消亡。
   1. 局部变量的作用域仅限于定义它的方法内。而成员变量的作用域在整个类内部都是可见的。
   2. 同时在相同的方法中，不能有同名的局部变量；在不同的方法中，可以有同名的局部变量。
   3. 成员变量和局部变量同名时，局部变量具有更高的优先级。 大家可以编写代码验证一下。
3. 每个类都有构造方法，在创建该类的对象的时候他们将被调用，如果没有定义构造方法，Java 编译器会提供一个默认构造方法。 创建一个对象的时候，至少调用一个构造方法。比如在新建一个对象new Object()，括号中没有任何参数，代表调用一个无参构造方法（默认构造方法就是一个无参构造方法）。构造方法的名称必须与类名相同，一个类可以定义多个构造方法。
4. 如果在定义类的时候没有写构造方法，系统会默认生成一个无参构造方法，这个构造方法什么也不会做。

当有指定的构造方法时，系统都不会再添加无参构造方法了。

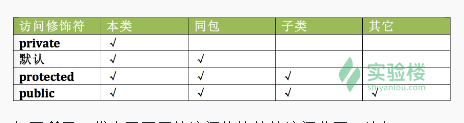
构造方法的重载：方法名相同，但参数不同的多个方法，调用时会自动根据不同的参数选择相应的方法。

1. Java 中被 static 修饰的成员称为静态成员或类成员。它属于整个类所有，而不是某个对象所有，即被类的所有对象所共享。静态成员可以使用类名直接访问，也可以使用对象名进行访问。
2. 被 static 修饰的方法是静态方法，静态方法不依赖于对象，不需要将类实例化便可以调用，由于不实例化也可以调用，所以不能有 this，也不能访问非静态成员变量和非静态方法。但是非静态成员变量和非静态方法可以访问静态方法。
3. final 修饰类，则该类不允许被继承，为最终类

final 修饰方法，则该方法不允许被覆盖（重写）

final 修饰属性：则该类的属性不会进行隐式的初始化（类的初始化属性必须有值）或在构造方法中赋值（但只能选其一）

final 修饰变量，则该变量的值只能赋一次值，即常量

1. 
2. 封装，即隐藏对象的属性和实现细节，仅对外公开接口，控制在程序中属性的读和修改的访问级别
   1. 好处？
      1. 只能通过规定的方法访问数据
      2. 隐藏类的实例细节，方便修改和实现。
3. this关键字代表当前对象。使用this.属性操作当前对象的属性，this.方法调用当前对象的方法。
4. 为什么需要继承？

如果有两个类相似，那么它们会有许多重复的代码，导致后果就是代码量大且臃肿，后期的维护性不高。通过继承就可以解决这个问题，将两段代码中相同的部分提取出来组成一个父类，实现代码的复用。

1. 继承的特点：

子类拥有父类除 private 以外的所有属性和方法

子类可以拥有自己的属性和方法

子类可以重写实现父类的方法

Java 中的继承是单继承，一个类只有一个父类

1. super关键字在子类内部使用，代表父类对象。

访问父类的属性 super.属性名

访问父类的方法 super.bark()

子类构造方法需要调用父类的构造方法时，在子类的构造方法体里最前面的位置：super()

1. 方法重载是指在一个类中定义多个同名的方法，但要求每个方法具有不同的参数的类型或参数的个数。方法重载一般用于创建一组任务相似但是参数不同的方法。
2. 方法重载有以下几种规则：

方法中的参数列表必须不同。比如：参数个数不同或者参数类型不同。

重载的方法中允许抛出不同的异常

可以有不同的返回值类型，但是参数列表必须不同

可以有不同的访问修饰符

1. 方法重写

子类可以继承父类的方法，但如果子类对父类的方法不满意，想在里面加入适合自己的一些操作时，就需要将方法进行重写。并且子类在调用方法中，优先调用子类的方法。

1. 多态是指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。多态也称作动态绑定（dynamic binding），是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。
2. 什么时候会用到抽象类

在某些情况下，某个父类只是知道其子类应该包含怎样的方法，但无法准确知道这些子类如何实现这些方法。也就是说抽象类是约束子类必须要实现哪些方法，而并不关注方法如何去实现。

从多个具有相同特征的类中抽象出一个抽象类，以这个抽象类作为子类的模板，从而避免了子类设计的随意性。

1. 接口用于描述类所具有的功能，而不提供功能的实现，功能的实现需要写在实现接口的类中，并且该类必须实现接口中所有的未实现方法。
   1. 接口不能用于实例化对象
   2. 接口中方法只能是抽象方法、default 方法、静态方法
   3. 接口成员是 static final 类型
   4. 接口支持多继承
2. 将一个类的定义放在另一个类的定义内部，这就是内部类。而包含内部类的类被称为外部类。
   1. 内部类的主要作用如下：
      1. 内部类提供了更好的封装，可以把内部类隐藏在外部类之内，不允许同一个包中的其他类访问该类
      2. 内部类的方法可以直接访问外部类的所有数据，包括私有的数据
      3. 内部类所实现的功能使用外部类同样可以实现，只是有时使用内部类更方便
      4. 内部类允许继承多个非接口类型（具体将在以后的内容进行讲解）
3. 内部类是一个编译时的概念，一旦编译成功，就会成为完全不同的两类。对于一个名为 outer 的外部类和其内部定义的名为 inner 的内部类。编译完成后出现 outer.class 和 outer$inner.class 两类。所以内部类的成员变量/方法名可以和外部类的相同。
4. 静态内部类通常被称为嵌套类。
   1. 静态内部类是 static 修饰的内部类，这种内部类的特点是：
   2. 静态内部类不能直接访问外部类的非静态成员，但可以通过 new 外部类().成员 的方式访问
   3. 如果外部类的静态成员与内部类的成员名称相同，可通过类名.静态成员访问外部类的静态成员；如果外部类的静态成员与内部类的成员名称不相同，则可通过成员名直接调用外部类的静态成员
   4. 创建静态内部类的对象时，不需要外部类的对象，可以直接创建 内部类 对象名= new 内部类();
5. 匿名内部类，顾名思义，就是没有名字的内部类。正因为没有名字，所以匿名内部类只能使用一次，它通常用来简化代码编写。但使用匿名内部类还有个前提条件：必须继承一个父类或实现一个接口。
6. 为了更好地组织类，Java 提供了包机制，用于区别类名的命名空间。

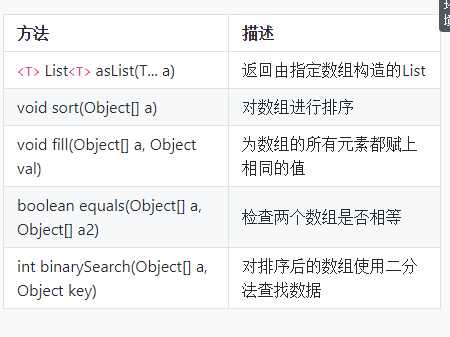
包的作用

把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。

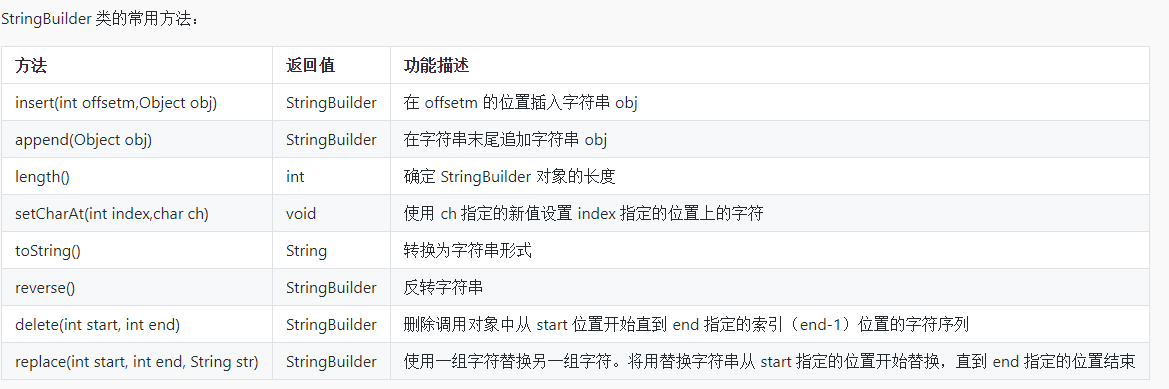
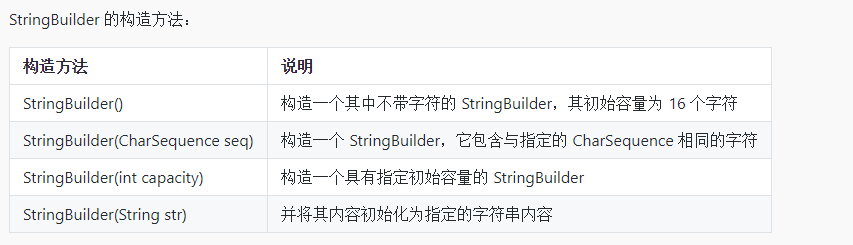
包采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。

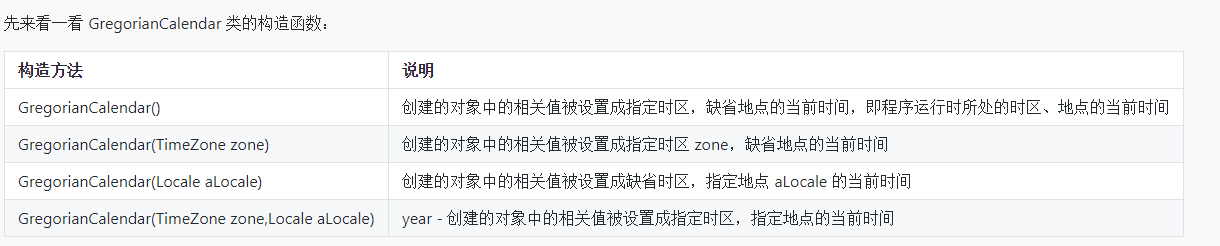
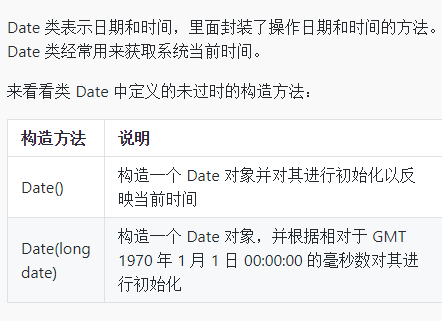
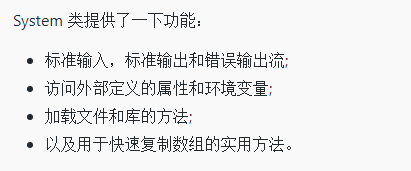
包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

1. Arrays 类包含用于操作数组的各种方法（例如排序和搜索）。还包含一个静态工厂，允许将数组转为 List。



1. StringBuilder 类是可变的。它是 String 的对等类，它可以增加和编写字符的可变序列，并且能够将字符插入到字符串中间或附加到字符串末尾（当然是不用创建其他对象的）



1. 
2. 
3. 
4. 
5. Random 类用于生成伪随机数流，在java.util包下。