Java编程思想笔记

1. bool类型的存储空间大小没有明确指定，只定义为能够取字面值true和false。
2. Java确保自定义类型内部的域会被初始化，但函数中的自动变量不会。数组是Array类型的实例，所以也会被初始化，并且Java保证不会越界访问，否则抛出异常。
3. 对char,byte,short使用移位运算，编译器会先将他们转换为int类型，操作结果也是int类型；
4. 可以使用java.util.Random类来产生随机int, float, double等随机数。也可使用Math.random()静态方法来产生一个0到1之间的随机数。
5. Java中唯一用到逗号运算的地方就是for循环中的控制表达式。
6. Java中标签起作用的唯一地方是刚好在迭代语句之前，break和continue与标签结合使用，它们就会中断循环直到标签所在的地方。Break会终止标签所在的循环，并且不会重新进入该循环，continue则跳转到标签所在循环，并开始下一轮迭代。
7. Java中类的创建和初始化时捆绑的，不能分离，这样有助于确保初始化，减少错误。
8. 静态数据只有在第一次被创建或访问时才会被初始化，且静态域先于非静态域初始化。如果静态域已经初始化，则不会再次初始化。
9. Java使用包package为类提供名字保护，使用import来是包中的类可用。Package并未将一个包下的总多.class文件组织为单个文件，为了创建独一无二的包名称和方便查找隐藏在某个目录下的类，Java利用操作系统的层次化文件结构来组织它们到一个文件目录下。环境变量CLASSPATH包含的路径作为查找包的根目录，接下来是包名的子目录。
10. Java中，没有使用package指定所属包的类将自动看作是隶属于该目录的默认包，于是他们为处于该目录下的其他文件都提供了包访问权限。不过不要这么用。
11. Java将访问控制权限的边界划在数据类型的内部，两个重要的原因是：一、可以设定客户程序员可以使用和不可以使用的权限；二、将接口和实现进行分离。这样数据类型保留了非public操作的修改权限，而不必担心客户程序员会受到影响。
12. 类如果只提供private构造器，则可以阻止任何人创建该类的对象，但是在该类的静态方法内部创建，并返回。这样可以构建使用静态工厂方法和创建单例模式的类。这种类无法子类化。
13. 继承中对于有参数的基类构造函数，必须使用super关键字调用基类构造函数，并置于派生类的构造函数中第一个语句。继承的初始化顺序是，由基类向派生类扩散。
14. 对于对象内存回收之外的清理活动，必须确保正确的清理，通常的实现方法是显式编写这样的清理函数，并在清理函数最后一个语句调用基类的清理函数，然后将该清理函数放在finally子句之中以确保清理动作不会因异常而没被执行。
15. Java中，派生类重定义的方法不会屏蔽基类中的其它版本（与C++不同）。可以在重定义的方法之前加上@override标签来防止意外重载而非覆盖基类的方法。
16. 继承与组合的选择：组合通常是想在新类中使用现有类的功能而非它的接口，并通常将现有类作为新类的私有域；继承则是使用现有类，并开发它的一个特殊版本，现有类为新类提供方法，并且新类是现有类的一种类型，java默认使用动态绑定，故可以使用基类引用来调用所期望的派生类方法。应该慎用继承，只有确定需要从新类向现有类进行向上类型转换时才使用继承重用。
17. 如果final数据在定义时初始化为基本类型的常量值，则该final数据是编译时常量。Final方法不能被覆盖；final类禁止继承。
18. 构造器内部的不要调用非final函数（private函数自动是final的），否则可能会在构造器内部引起多态调用——调用的方法访问了正在构造类的为初始化的域，这会导致难以发现的错误。
19. 多态对private方法、final方法、静态方法（构造函数默认为静态的）和域是无效的。
20. 继承类的初始化顺序：一、在任何事情发生之前，将分配给对象的存储空间初始化为二进制0；二、调用基类构造器；三、按生命顺序调用成员初始化方法；四、调用派生类的构造器主体。
21. 协变返回类型是指派生类中被覆盖的方法可以返回基类方法的返回类型的派生类。
22. 创建一个能够根据所传递参数对象不同而具有不同行为的方法,成为策略设计模式。
23. 接口最大的作用是解耦：如果一个方法的参数是一个类型，那么能传入的对象只能是该类型及其派生类对象；但是如果方法的参数的接口，就能传入所有实现了该接口的类的对象。这就使得方法与参数类之间的耦合更小，具有更高的灵活性。
24. 复用代码的一种常用方式就是客户程序员遵循某个接口来实现他们自己的类，这样他们的类就能被以该接口为参数的方法所处理，重用了已有的方法代码。但是如果客户程序员想使用已有类（自己不能更改）也实现了某个接口，就该使用适配器设计模式，也即是代理方法。
25. 利用接口可以实现多重继承。