## JAVA复习

1. 并行与并发
   1. 并行：多个CPU同时执行多个任务，比如多个人同时做不同的事情
   2. 一个CPU（采用时间片）同时执行多个任务，比如：秒杀，多个人做同一件事情
2. 线程的创建
   1. 继承Thread类
   2. 创建一个继承于Thread的子类
   3. 重写Thread类的Run()---将此线程执行的操作声明在run()中
   4. 创建Thread类子类的对象
   5. 通过此对象调用start()
3. 线程创建2：实现Runnable接口
   1. 创建一个实现了Runnable接口的类
   2. 实现类去实现Runnable中的抽象方法：run()
   3. 创建实现类的对象
   4. 将此对象作为参数传递到Thread类的构造器中，创建Thread类的对象
   5. 通过Thread类的对象调用start()
4. 线程优先级等级
   1. 最高优先级：MAX-PRIORITY:10
   2. 最低优先级：MIN-PRIORITY:1
   3. 默认优先级：NORM-PRIORITY:5
5. 线程常用方法
   1. 中断线程：interrupt()
   2. 将该线程标记为守护线程或用户线程：setDaemon(boolean an)
   3. 当A线程执行到B线程的join方法时A线程就是等待，等B线程都执行完，A线程才会执行：join()
   4. 返回该线程的字符串表示形式，包括线程名称，优先级和线程组：toString()
   5. 暂停当前正在执行的线程，并执行其他线程：yield()
6. IDEA中的Project相当于eclipse中的工作区（work area），Module相当于eclipse中的Project
7. 线程的生命周期：新建-->就绪-->运行-->阻塞-->死亡
8. 重写方法抛异常不能比父类范围大，并且父类没有抛异常子类也不能抛异常，只能使用try{}catch{}
9. 同步方法的总结
   1. 同步方法仍然涉及到同步监视器，只是不需要我们显式的声明
   2. 非静态的同步方法，同步监视器是this

静态的同步方法，同步监视器是：当前类本身 类名.class

1. 设计模式
   1. 创建型模式，共五种：工厂方法模式，抽象工厂模式，单例模式，建造者模式，原型模式。
   2. 结构型模式，共七种：适配者模式，装饰器模式，代理模式，外观模式，桥接模式，组合模式，享元模式
   3. 行为型模式，共十一种：策略模式，模板方法模式，观察者模式，迭代子模式，责任链模式，命令模式，备忘录模式，状态模式，访问者模式，中介者模式，解释器模式
2. 单例模式
   1. 饿汉式：
      1. 缺点：对象加载事件过长
      2. 好处：线程安全
   2. 懒汉式：
      1. 好处：延迟对象的创建
3. 类的成员
   1. 代码块：
      1. 用来初始化类，对象
      2. 如果有修饰的话，只能使用static
      3. 静态代码块随着类的加载而执行，只会执行一次
      4. 非静态代码块随着对象的调用而加载,每创意一个对象，执行一次
      5. 静态代码块优先于非静态代码块执行
      6. 静态代码块不能调动非静态结构
4. 对属性可以赋值的位置，和顺序
   1. 默认初始化
   2. 显示初始化 / (5) 在代码块中赋值
   3. 构造器中初始化
   4. 有了对象之后，可以通过“对象.属性”或者“对象.方法”的方式进行赋值
   5. 在代码块中赋值
5. JDK8：除了定义全局常量和抽象方法之外，还可以定义抽象方法，默认方法
6. 接口
   1. 接口中定义的静态方法，只能通过接口名调用
   2. 通过实现类的对象，可以调用接口中的默认方法，如果实现类重写了接口中的默认方法，调用时，仍然调用的是重写后的方法
   3. 如果子类（或实现类）继承的父类和实现的接口中声明了同名同参数的默认方法，那么子类在没有重写此方法的情况下，默认调用的是父类中的同名同参数的方法-->类优先原则
   4. 如果实现类实现了多个接口，而这多个接口中定义了同名同参数的默认方法，那么在实现类没有重写此方法的情况下，报错-->接口冲突，这就需要我们在实现类中重写方法
   5. 如何在子类（或实现类）的方法中调用接口中的默认方法：接口名.super.默认方法名
7. 类的内部成员
   1. Java中允许将一个类A声明在另一个类B中，则类A就是内部类，类B称为外部类
   2. 内部类的分类：成员内部类（静态，非静态）vs局部内部类（方法内，代码块内，构造器内）
   3. 内部类
      1. 成员内部类
         1. 一方面，作为外部类的成员：
            1. 调用外部类的结构
            2. 可以被static修饰
            3. 可以被四种不同的权限修饰
         2. 另一方面，作为一个类
            1. 类内可以定义属性，方法，构造器等
            2. 可以被final修饰，表示此类不能被继承，言外之意，不使用final就可以被继承
            3. 可以被abstract修饰
   4. 关注三个问题
      1. 如何实例化成员内部类的对象
      2. 如何在成员内部类中区分调用外部类的结构
      3. 开发中局部内部类的使用
   5. 注意点：在局部内部类的方法中如果要调用局部内部类所在的方法中的变量的话，该变量需要声明为final，jdk 7 及以前的版本：要求局部变量显式的声明为final，jdk8之后的版本可以省略final的声明
8. 静态修饰的类属性跟方法最先被加载
9. Try\_catch\_finally:真正将异常处理掉了，throws方式只是将异常抛给了方法的调用者，并没有真正将异常处理掉
10. 死锁的问题
    1. 不同的线程分别占用对方需要的资源不放弃，都在等待对方放弃自己需要的同步资源，就形成了线程的死锁
    2. 出现死锁后，不会出现异常，不会出现提示，只是所有的线程都出现阻塞状态，无法继续
    3. 解决方法：
       1. 专门的算法原则
       2. 尽量避免同步资源的定义
       3. 尽量避免嵌套同步
11. synchronized与Lock的对比
    1. Lock是显式锁（手动开启手动关闭，不能忘记关闭锁），synchronized是隐式锁，出了作用域自动释放
    2. Lock只有代码块锁，synchronized有代码块锁和方法锁
    3. 使用Lock锁，JVM将花费较少的时间来调度线程，性能更好，并且具有更好的扩展性，提供更多的子类
    4. 优先使用顺序：Lock-->同步代码块（已经进入了方法体分配了相应的资源）-->同步方法（在方法体之外）
12. Sleep不会释放锁，wait会释放锁，notify只会唤醒一个，并且是优先级最高的内个，notifyAll会唤醒所有被wait的线程
13. Notify，notifiyAll跟wait只能用在同步代码块跟同步方法中，这三个方法的调用者必须是同步代码块或同步方法中的同步监视器，否则会出现异常，这三个方法定义在object中
14. 编码：字符串 --> 字节 （看得懂 --> 看不懂的二进制数据）

解码：编码的逆过程，字节 --> 字符串 （看不懂的二进制数据 --> 看得懂）

1. 字符串拼接结论
   1. 常量跟常量的拼接结果在字符串常量池，且常量池中不会存在相同内容的常量
   2. 只要其中有一个是变量，结果就在堆中
   3. 如果拼接的结果调用intern()方法，返回值就在常量池中
2. 日期
   1. 格式化：日期 ---> 字符串
   2. 解析：格式化的逆过程，字符串 ---> 日期
3. LocalDateTime:比较常用
4. Java比较器
   1. Comparable接口
   2. Comparator接口
   3. 默认都是从低到高
5. enum类继承与java.long.enum类
6. 自定义异常类继承Exception
7. 框架 = 注解 + 反射 + 设计模式
8. 元注解：对现有注解进行解释的注解
9. 异常
   1. Throw：生成一个异常对象，并抛出，使用在方法内部， 自动抛出异常对象
   2. Throws：处理异常的方式，使用在方法声明处的末尾，try-cath-finally，上游排污，下游治污
10. 向collection接口的实现类的对象中添加数据obj时，要求obj所在的类要重写equals（）。
11. Foreach遍历集合本质上还是调用了迭代器iterator
12. Set接口中没有定义额外新的方法，使用的都是collection中声明的方法
13. Hashset底层：数组+列表结构
14. Comepare:是当前类跟obj作比较
    1. 我的测试，当当前类跟obj作比较时，为降序排序
    2. 当obj跟当前类作比较时，为升序排序
15. 临界值，超过多少个就开始扩容，默认的加载因子
16. 数据结构和算法
    1. 数据结构只是静态的描述了数据元素之间的关系
    2. 高效的程序需要在数据结构的基础上设计和选择算法
    3. 算法是为了解决实际问题而设计的，数据结构是算法需要处理的问题载体
17. Collection是集合list和set的接口，collections是一个操作set，list和map的等集合的工具类
18. 如果定义了泛型类，实例化没有指定类的泛型，则认为此泛型为object类型，如果定义了泛型，建议实例化时使用泛型
19. 泛型方法：
    1. 在方法中出现了泛型的结构，泛型参数与类的泛型参数没有任何关系
    2. 泛型方法所属的类是不是泛型结构类都没有关系
    3. 泛型方法可以被声明为静态static的，因为泛型参数是在调用方法时确定的，并非在实例化类时确定
20. 泛型在继承方面的体现，虽然类A是类B的父类，但是G<A> 和G<B>二者不具备子父类的关系，二者是并列关系。补充：类A是类B的父类，A<G> 是 B<G>的父类
21. IO流的分类
    1. 节点流（文件流）
    2. 缓冲流
    3. 转换流
    4. 标准输入输出流
    5. 打印流
    6. 数据流
    7. 对象流
    8. 随机存储流
22. 相对路径：相较于某个路径下，指明的路径，绝对路径：包含盘符在内的文件或文件目录的路径
23. 当file文件存在于路径所指向的位置的话，则会显示文件的值：字节，修改时间等等，如果没有创建则显示的是 初始默认值
24. 序列化
    1. 序列化过程：将内存中的java对象保存到磁盘中或通过网络传播出去
    2. 反序列化过程：将网络或磁盘中的二进制读取到内存中的一个java对象
    3. 继承serializable之后必须要声明serialVersionUID常量
25. Class类
    1. Class实例对应着加载到内存中的一个运行时类
26. JDK9
    1. 接口中的静态方法只能由接口自己调用
    2. 接口的实现类不能调用接口的静态方法
    3. 接口的私有方法不能再外部调用
    4. 接口中的静态方法不能被重写
    5. 接口中的默认方法可以不重写
27. 钻石操作符与匿名内部类在java 8 中不能共存，在java9中可以