====2020/08/13====

1. 状态：

    a. 线程：创建，就绪，运行，阻塞，死亡；New，Runnable，Running，Blocked，Dead；

    b. 进程：创建，就绪，运行，阻塞，死亡；

2. blocked，waiting，waiting-timed：

    a. 阻塞状态等待事件发生，转换为就绪状态；

    b. blocked：等待monitor lock；

    c. waiting：等待notify；

    d. waiting-timed：等待notify或者定时器数到0；

3. 文件描述符：fd（file descriptor）；linux中每打开一个文件，都有一个小的整数与之对应，就是文件描述符，0标准输入，1标准输出，2标准保存输出，<输入重定向符，>输出重定向符；

4. IO多路复用模型：

    a. 多个io复用一个线程；

    b. select：轮询；

    c. poll：最大连接数无上限；

    d. epoll：事件通知。

5. select：

    a. select函数，返回就绪的文件描述符fd的个数，找到就绪的文件描述符，保存在fdset中。每次调用select，需要轮询一遍fd，查看就绪状态；且支持的最大fd数量有限，32位系统默认是1024；需要把fdset拷贝从用户态拷贝到内核态，开销大；

    b. poll：不存在最大文件描述符限制；

c. epoll：事件通知的方式。包含epoll_create，epoll_ctl，epoll-wait三个方法；

i. epoll_create：创建一个句柄（类似于指针），占用一个fd；

ii. epoll_ctl：注册监听事件，把所有fd拷贝进内核一次，并为每一个fd指定一个回调函数，不需要每次轮询遍历fd；当fd就绪，会调用回调函数，把就绪的文件描述符和事件加入一个就绪链表，并拷贝到用户空间内存，应用程序不用亲自从内核拷贝。

iii. epoll_wait：监听epoll_ctl中注册的fd，在就绪链表中查看有没有就绪的fd，不用遍历fd。两种工作方式；

1. 水平触发：默认工作方式，epoll_wait检测到fd就绪，通知程序，不会立刻处理，下次epoll还会通知；

2. 边缘触发：epoll_wait通知会被立刻处理，下次不会通知；

6. hashset和treeset

====2020/08/13====

1. 24点问题：

a. what：4个数，加减乘除得到24，返回true，否则false。

2. 文本传输：

a. 服务端：

i. 创建服务器套接字并等待客户请求；

ii. 收到请求并建立连接；

iii. 按行读取客户端数据并写入到文件l；

iv. 完成后向客户端发送响应。

b. 客户端：

i. 创建套接字；

ii. 按行读取文本文件并发送；

====2020/08/12====

1. 适配器模式：

a. 定义：将一个类的接口转换成客户端需要的另一个接口，主要目的是**兼容性**，让原本接口不匹配的两个类协同工作。

b. 角色：目标接口，被适配者，适配器。

c. 通过适配器类，继承源角色，实现目标角色的接口，在适配器类中进行具体实现，达到适配的目的。

d. 类、对象、接口 适配器：

i. 类适配器：通过继承来实现，继承源角色，实现目标目标角色接口；

ii. 对象适配器：适配器拥有源角色实例，通过组合来实现适配功能，持有源角色，实

现目标接口；

iii. 接口适配器：接口中有多个方法，用抽象类实现这个接口和方法，在创建子类时，只需要重写其中几个方法就行。

2. 装饰者模式：

a. 定义：以透明动态的方式来动态扩展对象的功能，是继承关系的一种代替方案。

b. 角色：抽象类，抽象装饰者，装饰者具体实现。

c. 一个抽象类有A方法，定义一个类作为抽象装饰者继承该抽象类，再创建具体装饰者类继承抽象装饰者类，并对其进行方法扩展，不用改变原来层次结构。

3. 适配器模式，装饰者模式，外观模式，区别：

a. 适配器模式将对象包装起来改变其接口；

b. 装饰者模式包装对象扩展其功能；

c. 外观模式保证对象简化其接口。

4. 代理模式：

a. what：**给某个对象提供一个代理对象，由代理对象控制该对象的引用。**

b. why：

**i. 中介隔离作用**，在客户类和委托对象之间，起到中介作用；

**ii. 开闭原则**，增加功能，对扩展开发，对修改封闭，给代理类增加新功能。

c. where：需要隐藏某个类，使用代理模式。

d. how：代理角色、目标角色、被代理角色，**静态代理，动态代理**；

i. 静态代理：

1. what：代理类创建实例并调用方法，在程序运行前，代理类已经创建好了；

2. why：

a. 优点：开闭原则，功能扩展；

b. 缺点：**接口发生改变，代理类也需要修改。**

3. where：需要代理某个类。

4. how：代理对象和被代理对象实现相同接口，通过调用代理对象的方法来调用目标对象。

ii. 动态代理：

1. what：程序运行时通过反射机制动态创建代理类；

2. why：

a. 优点：不需要继承父类，利用反射机制；

b. 缺点：目标对象需要实现接口。

3. where：代理某个类；

4. how：实现InvocationHandler接口，重写invoke方法，返回值时被代理接口的一个实现类。

iii. Cglib代理：

1. what：通过字节码创建子类，在子类中采用方法拦截来拦截父类的方法调用，织入横切逻辑，完成动态代理。

2. why：

a. 优点：不需要接口；

b. 缺点：对final无效。

3. where：不需要接口，代理。

a. SptingAOP中，加入容器的目标对象有接口，用动态代理；

b. 目标对象没有接口，用CGLib代理。

4. how：字节码。

5. 接口和抽象类：

a. 相同点：

i. 不能直接实例化；

ii. 包含抽象方法，则必须实现。

b. 不同点：

i. 继承extends只能支持一个类抽象类，实现implements可以实现多个接口；

ii. 接口不能为普通方法提供方法体，接口中普通方法默认为抽象方法；

iii. 接口中成员变量是public static final，抽象类任意；

iv. 接口不能包含构造器、初始化块。