Java

对象串行化
你有一个类，只要实现序列化接口（implements Serializable）
Eclipse 强制要求需要有版本号
保持你的类和读的数据是同一个版本（数据文件有相应部份）

反串行化：
申请内存 → 从数据文件中读数据（填入内存） 无构造！不是初始状态，不必构造（但会变成一块0）

关键字：transient（易失的）不能被从数据文件中填入内容。

反串行化需要对应的类（版本号也相同）能读取流的数据
java -cp ./bin:./tt exp.Main

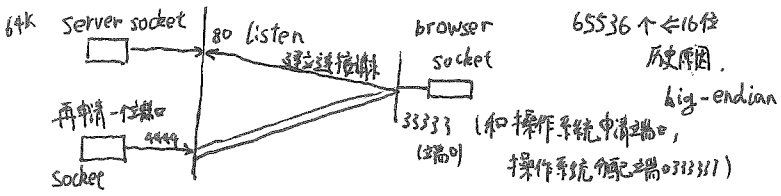
到两个目录下各自的exp文件里去找

网络：

两种形式的地址：DNS（Domain Name Service） 域名≠机器名
The "dotted quad" form
InetAddress.getByName() ⇐ 他的构造函数是私有的，只能用静态方式

TCP是有连接的协议。要先建立端对端连接。传送中校验数据
UDP是无连接的。不管对方是否存在可以发送。不需要保证顺序

每台计算机上有个端口，port。
socket — 原型 袜子
TCP 通信 — 非对等过程。一定有确定的服务端和客户端 pop-11



Server socket
申听在固定端口，等待client连接的对象。

需 import java.net.ServerSocket
import java.net.Socket
Socket s = ss.accept();
s.getOutputStream();
构造成功就连接成功

⌘：telnet 工具（不整）
telnet localhost 14500
nc localhost 14500
nc -l 14500
Listen

SeverSocket 可以多个客户端
accept 以后马上交给新线程，主线程负责监听

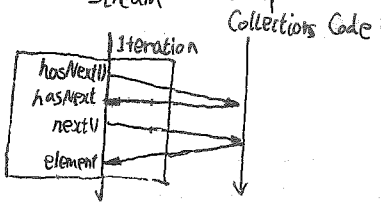
class <? extends xxx>
? 任意 extends xxx 的类

Lamda 表达式
Event handle
code for data — The object passed to the target is code.

btn.addActionListener(event) → System.out.println("ok"));
这句话并没有转成类
Lambda 的值是一个类，类型看等式左边
Runnable run = () → System.out.println("ok");
BinaryOperator<Long> add = (x,y) → x+y;
add.apply(100, 200);
执行
表达式的结果会做为函数的值返回

① Function interface
② type inference
③ Predicate (论断)
④ A lambda expression is a function

Stream 流式计算




```
long cnt = list.stream()
        .filter(x->{
            return x.startsWith("A");
        })
        .count();
```

filter() returns Stream → lazy way
count() returns int → eager way
collect() eager way
map() lazy way

filter 出一个就交给下一个

对每个element做filter→map……

```
.filter(x->{
    return x.startsWith("A");
})
.map(system.out.println("map"+x);
    return x.toLowerCase());
,
.count()
```

流的遍历和自己会并行计算

Lambda 不能阻止副作用。
@FunctionalInterface

接口可以没有函数
接口无法定义声明实例字段

适配器模式 采用代理而非继承时会有更大的灵活性
系统并不知道门面，门面是他的客户

合成模式 一定能把自己放进配这个容器中

String类不可变 对象
好处：多个值相同的String类对象只需一个 享元

effectively final
↓
一个可以被内部类引用的本地变量
必须是final（Java 8之前）

代码重用：
继承、组合、模板

GOF开始看

将对象的构建逻辑转移到外部