**仿****airdrop软件**

一、功能分析

airdrop用于在多台设备之间分享文件，仿造airdrop需要做到的关键功能是实现通过网络的文件复制功能。

二、已有的项目

（1）通过java中的socket实现客户端和服务端之间的字符串传递。

（2）java文件复制。

三、设计思路

将java文件复制项目于java中的socket通讯相结合。具体来讲，文件复制中有FileInputStream fis = **new** FileInputStream(from);

FileOutputStream fos = **new** FileOutputStream(to);

作为输入和输出流，在Socket通讯中有

InputStream is = socket.getInputStream();

OutputStream os = socket.getOutputStream();

作为输入和输出流，这两种流继承于同一种父类，具有相似的使用方法，于是文件的发送方需要FileInputStream fis = **new** FileInputStream(from);

和OutputStream os = socket.getOutputStream();这两种流发送文件。

接收方需要InputStream is = socket.getInputStream();

和FileOutputStream fos = **new** FileOutputStream(to);这两种流接收文件。

四、关键功能

（1）指定自身是发送方还是接收方，选择文件路径向接收方发送文件。

（2）根据端口确定发送方和对应的接收方。

五、关键代码

（1）、接收部分

// 从服务端接收文件

InputStream is = socket.getInputStream();

FileOutputStream fos = **new** FileOutputStream(dir);

**byte**[] buff = **new** **byte**[8192];// 字节数组

//获取文件逻辑长度

is.read(buff);

String str = **new** String(buff, 0, 4);// 字节数组转换为字符串

**int** len = Integer.*parseInt*(str);

System.***out***.println(str);

//接收文件的大部分数据

**for**(**int** i=0;i<len;i++) {

is.read(buff);

fos.write(buff);

}

//获取文件尾长度

is.read(buff);

str = **new** String(buff, 0, 4);

System.***out***.println(str);

**int** size = Integer.*parseInt*(str);

System.***out***.println("从服务端接收的文件尾的长度为"+size);

//接收文件尾

is.read(buff);

fos.write(buff,0,size);

（2）发送部分

/\*\*

\* 获取服务端的输出流，往客户端传输文件

\*/

OutputStream os1 = socket1.getOutputStream();//字节流

FileInputStream fis = **new** FileInputStream(dir1);//文件流

**int** len1 = fis.available()/8192;

os1.write((len1+"").getBytes());

//发送文件大部分数据

**for**(**int** i=0;i<len1;i++) {

fis.read(buff1);

os1.write(buff1);

}

//发送文件的尾部

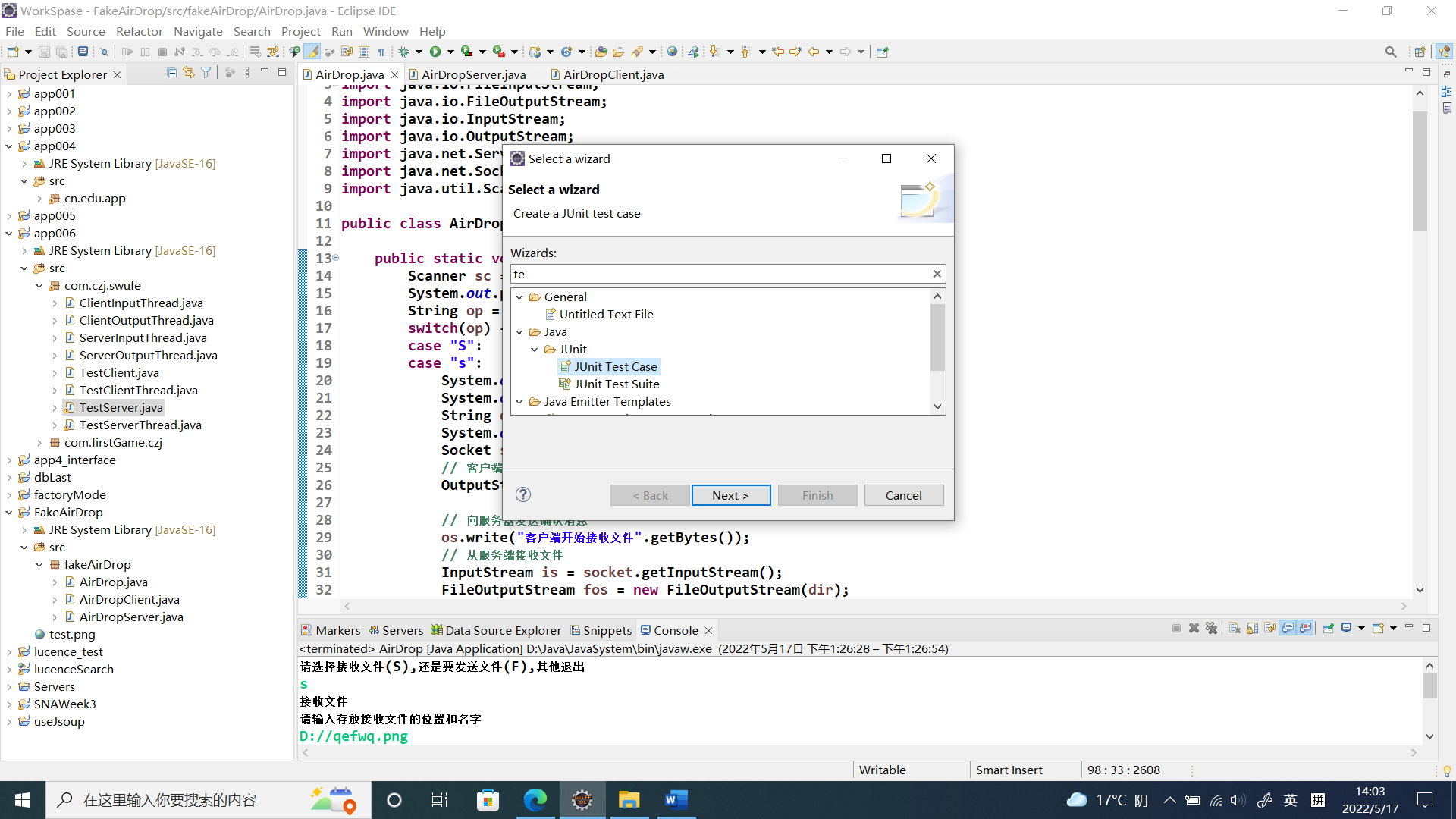
**int** size1 = fis.read(buff1);

os1.write((size1+"").getBytes());

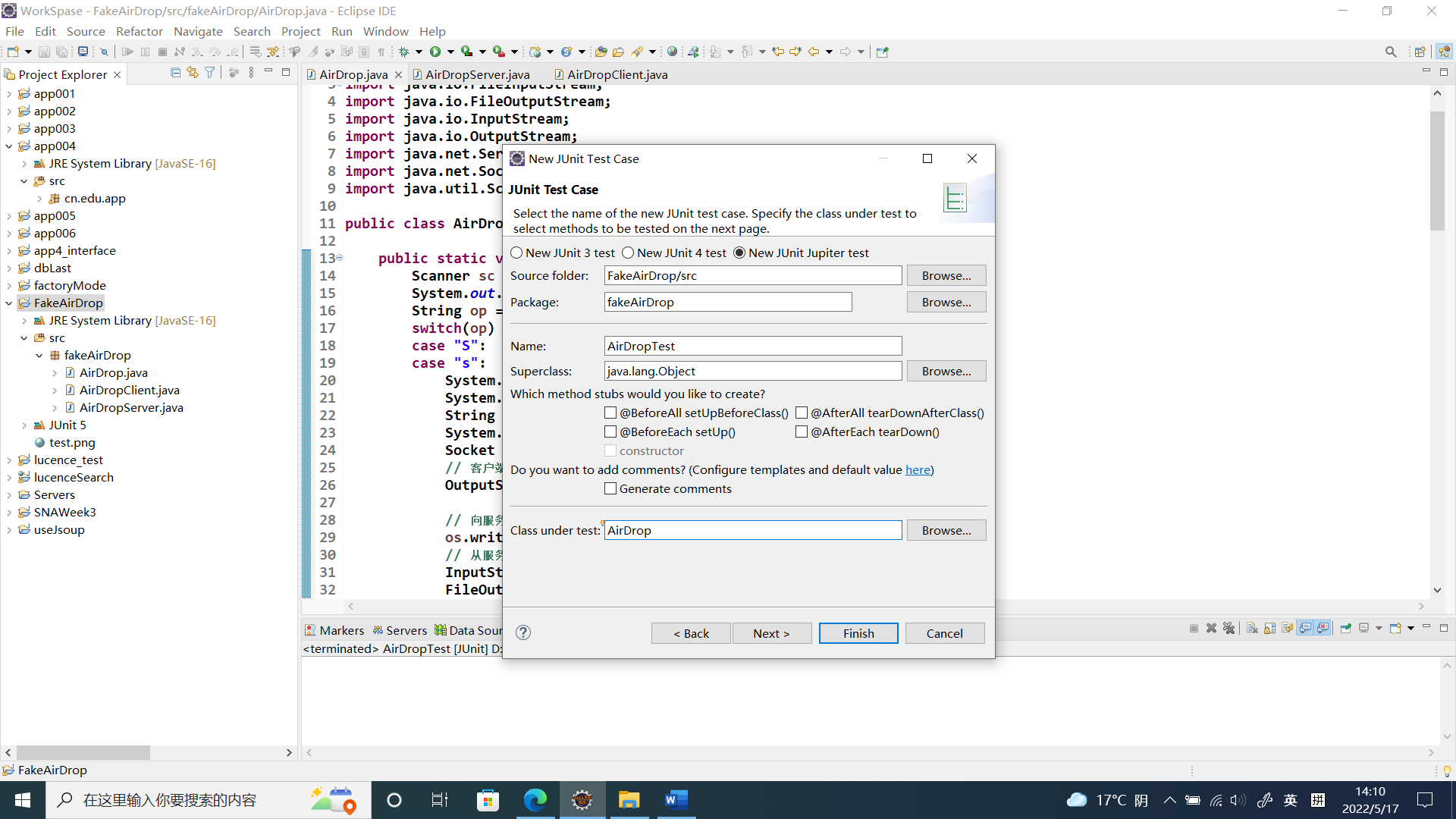
os1.write(buff1,0,size1);

六、单元测试

（1）首先点击文件，新建，选择Junit Text Case创建一个测试

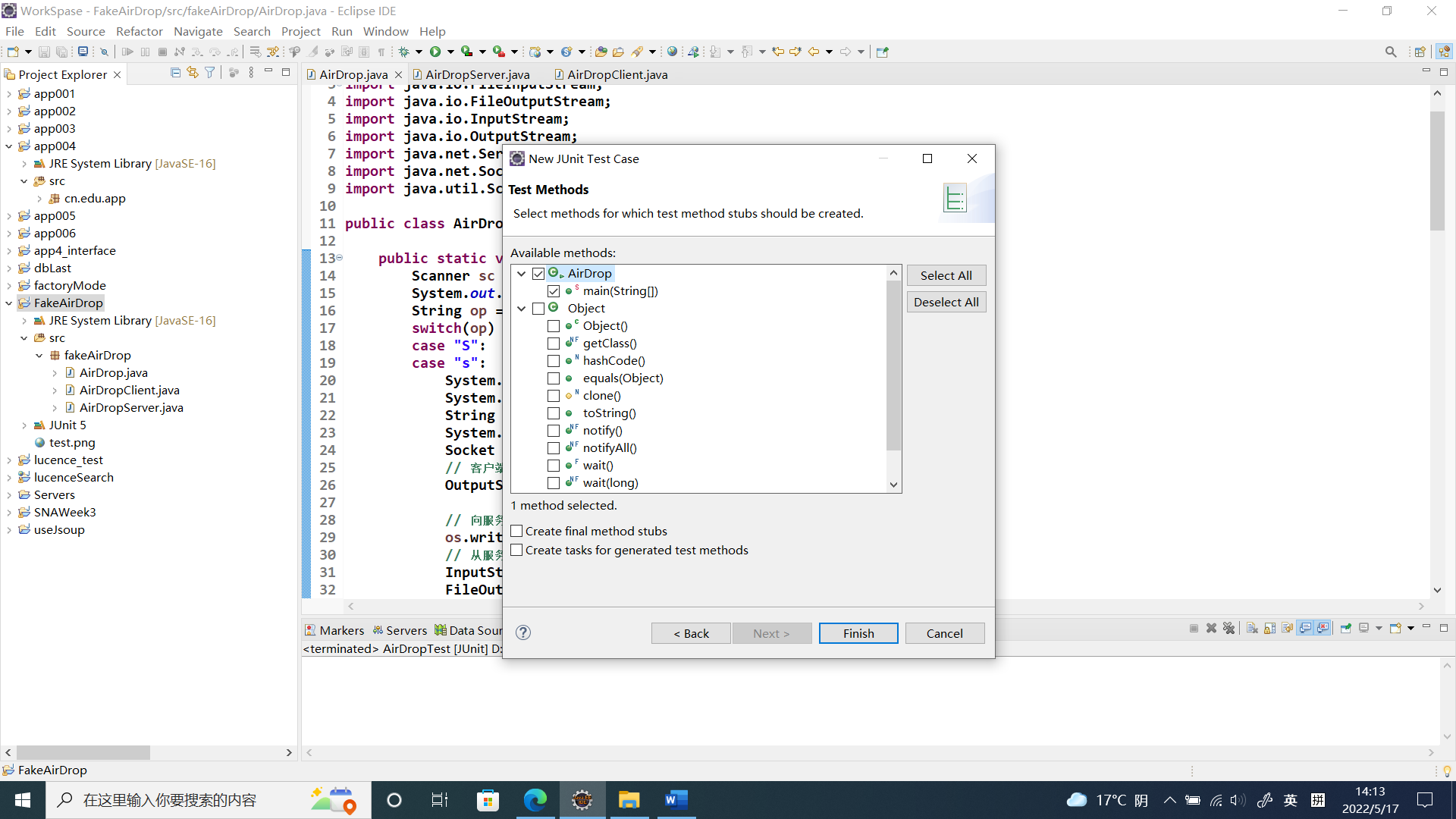


（2）填写测试类的名字，和选择待测试的类





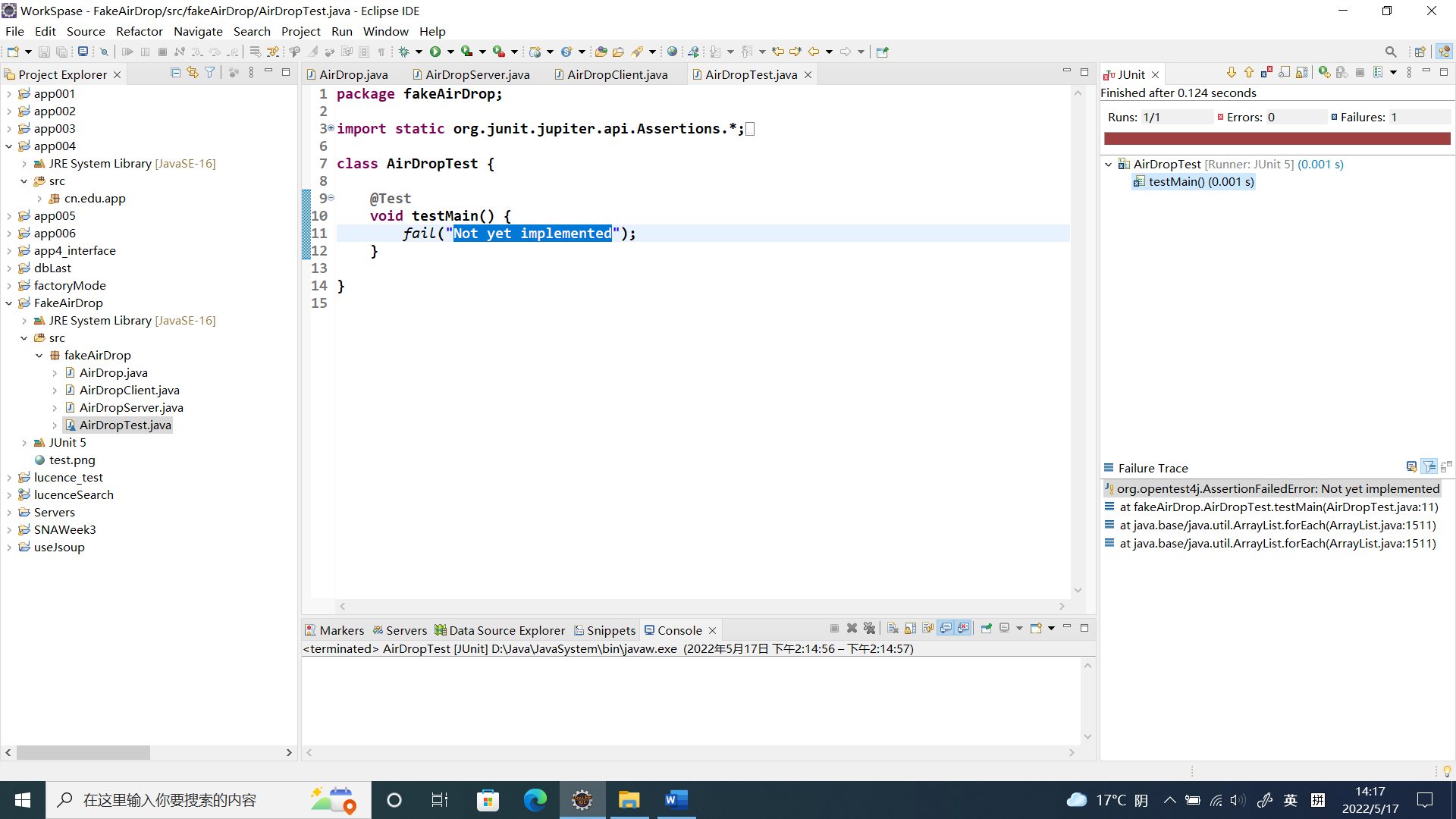
（3）选择要测试的类和方法





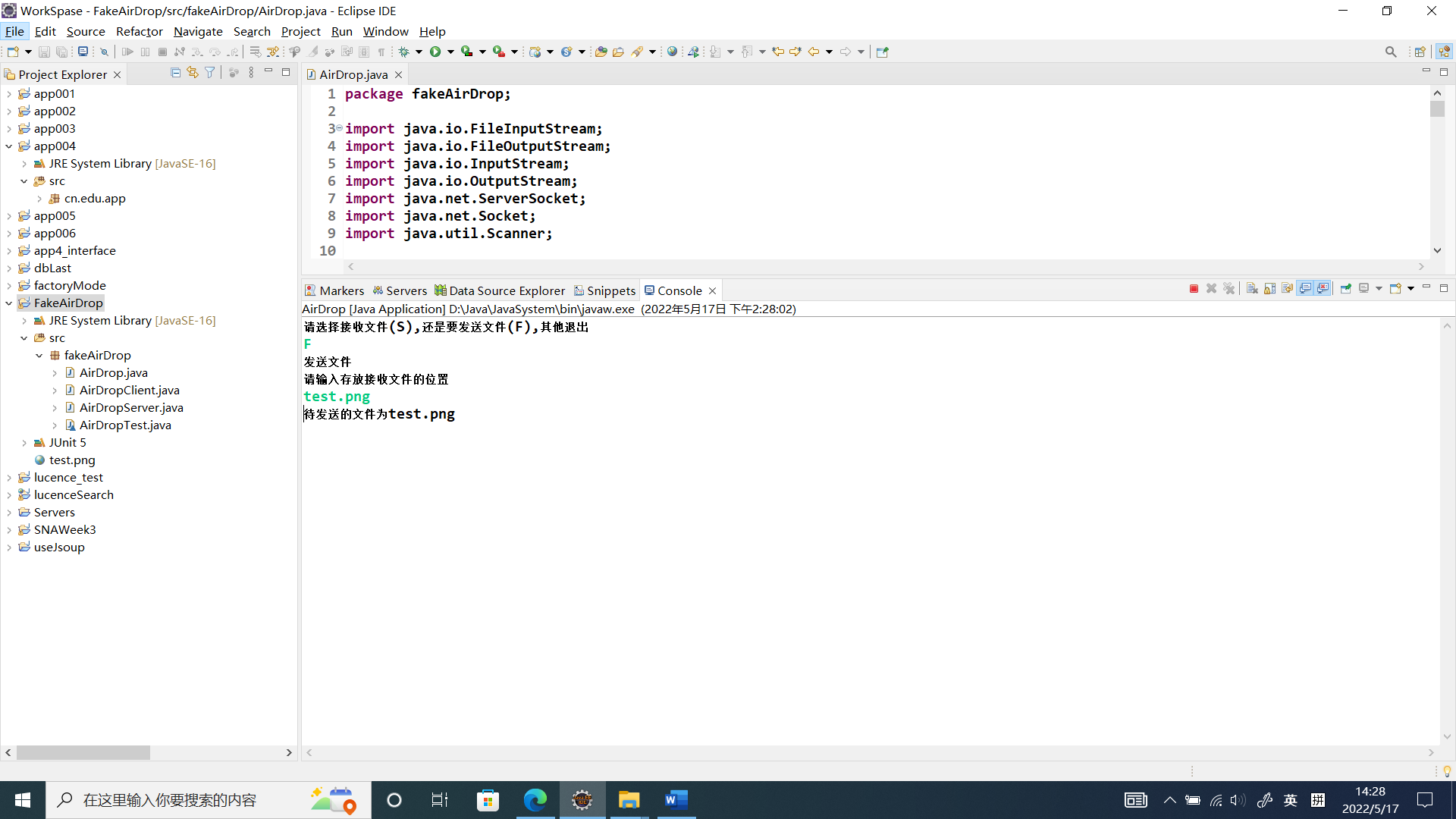
（4）以JUnit Test的方式运行结果如下。

单元测试没有成功，因为单元测试是对软件中的最小可测试单元进行检查和验证。直接测试main方法，不会有好的结果。

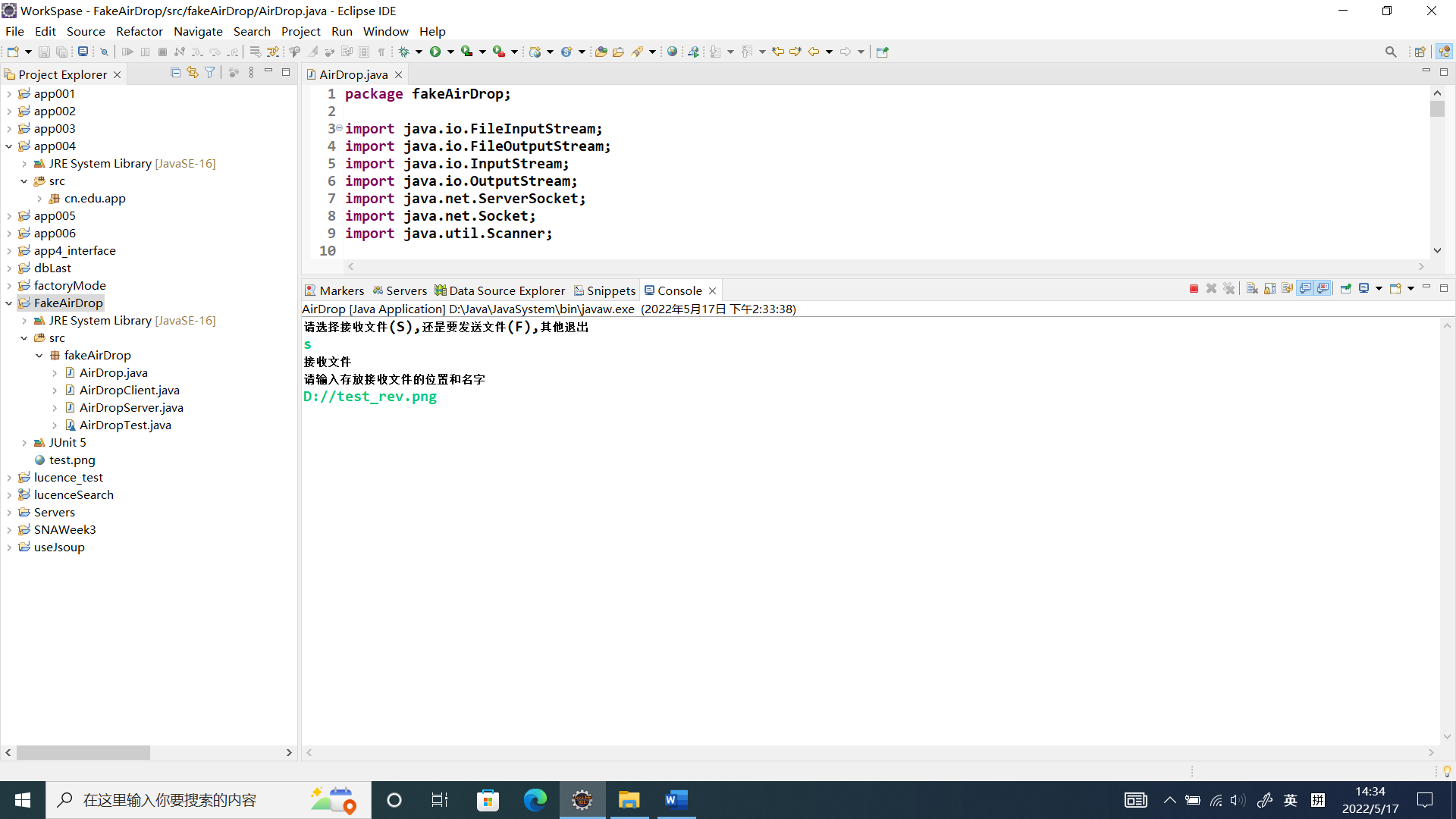


七、使用教程

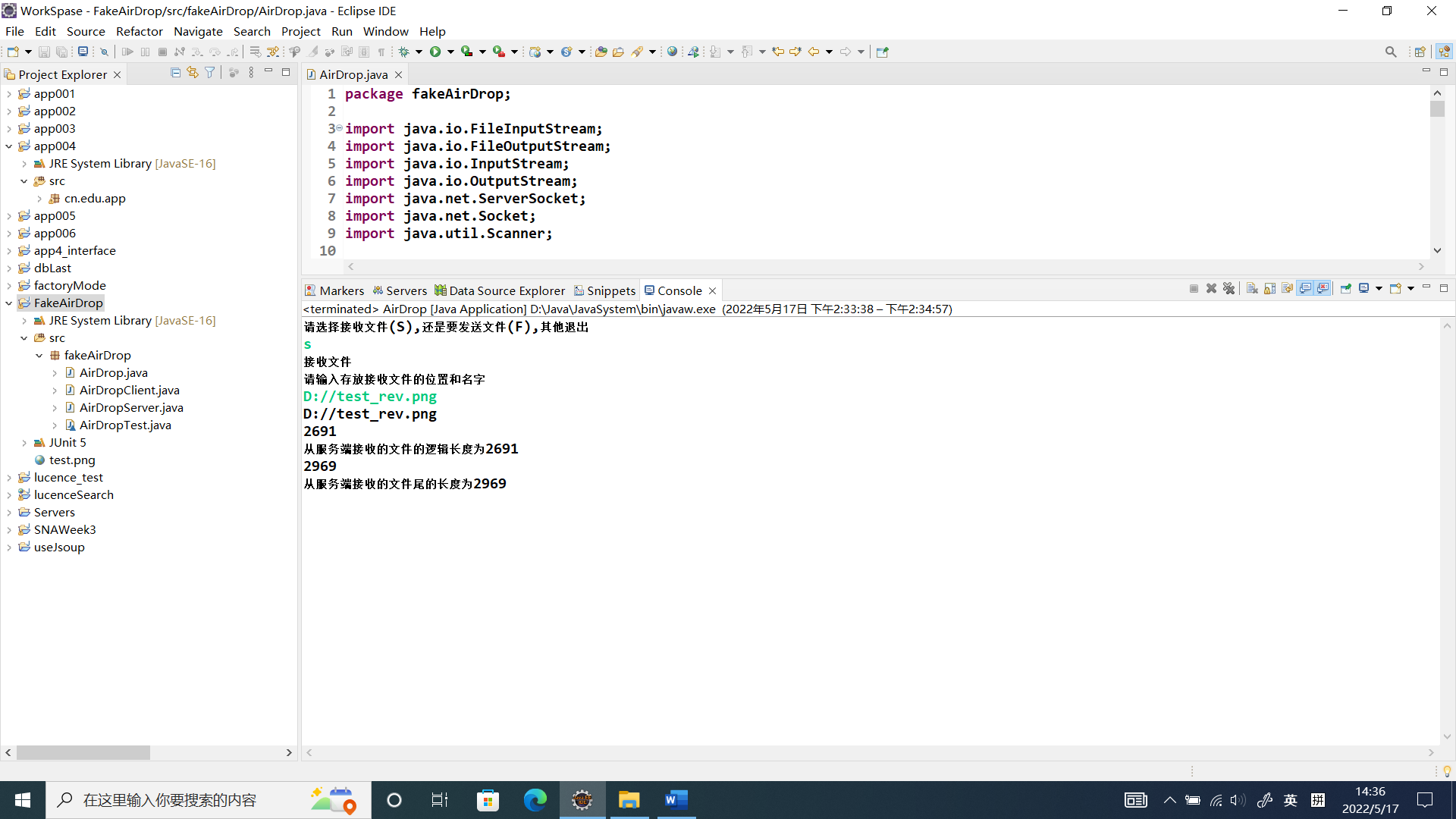
（1）开始时选择发送文件和待发送的文件，（发送文件为先）



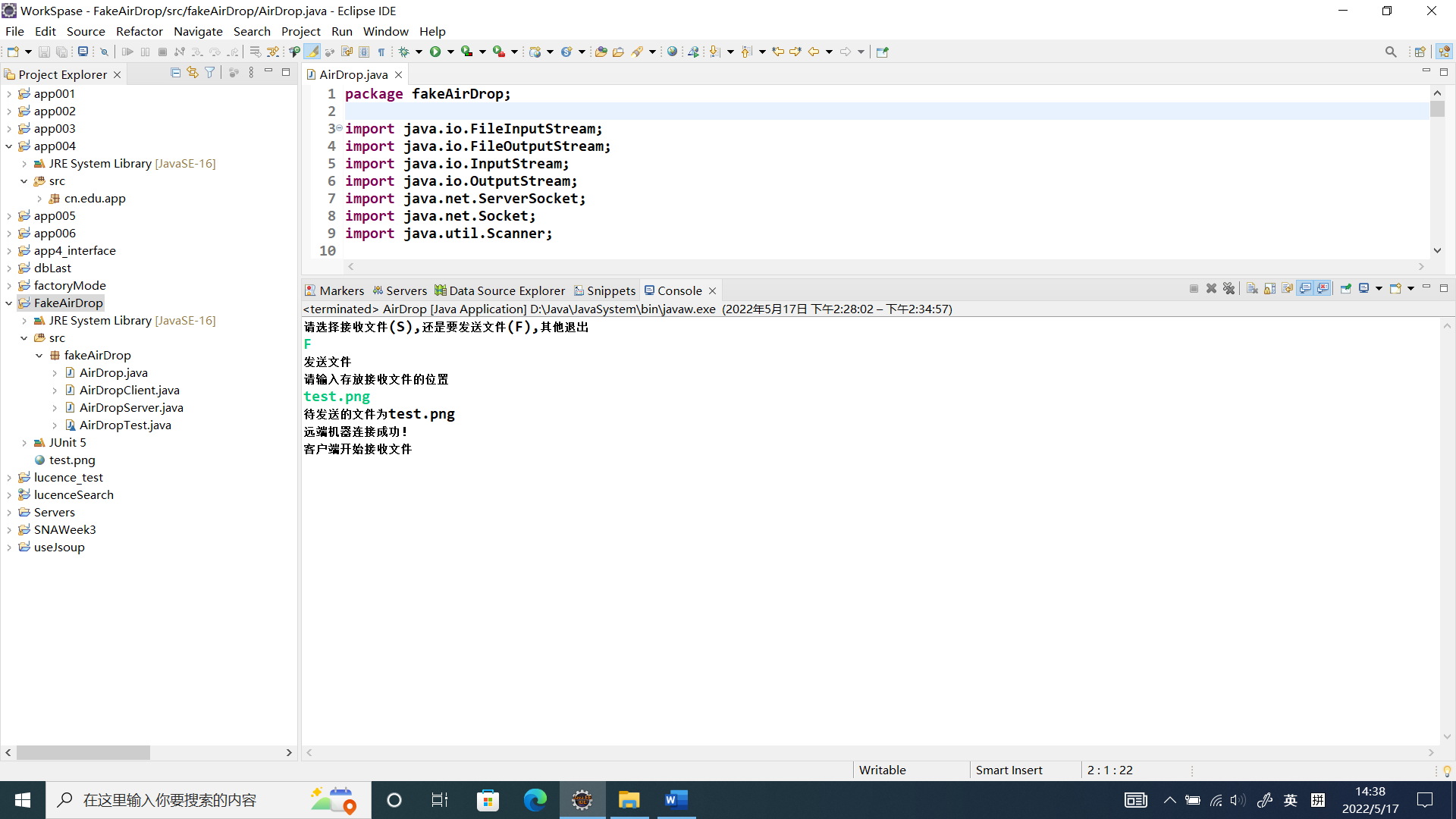
（2）再运行一个实例，选择接收文件和地址。（这些操作在airdrop中应该由程序决定）



（3）收文件方敲下回车键，开始接受文件

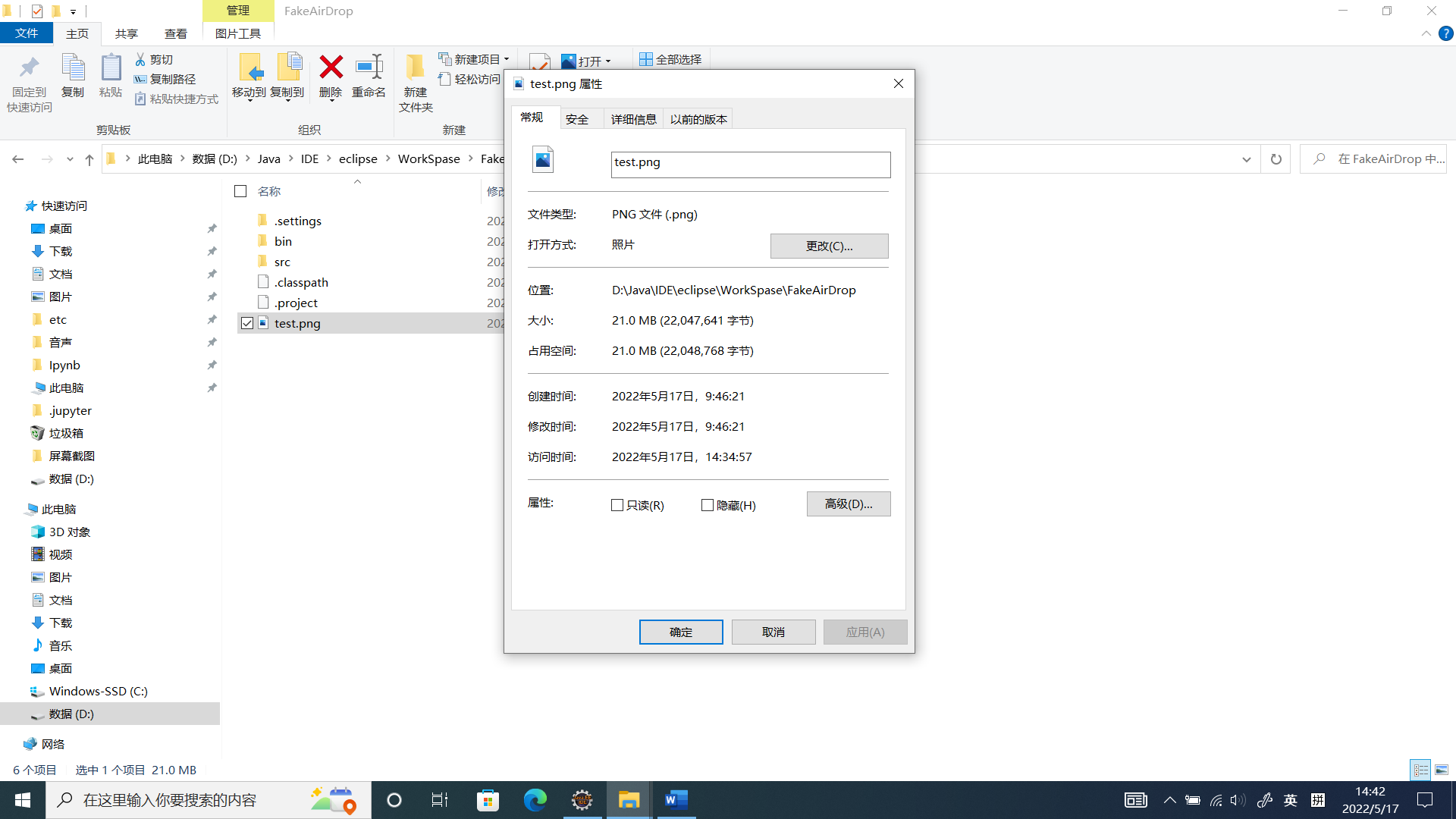


这里的逻辑长度是指文件有2691个缓冲区大小，文件尾长度指的是文件除2691个缓冲区大小后的零头大小（字节）。这时的传文件方。

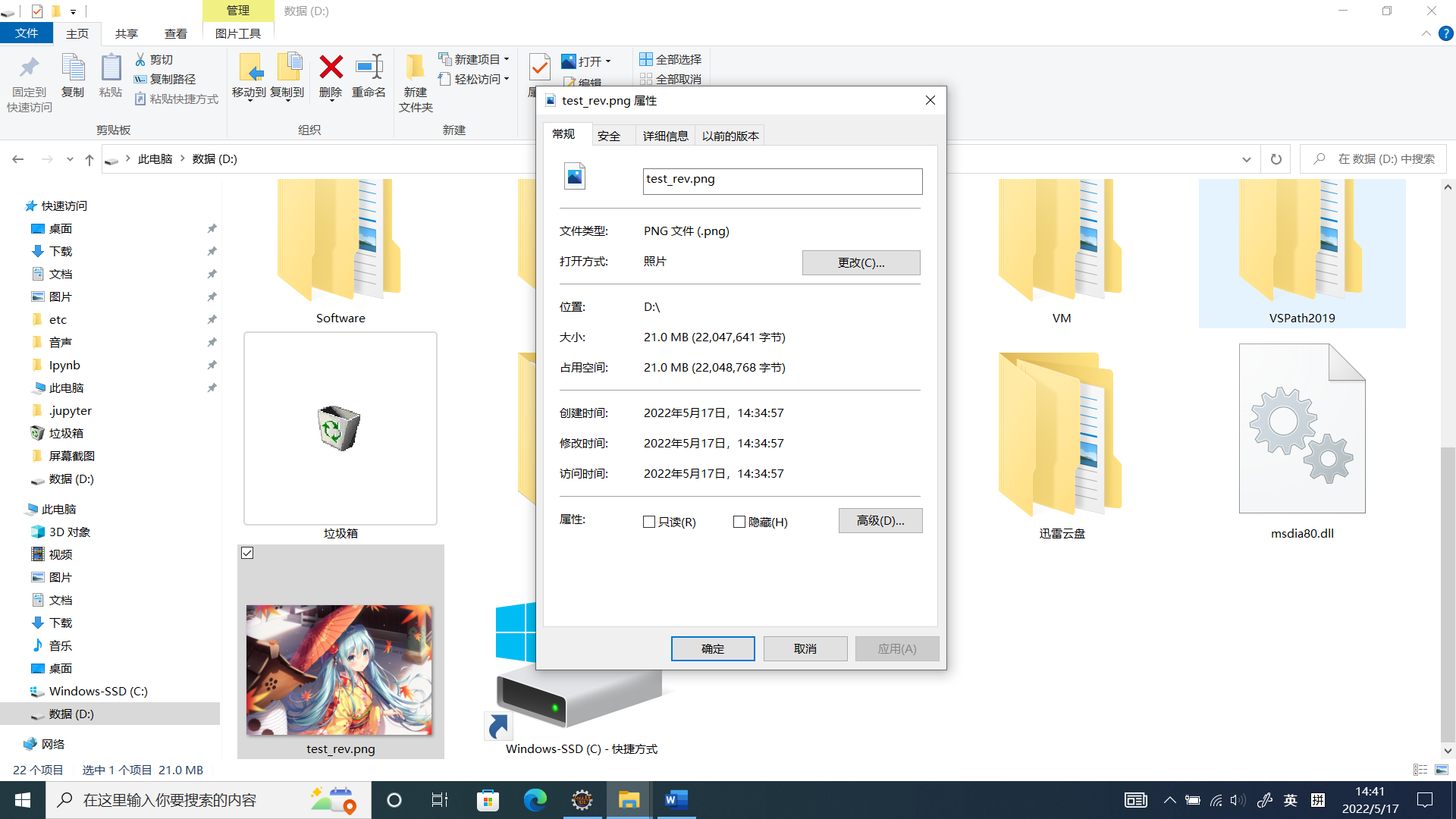


结果

待发文件的属性



接受到的文件的属性



待发文件的访问时间与接受到的文件的创建时间相同，而且大小和占用空间相同，实现了文件的复制。

八、代码附录

**package** fakeAirDrop;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.Scanner;

**public** **class** AirDrop {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请选择接收文件(S),还是要发送文件(F),其他退出");

String op = sc.nextLine();

**switch**(op) {

**case** "S":

**case** "s":

System.***out***.println("接收文件");

System.***out***.println("请输入存放接收文件的位置和名字");

String dir = sc.nextLine();

System.***out***.println(dir);

Socket socket = **new** Socket("127.0.0.1", 7888);

// 客户端的输出流

OutputStream os = socket.getOutputStream();

// 向服务器发送确认消息

os.write("客户端开始接收文件".getBytes());

// 从服务端接收文件

InputStream is = socket.getInputStream();

FileOutputStream fos = **new** FileOutputStream(dir);

**byte**[] buff = **new** **byte**[8192];// 字节数组

//获取文件逻辑长度

is.read(buff);

String str = **new** String(buff, 0, 4);// 字节数组转换为字符串

**int** len = Integer.*parseInt*(str);

System.***out***.println(str);

System.***out***.println("从服务端接收的文件的逻辑长度为"+len);

//接收文件的大部分数据

**for**(**int** i=0;i<len;i++) {

is.read(buff);

fos.write(buff);

}

//获取文件尾长度

is.read(buff);

str = **new** String(buff, 0, 4);

System.***out***.println(str);

**int** size = Integer.*parseInt*(str);

System.***out***.println("从服务端接收的文件尾的长度为"+size);

//接收文件尾

is.read(buff);

fos.write(buff,0,size);

//关闭资源

fos.close();

is.close();

os.close();

socket.close();

**break**;

**case** "F":

**case** "f":

System.***out***.println("发送文件");

System.***out***.println("请输入存放接收文件的位置");

String dir1 = sc.nextLine();

System.***out***.println("待发送的文件为"+dir1);

/\*----------\*/

//创建服务器ServerSocket对象(默认为127.0.0.1)

ServerSocket serverSocket = **new** ServerSocket(7888);

//监听连接

Socket socket1 = serverSocket.accept();

//直到监听到链接，才会继续向下执行

System.***out***.println("远端机器连接成功！");

/\*----------\*/

//获取服务端的输入流，从客户端接收信息

InputStream is1 = socket1.getInputStream();

//创建缓冲区

**byte**[] buff1 = **new** **byte**[8192];//字节数组

**int** length1 = 0;

length1 = is1.read(buff1);

String str1 = **new** String(buff1,0,length1);//字节数组转换为字符串

System.***out***.println(str1);//接收确认消息

/\*\*

\* 获取服务端的输出流，往客户端传输文件

\*/

OutputStream os1 = socket1.getOutputStream();//Socket字节流

FileInputStream fis = **new** FileInputStream(dir1);//文件流

**int** len1 = fis.available()/8192;

os1.write((len1+"").getBytes());

//发送文件大部分数据

**for**(**int** i=0;i<len1;i++) {

fis.read(buff1);

os1.write(buff1);

}

//发送文件的尾部

**int** size1 = fis.read(buff1);

os1.write((size1+"").getBytes());

os1.write(buff1,0,size1);

fis.close();

//关闭资源

is1.close();

os1.close();

socket1.close();

**break**;

**default**:

System.***out***.println("退出");

**break**;

}

sc.close();

}

}