## Uploadinfo 上传信息实体类：

分段上传**multiPartUpload()**中要用到，主要是把上传信息序列化保存到本地中

## Utils 工具类：

**getS3()** 获取S3对象

通过bucketName、accessKey、secretKey、serviceEndpoint构造ClientBuilder然后build出目标对象AmazonS3。

**getS3FilesLists()** 获取S3文件列表的函数

这个函数把s3里面的所有文件的路径列出来，然后以这些路径为key构造HashMap，value为S3ObjectSummary。

S3ObjectSummary是S3中每一个文件的基础信息，如文件大小，文件修改时间等。

**getLocalFilesLists()** 获取本地目录文件列表的函数

与上一个函数类似，把本地所有文件的路径列出来，用路径构造List。

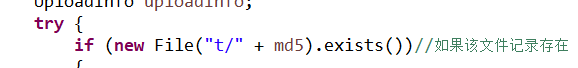
**upLoadFile()** 小文件上传函数，文件大小小于20M使用

**multiObjectDelete()** 删除S3文件函数

**downloadFile()** 下载函数，从s3拉取指定文件到本地

**multiPartUpload()** 大文件，分段上传

首先获取要上传的文件的md5，然后通过存储上传进度的目录判断上传记录文件是否存在。



t是临时文件夹，t文件夹储存的是上面的上传信息实体类的序列化保存文件，通过md5来保证文件名唯一。

所以如果记录存在，说明是传到一半，如果记录不存在，说明是第一次传；

1. 记录不存在的情况



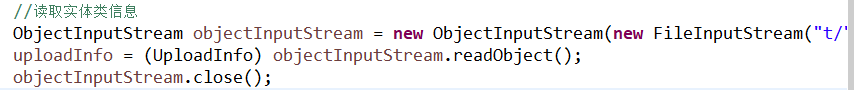
新建实体类，初始化上传请求，得到uploadid，把uploadid给实体类；

然后通过for循环上传分块，每一个小块传完后，就把uploadid，FileOffset，PartNumber这些信息写到实体类，然后实体类序列化成文件写到硬盘上。这样便可以实现断点续传，保留已存块。



1. 如果记录存在，就将实体类数据反序列化，直接读入实体类，通过实体类储存的信息，继续上传。

此后每一个分块上传完也重新写入到硬盘。



## Main类 主逻辑：

**path** path是本地同步的目录

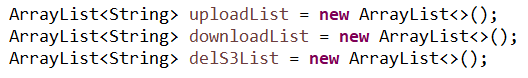
**main()** 分为首次同步和之后的同步，首次同步选择s3为基准，此后选择由输入数字决定



**sync()** 同步函数



clashMode决定冲突内容选择(就是同步的时候，s3和本地都有同路径同名文件，但是文件内容不同，所取的那个)



分别为每次同步要上传的文件，要下载的文件，要删除的文件。



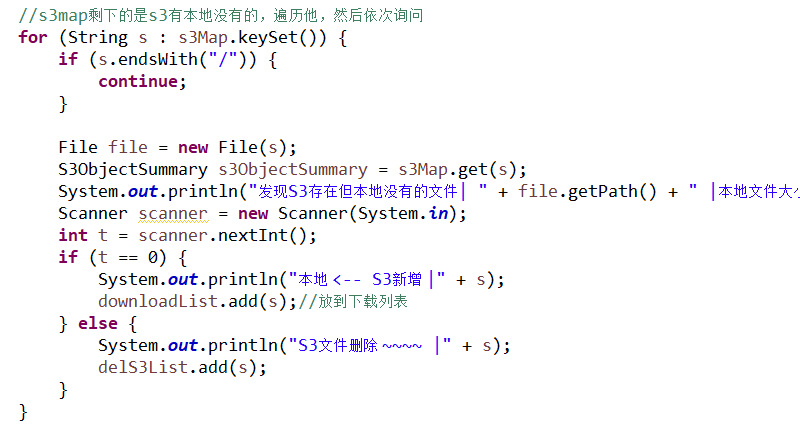
取两个列表，一个为s3的文件路径列表，一个为本地的文件路径列表



1. 循环本地路径list，如果是/结尾，说明是目录路径，不需要处理。
2. 如果s3map中有跟本地路径同名的key，说明这个文件本地s3都有，那么就分别取两地文件的md5比较。

最后把s3map中跟本地同路径那一项删掉。

1. 如果list有，s3map没有，说明这一项是本地有，s3没有的，就放到上传列表里面。



除此以外，s3列表剩余的都是s3有而本地没有的，

此时有两种情况，一种是本地删掉了这个文件，第二种是s3新增了这个文件。所以询问用户要选择哪种情况，第一种就放到下载列表，第二种就放到s3删除列表。



最后就是循环三个列表，分别调用工具类中的上传、下载、删除函数进行同步更新。