**《Java程序开发》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | **任齐轩** | | | **年级** | **2020级** |
| **学号** | | **20204154** | | | **专业、班级、** | **计卓2班** |
| **实验名称** | **基于命令行的文件管理器** | | | | | |
| **实验时间** | **2021.10.28** | | **实验地点** | **DS1408** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 □综合性** | | |
| 教师评价：  评价教师签名： | | | | | | |
| 一、实验目的  1） 学会文件夹的创建、删除与进入；  2）学会对文件夹下内容进行罗列；  3）学习文件与文件夹的拷贝；  4）学习对文件进行加密与解密。 | | | | | | |
| 二、实验项目内容  1）实现文件夹创建、删除、进入；  2）实现当前文件夹下的内容罗列；  3）实现文件拷贝和文件夹拷贝（文件夹拷贝指深度拷贝，包括所有子目录和文件）；  4）实现指定文件的加密和解密。 | | | | | | |
| 1. 实验过程或算法（源程序）   功能类：  package Project\_03; import java.io.\*; public class file\_manager {  private String Path = "";  private String Encoding = "UTF-8";   // class file manager creation  public file\_manager() {  File file = new File("");  Path = file.getAbsolutePath() + File.*separator*;  }   // output the current path  public final void cur\_pos() {  System.*out*.println("Current Path is :" + this.Path);  }   // the input file is not a file  private final void file\_denied(File file) {  System.*out*.println("Reason: " + file.getName() + " is not a file!");  }   // the input filedir is not a file dir  private final void filedir\_denied(File filedir) {  System.*out*.println("Reason: " + filedir.getName() + " is not a file dictionary!");  }   // the input path is can't be found  private final void path\_missing(File file) {  System.*out*.println("Reason: Can't find " + file.getAbsolutePath() + " !");  }   // to create a file in the input path  public boolean file\_creation(String file\_name) {  boolean create\_bool = false;  File file = new File(this.Path + file\_name);  if (file.exists()) {  System.*out*.println("Creation Failed!");  System.*out*.println("Reason: " + file\_name + " already exits!");  }  else {  try {  file.createNewFile();  create\_bool = true;  System.*out*.println("Creation Succeed!");  } catch (IOException e) {  e.printStackTrace();  }  }  return create\_bool;  }   // to create a file dir in the input path  public boolean filedir\_creation(String filedir\_name) {  boolean create\_bool = false;  File filedir = new File(this.Path + filedir\_name);  if (filedir.exists()) {  System.*out*.println("Creation Failed!");  System.*out*.println("Reason: " + filedir\_name + " already exits!");  }  else {  try {  filedir.mkdirs();  create\_bool = true;  System.*out*.println("Creation Succeed!");  } catch (Exception e) {  e.printStackTrace();  }  }  return create\_bool;  }   public boolean delete(String file\_name) {  boolean delete\_bool = false;  File file = new File(this.Path + file\_name);  if (file.exists()) {  if (file.isFile()) {  delete\_bool = delete\_file(file.getAbsolutePath());  if (delete\_bool) {  System.*out*.println("Delete file" + "\"" + file\_name + "\"" + "succeeded!");  }  }  else {  delete\_bool = delete\_filedir(file.getAbsolutePath());  if (delete\_bool) {  System.*out*.println("Delete dir" + "\"" + file\_name + "\"" + "succeeded!");  }  }  }  else {  System.*out*.println("Delete failed!");  path\_missing(file);  return delete\_bool;  }  return delete\_bool;  }   // delete the file in the input path  public boolean delete\_file(String file\_name) {  boolean delete\_bool = false;  File file = new File(file\_name);  if (file.delete()) delete\_bool = true;  return delete\_bool;  }   // delete the file dir in the input path  public boolean delete\_filedir(String filedir\_name) {  boolean delete\_bool = false;  if (!filedir\_name.endsWith(File.*separator*)) {  filedir\_name = filedir\_name + File.*separator*;  }   File dirFile = new File(filedir\_name);  File[] fileList = dirFile.listFiles();  for (int i = 0; i < fileList.length; i++) {  if (fileList[i].isFile()) {  delete\_bool = delete\_file(fileList[i].getAbsolutePath());  if (!delete\_bool) break;  }  else {  delete\_bool = delete\_filedir(fileList[i].getAbsolutePath());  if (!delete\_bool) break;  }  }  if (dirFile.delete()) delete\_bool = true;  return delete\_bool;  }   // change the current path  public boolean change\_dir(String filedir\_name) {  boolean change\_bool = false;  File filedir = new File(filedir\_name);   boolean absPath\_bool = false; // check abspath  if (filedir.isAbsolute()) {  absPath\_bool = true;  }  else {  filedir = new File(this.Path + filedir\_name);  }   if (filedir.exists()) { // file dictionary exists  if (filedir.isDirectory()) {// input filedir is dictionary  if (!filedir\_name.endsWith(File.*separator*)) {  filedir\_name = filedir\_name + File.*separator*;  }   if (absPath\_bool) {  this.Path = filedir\_name;  change\_bool = true;  }  else {  this.Path += filedir\_name;  change\_bool = true;  }  }  else { // input filedir is not dictionary  System.*out*.println("Failed to change dictionary!");  filedir\_denied(filedir);  }  }  else { // file dictionary does not exists  System.*out*.println("Failed to change dictionary!"); // report error  file\_denied(filedir);  }   if (absPath\_bool) {  System.*out*.println("Change succeeded!"); // change succeeded  }   // check if the path is  return change\_bool;  }   // list all the files' names in the current path  public String[] file\_list() {  File filedir = new File(this.Path);  String[] file\_lst = {};   if (!filedir.exists()) { // filedir does not exists  System.*out*.println("List failed!");  file\_denied(filedir);  return file\_lst;  }  if (filedir.isFile()) {  System.*out*.println("List failed!");  filedir\_denied(filedir);  return file\_lst;  }  // current filedir is not dictionary  System.*out*.println("Current File dictionary includes: ");  File[] files = filedir.listFiles();  for (File file:files) {  if (file.isFile()) {  System.*out*.println("[File]" + " " + file.getName());  }  else {  System.*out*.println("[Dictionary]" + " " + file.getName());  }  }  file\_lst = filedir.list();  return file\_lst;  }  public boolean copy(String file\_name) {  boolean copy\_bool = false;  File file = new File(this.Path + file\_name);   if (!file.exists()) {  System.*out*.println("Copy failed!");  path\_missing(file);  return copy\_bool;  }   String currPath = this.Path + file\_name;  String destPath = this.Path;    if (file.isFile()) copy\_bool = file\_copy(currPath, destPath);  else copy\_bool = filedir\_copy(currPath, destPath);   if (copy\_bool) {  System.*out*.println("Copy succeeded！");  }  else {  System.*out*.println("Copy failed！");  }  return copy\_bool;  }   // copy the file from former path to target path  public boolean file\_copy(String former\_path, String target\_path) {  boolean copy\_bool = false;  File file = new File(former\_path);  File copyFile = new File(target\_path + file.getName());   String fileFullName = file.getName();  String filePriName = fileFullName.substring(0, fileFullName.lastIndexOf("."));  String fileExtenName = fileFullName.substring(fileFullName.lastIndexOf("."));   if (copyFile.exists()) {  copyFile = new File(target\_path + filePriName + "-copy" + fileExtenName);  }   FileInputStream fIn = null;  BufferedInputStream bIn = null;  FileOutputStream fOut = null;  BufferedOutputStream bOut = null;   try {  fIn = new FileInputStream(file);  bIn = new BufferedInputStream(fIn);  fOut = new FileOutputStream(copyFile);  bOut = new BufferedOutputStream(fOut);   byte[] Bytes = new byte[1024];  int length = 0;  while((length = bIn.read(Bytes)) != -1) {  bOut.write(Bytes, 0, length);  }  bOut.flush();   copy\_bool = true;  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }   try {  bOut.close();  fOut.close();  bIn.close();  fIn.close();  } catch (IOException e) {  e.printStackTrace();  }  return copy\_bool;  }   // copy the file dictionary from former path to target path  public boolean filedir\_copy(String former\_path, String target\_path) {  boolean copy\_bool = false;  File dirFile = new File(former\_path);  File destDirFile = new File(target\_path + dirFile.getName());   if (destDirFile.exists()) {  destDirFile = new File(target\_path + dirFile.getName() + "-副本");  }   destDirFile.mkdirs();   File temFile = null;  String[] fileNameList = dirFile.list();  for (String fileName:fileNameList) {  temFile = new File(former\_path + File.*separator* + fileName);   if (temFile.isFile()) {  if (!file\_copy(former\_path + File.*separator* + fileName, destDirFile.getAbsolutePath() + File.*separator*)) {  return copy\_bool;  }  }  else {  if (!filedir\_copy(former\_path + File.*separator* + fileName, destDirFile.getAbsolutePath() + File.*separator*)) {  return copy\_bool;  }  } // end else   } // end while   copy\_bool = true;  return copy\_bool;  }   // encode the target file  public boolean file\_encode(String file\_path) {  boolean encode\_bool = false;   File file = new File(this.Path + file\_path);  if (!file.exists()) {  System.*out*.println("Decode Failed!");  path\_missing(file);  return encode\_bool;  }  if (!file.isFile()) {  System.*out*.println("Decode Failed!");  file\_denied(file);  return encode\_bool;  }   final byte key = 101; // code   // create input/output stream target  FileInputStream fileinput = null;  BufferedReader bufferreader = null;  FileOutputStream fileout = null;  BufferedWriter bufferwrite = null;   try {  fileinput = new FileInputStream(file);  bufferreader = new BufferedReader(new InputStreamReader(fileinput, this.Encoding));  fileout = new FileOutputStream(this.Path + "Encryp-" + file\_path);  bufferwrite = new BufferedWriter(new OutputStreamWriter(fileout, this.Encoding));   System.*out*.println("Encoding...");  int c = 0;  while ((c = bufferreader.read()) != -1) {  bufferwrite.write(c ^key);  }   bufferwrite.flush();  encode\_bool = true;  System.*out*.println("Encode Succeeded!");   } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }   // close input & output stream  try {  bufferreader.close();  bufferwrite.close();  fileinput.close();  fileout.close();  } catch (IOException e) {  e.printStackTrace();  }  return encode\_bool;  }     // decode the target file  public boolean file\_decode(String file\_path) {  boolean decode\_bool = false;  File encode\_file = new File(this.Path + file\_path);   if (!encode\_file.exists()) {  System.*out*.println("Decode failed!");  path\_missing(encode\_file);  return decode\_bool;  }  if (!encode\_file.isFile()) {  System.*out*.println("Decode failed!");  path\_missing(encode\_file);  return decode\_bool;  }   final byte key = 101;  // create input/output stream target  FileInputStream fileinput = null;  BufferedReader bufferreader = null;  FileOutputStream fileout = null;  BufferedWriter bufferwrite = null;   try {  fileinput = new FileInputStream(encode\_file);  bufferreader = new BufferedReader(new InputStreamReader(fileinput, this.Encoding));  fileout = new FileOutputStream(this.Path + "Decryp-" + file\_path);  bufferwrite = new BufferedWriter(new OutputStreamWriter(fileout, this.Encoding));   System.*out*.println("Decoding...");  int c = 0;  while ((c = bufferreader.read()) != -1) {  bufferwrite.write(c ^key);  }   bufferwrite.flush();  decode\_bool = true;  System.*out*.println("Decode Succeeded!");   } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }   // close input & output stream  try {  bufferreader.close();  bufferwrite.close();  fileinput.close();  fileout.close();  } catch (IOException e) {  e.printStackTrace();  }  return decode\_bool;  }   }  实现类：  package Project\_03;  import java.util.Scanner;  public class FM\_main {  public static final void menue() {  System.*out*.println("----------------Menue:----------------");  System.*out*.println("1.List files");  System.*out*.println("2.Enter filedir");  System.*out*.println("3.Create file");  System.*out*.println("4.Create filedir");  System.*out*.println("5.Delete file or filedir");  System.*out*.println("6.Copy file or filedir");  System.*out*.println("7.Encode file");  System.*out*.println("8.Decode file");  System.*out*.println("9.menue");  System.*out*.println("0.exit()");  }   public static void main(String[] args) {  file\_manager fm = new file\_manager();  Scanner sc =new Scanner(System.*in*);   System.*out*.println("---------The CMD File Manager---------");   fm.cur\_pos();  *menue*();  boolean flag = true;  while(flag) {  System.*out*.println("");  System.*out*.print("Please enter the num you want to choose:");  int iFuncIndex = 0;  // input check  while (true) {  if (sc.hasNextInt()) {  iFuncIndex = sc.nextInt();  if ((0 <= iFuncIndex) && (iFuncIndex <= 11)) {  break;  }  else {  System.*out*.println("Input Error!");  System.*out*.println("Reason：input num is supposed in 0-9");  System.*out*.println("");  System.*out*.print("Please enter the num you want to choose");  }  }  else {  System.*out*.println("Input Error!");  System.*out*.println("Reason: input tpye incorrect!");  sc.next();  System.*out*.println("");  System.*out*.print("Please enter the num you want to choose");  }  }   if (iFuncIndex != 0) { fm.cur\_pos(); }   // menue selection  switch (iFuncIndex) {  case 0: {  flag = false;  System.*out*.println("Exit succeeded! Thx for using");  break;  }  case 1:  {  fm.file\_list();  break;  }  case 2:  {  System.*out*.print("Input the dictionary you want to enter:");  String sPath = sc.next();  fm.change\_dir(sPath);  break;  }  case 3:  {  System.*out*.print("Input the File name you want to create:");  String creatFileName = sc.next();  fm.file\_creation(creatFileName);  break;  }  case 4:  {  System.*out*.print("Input the File dictionary name you want to create:");  String creatDirectoryName = sc.next();  fm.filedir\_creation(creatDirectoryName);  break;  }  case 5:  {  System.*out*.print("Input the name of File or File dictionary you want to delete:");  String deleteFileName = sc.next();  fm.delete(deleteFileName);  break;  }  case 6:  {  System.*out*.print("Input the name of File or File dictionary you want to copy:");  String fileName = sc.next();  fm.copy(fileName);  break;  }  case 7:  {  System.*out*.print("Input the name of File you want to encode:");  String encrypFileName = sc.next();  fm.file\_encode(encrypFileName);  break;  }  case 8:  {  System.*out*.print("Input the name of File you want to decode:");  String decrypFileName = sc.next();  fm.file\_decode(decrypFileName);  break;  }  case 9: {  *menue*();  break;  }  }  }  sc.close();  } } | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  1. 启动初始状态  截屏2021-11-01 上午11.05.09   1. 创建文件截屏2021-11-01 上午11.07.21 2. 创建文件夹   截屏2021-11-01 上午11.07.39   1. 删除文件   截屏2021-11-01 上午11.08.00   1. 删除文件夹 2. 进入文件夹   截屏2021-11-01 上午11.06.42   1. 罗列文件夹所有文件   截屏2021-11-01 上午11.05.19   1. 拷贝文件   截屏2021-11-01 上午11.22.09   1. 拷贝文件夹（深拷贝）   截屏2021-11-01 上午11.08.27  截屏2021-11-01 上午11.22.28   1. 加密文件或文件夹   截屏2021-11-01 上午11.09.30   1. 解密文件或文件夹   截屏2021-11-01 上午11.09.44   1. 退出文件管理器截屏2021-11-01 上午11.09.57 2. 功能罗列   截屏2021-11-01 上午11.09.50 | | | | | | |

备注：

1、教师在布置需撰写实验报告的实验前，应先将报告书上的“实验题目”、“实验性质”、“实验目的”、“实验项目内容”等项目填写完成，然后再下发给学生。

2、教师在布置需撰写报告的实验项目时，应告知学生提交实验报告的最后期限。

3、学生应按照要求正确地撰写实验报告：

* 1. 在实验报告上正确地填写“实验时间”、“实验地点”等栏目。
  2. 将实验所涉及的源程序文件内容（实验操作步骤或者算法）填写在“实验过程或算法（源程序）”栏目中。
  3. 将实验所涉及源程序调试过程（输入数据和输出结果）或者实验的分析内容填写在“实验结果及分析和（或）源程序调试过程”栏目中。
  4. 在实验报告页脚的“报告创建时间：”处插入完成实验报告时的日期和时间。
  5. 学生将每个实验完成后，按实验要求的文件名通过网络提交（上载）到指定的服务器所规定的共享文件夹中。每个实验一个电子文档，如果实验中有多个电子文档（如源程序或图形等），则用WinRAR压缩成一个压缩包文档提交，压缩包文件名同实验报告文件名（见下条）。
  6. 提交的实验报告电子文档命名为：“年级（两位数字不要“级”字）专业（缩写：计算机科学与技术专业（计科）、网络工程专业（网络）、信息安全专业（信息）、物联网工程（物联网））班级（两位数字）学号（八位数字）姓名实验序号（一位数字）．doc。如学号为20115676、年级为2011级、专业为“计算机科学与技术”专业、班级为“02班”、姓名为“王宇”的学生，完成的第一次实验命名为： 11计科02班20115676王宇1．Doc，以后几次实验的报告名称以此类推。

4、教师（或助教）在评价学生实验时，应根据其提交的其他实验相关资料（例如源程序文件等）对实验报告进行仔细评价。评价后应完成的项目有:

1. 在“成绩”栏中填写实验成绩。每次实验成绩按照十分制方式评分，取整数分值。
2. 在“教师评价”栏中“评价教师签名”填写评价教师（或助教）姓名。
3. 课程实验环节结束后，任课教师将自己教学班的实验报告文件夹进行清理。在提交文件夹中，文件总数为实验次数×教学班学生人数（如，教学班人数为90人，实验项目为5，其文件数为：90×5=450）。任课教师一定要认真清理，总数相符，否则学生该实验项目不能得分。最后将学生提交的实验报告刻光盘连同实验成绩一起放入试卷袋存档。