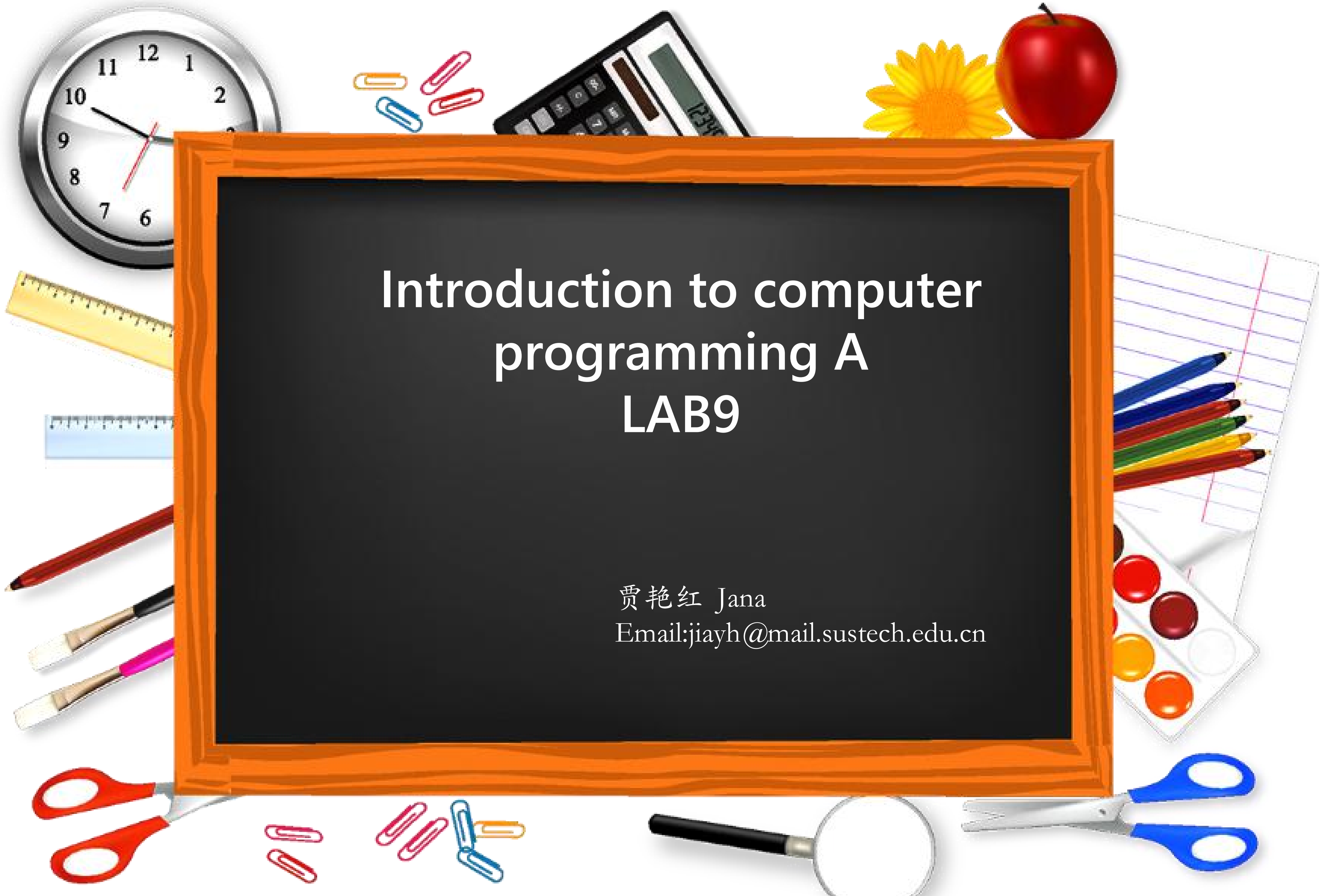


Introduction to computer programming A
LAB9

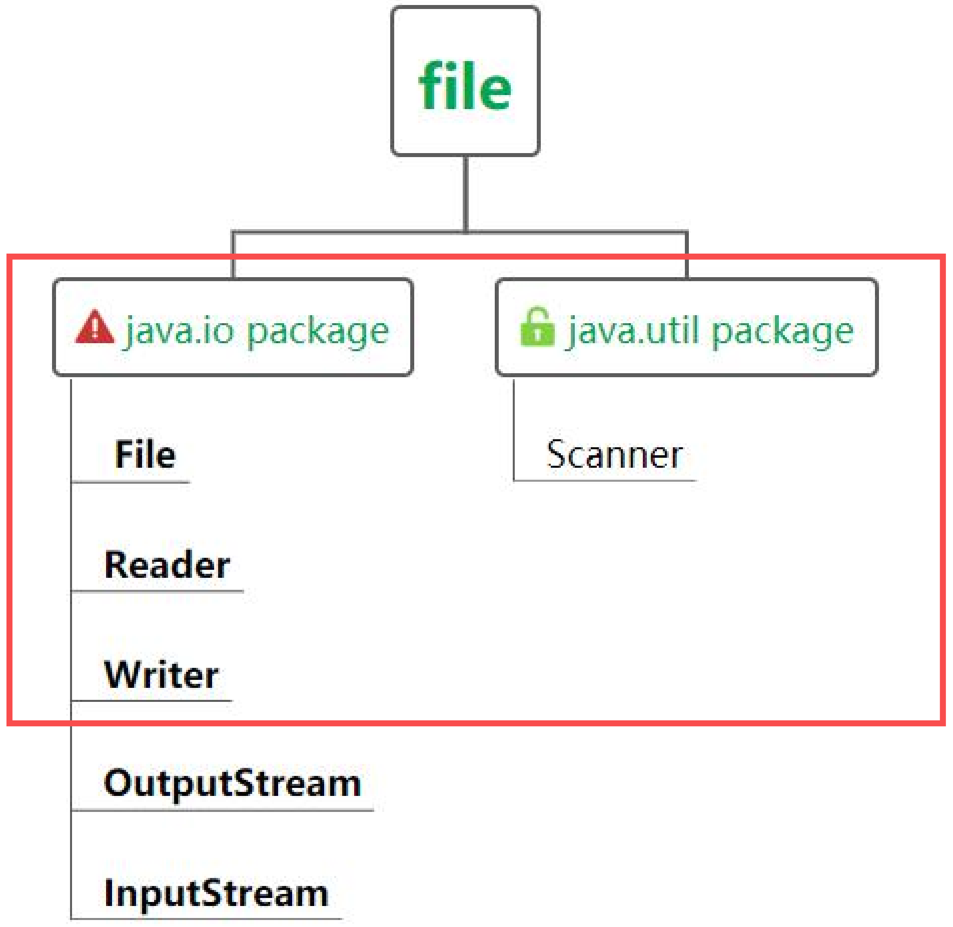
贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn

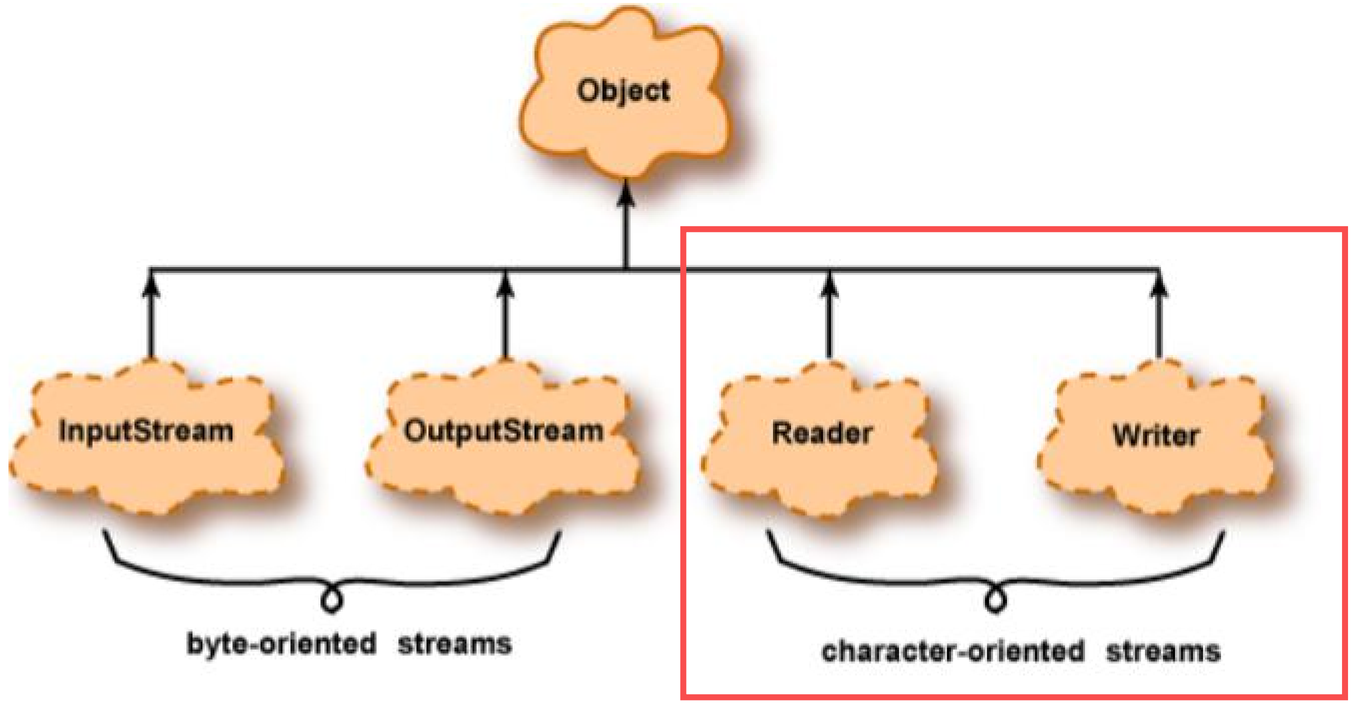
LAB OBJECTIVES

- **Learn how to read / write text files in JAVA**
- **Learn to create packages to organize classes**

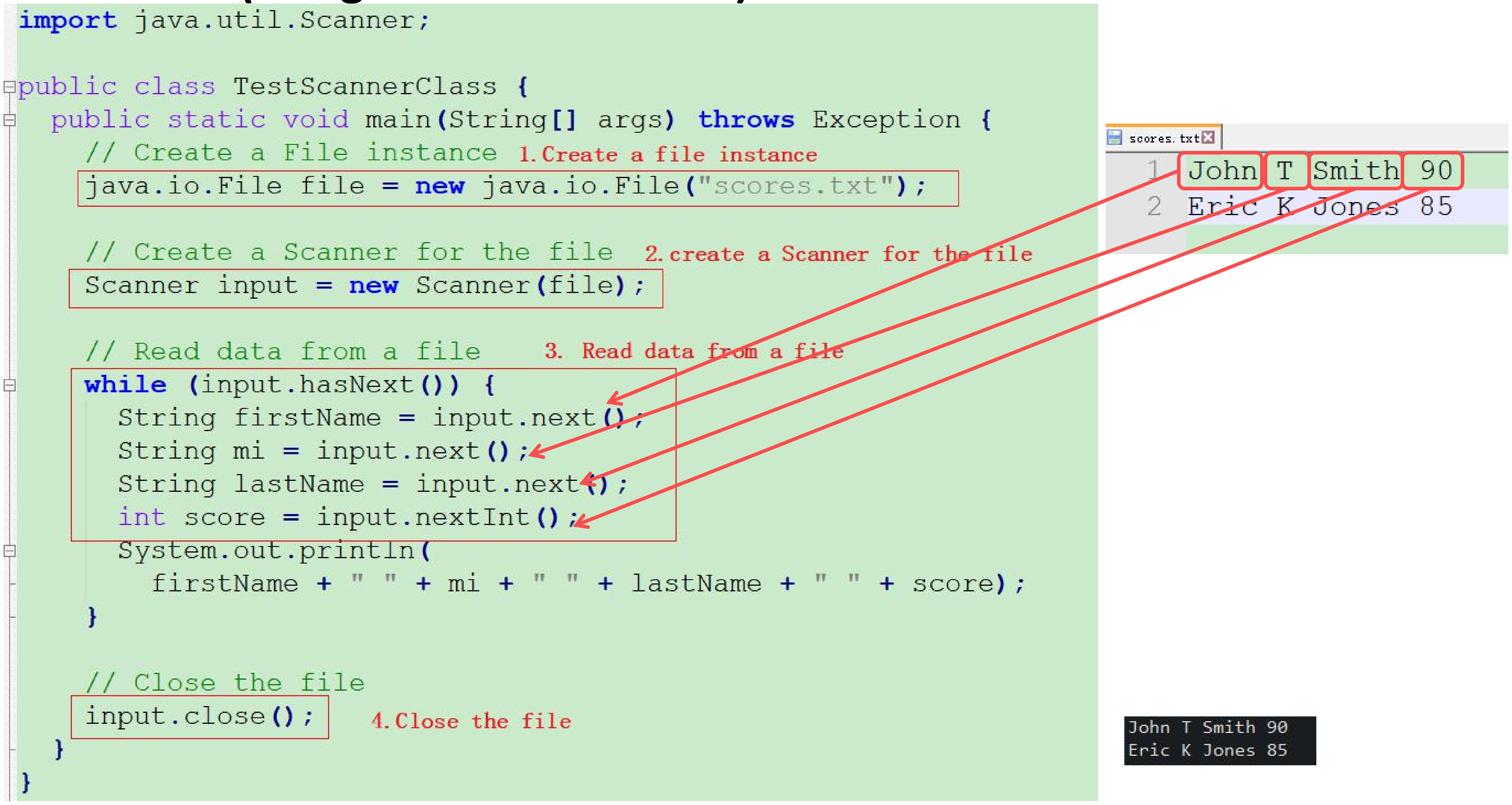
knowledge points

File Input and Output





Read a file(using the Scanner class)



```java
import java.util.Scanner;

public class TestScannerClass {
  public static void main(String[] args) throws Exception {
    // Create a File instance
    java.io.File file = new java.io.File("scores.txt");

    // Create a Scanner for the file
    Scanner input = new Scanner(file);

    // Read data from a file
    while (input.hasNext()) {
      String firstName = input.next();
      String mi = input.next();
      String lastName = input.next();
      int score = input.nextInt();
      System.out.println(
        firstName + " " + mi + " " + lastName + " " + score);
    }

    // Close the file
    input.close();
  }
}
```

1.Create a file instance

2.create a Scanner for the file

3. Read data from a file

4.Close the file

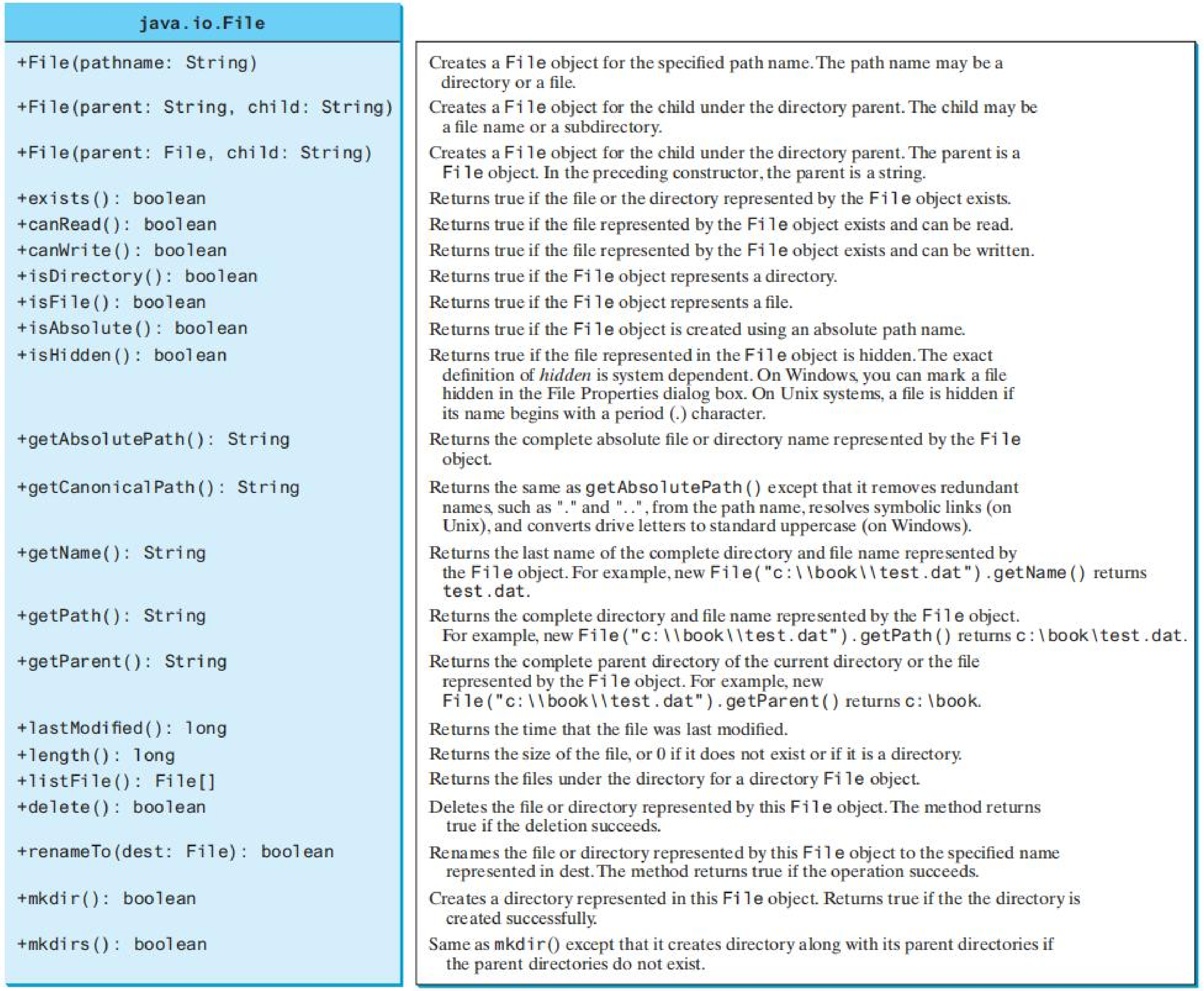
scores.txt

```
John T Smith 90
Eric K Jones 85
```

```
John T Smith 90
Eric K Jones 85
```

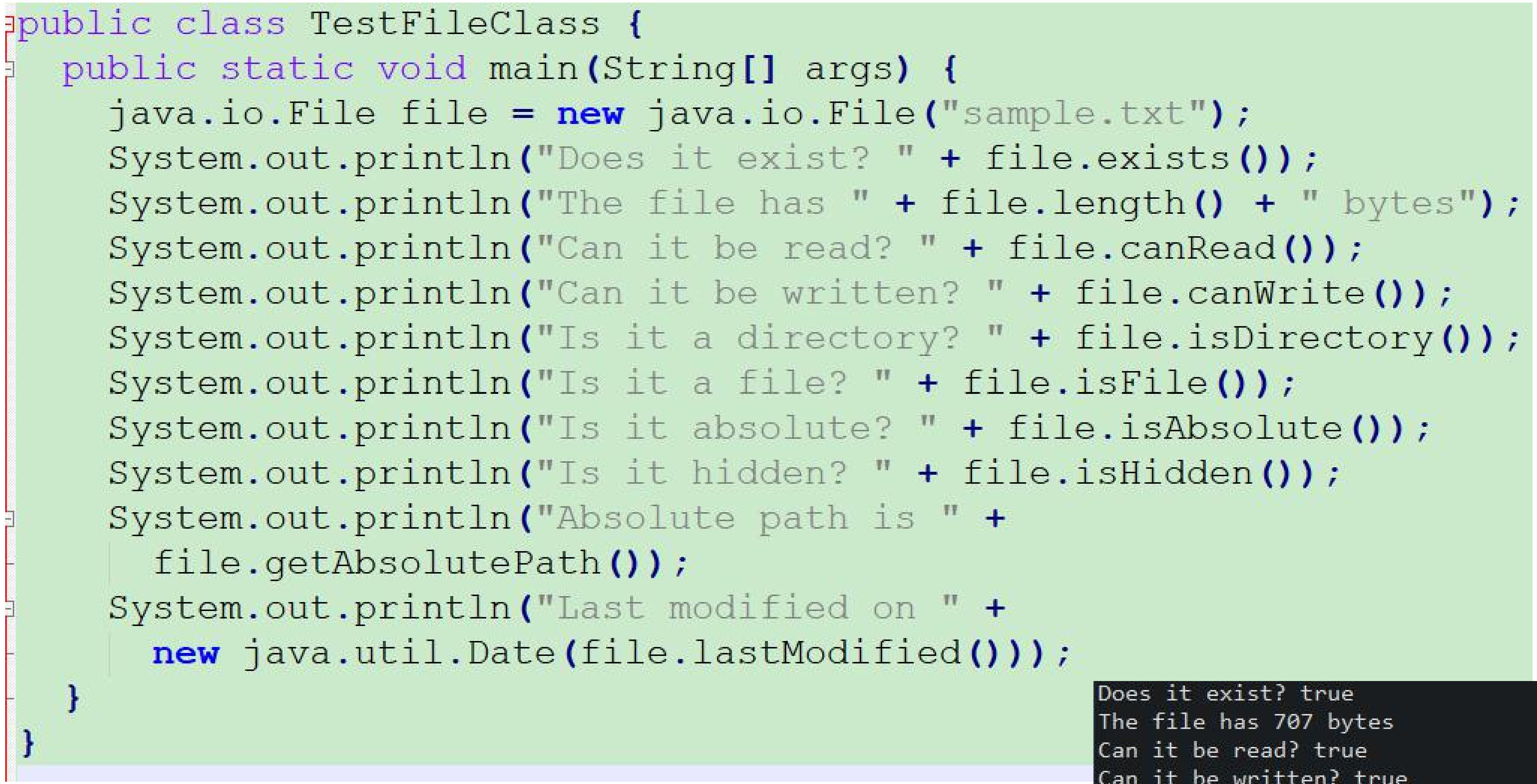
The File Class

The File class can be used to obtain file and directory properties, to delete and rename files and directories, and to create directories

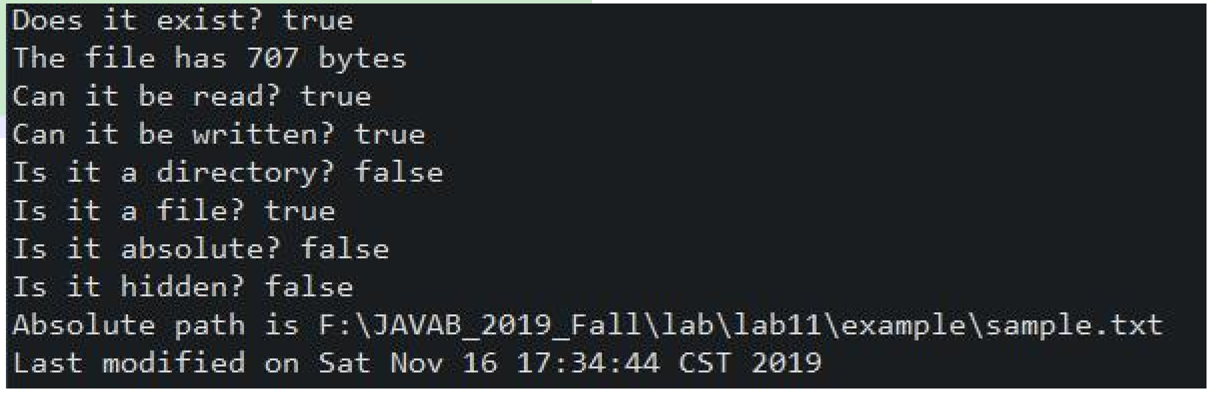


java.io.File	
`+File(pathname: String)`	Creates a `File` object for the specified path name. The path name may be a directory or a file.
`+File(parent: String, child: String)`	Creates a `File` object for the child under the directory parent. The child may be a file name or a subdirectory.
`+File(parent: File, child: String)`	Creates a `File` object for the child under the directory parent. The parent is a `File` object. In the preceding constructor, the parent is a string.
`+exists(): boolean`	Returns true if the file or the directory represented by the `File` object exists.
`+canRead(): boolean`	Returns true if the file represented by the `File` object exists and can be read.
`+canWrite(): boolean`	Returns true if the file represented by the `File` object exists and can be written.
`+isDirectory(): boolean`	Returns true if the `File` object represents a directory.
`+isFile(): boolean`	Returns true if the `File` object represents a file.
`+isAbsolute(): boolean`	Returns true if the `File` object is created using an absolute path name.
`+isHidden(): boolean`	Returns true if the file represented in the `File` object is hidden. The exact definition of *hidden* is system dependent. On Windows, you can mark a file hidden in the File Properties dialog box. On Unix systems, a file is hidden if its name begins with a period (.) character.
`+getAbsolutePath(): String`	Returns the complete absolute file or directory name represented by the `File` object.
`+getCanonicalPath(): String`	Returns the same as `getAbsolutePath()` except that it removes redundant names, such as `"."` and `".."`, from the path name, resolves symbolic links (on Unix), and converts drive letters to standard uppercase (on Windows).
`+getName(): String`	Returns the last name of the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getName()` returns `test.dat`.
`+getPath(): String`	Returns the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getPath()` returns `c:\book\test.dat`.
`+getParent(): String`	Returns the complete parent directory of the current directory or the file represented by the `File` object. For example, new `File("c:\\book\\test.dat").getParent()` returns `c:\book`.
`+lastModified(): long`	Returns the time that the file was last modified.
`+length(): long`	Returns the size of the file, or 0 if it does not exist or if it is a directory.
`+listFile(): File[]`	Returns the files under the directory for a directory `File` object.
`+delete(): boolean`	Deletes the file or directory represented by this `File` object. The method returns true if the deletion succeeds.
`+renameTo(dest: File): boolean`	Renames the file or directory represented by this `File` object to the specified name represented in dest. The method returns true if the operation succeeds.
`+mkdir(): boolean`	Creates a directory represented in this `File` object. Returns true if the the directory is created successfully.
`+mkdirs(): boolean`	Same as `mkdir()` except that it creates directory along with its parent directories if the parent directories do not exist.

The File Class(case study)

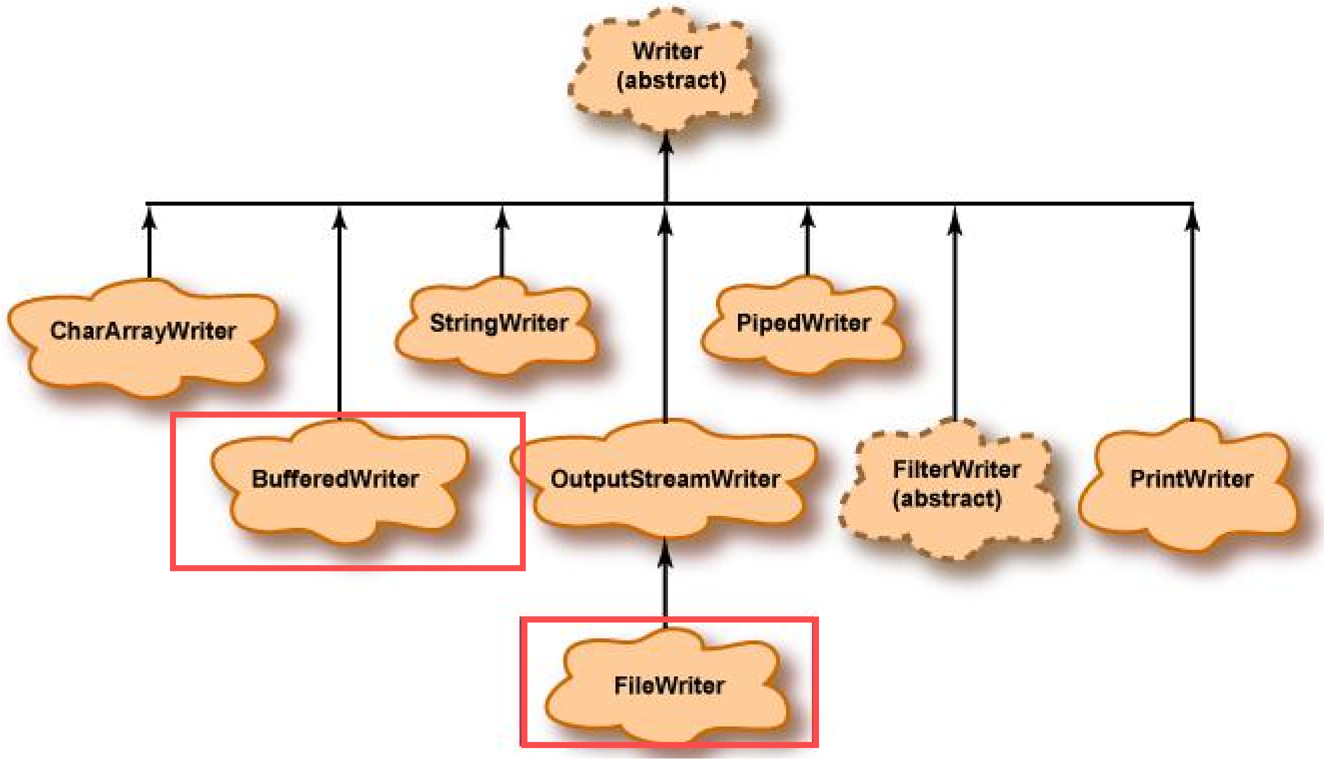


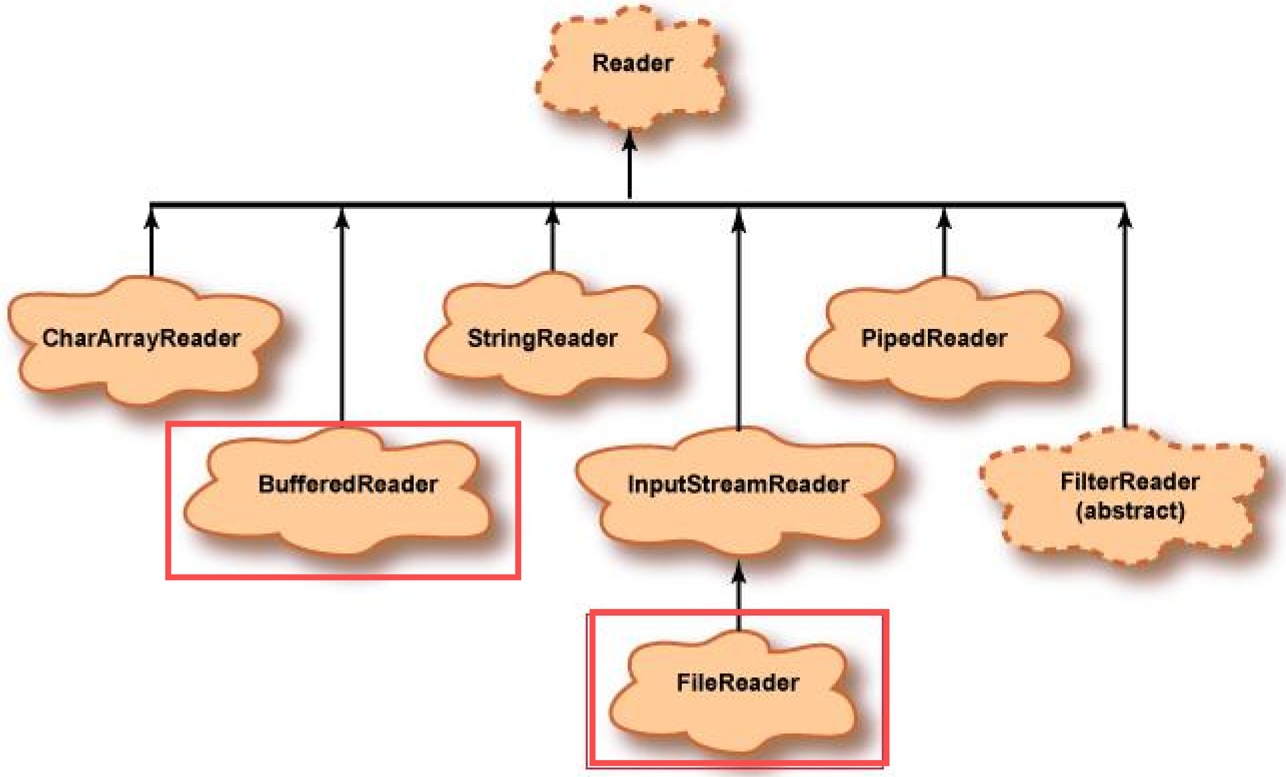
```java
public class TestFileClass {
  public static void main(String[] args) {
    java.io.File file = new java.io.File("sample.txt");
    System.out.println("Does it exist? " + file.exists());
    System.out.println("The file has " + file.length() + " bytes");
    System.out.println("Can it be read? " + file.canRead());
    System.out.println("Can it be written? " + file.canWrite());
    System.out.println("Is it a directory? " + file.isDirectory());
    System.out.println("Is it a file? " + file.isFile());
    System.out.println("Is it absolute? " + file.isAbsolute());
    System.out.println("Is it hidden? " + file.isHidden());
    System.out.println("Absolute path is " +
      file.getAbsolutePath());
    System.out.println("Last modified on " +
      new java.util.Date(file.lastModified()));
  }
}
```



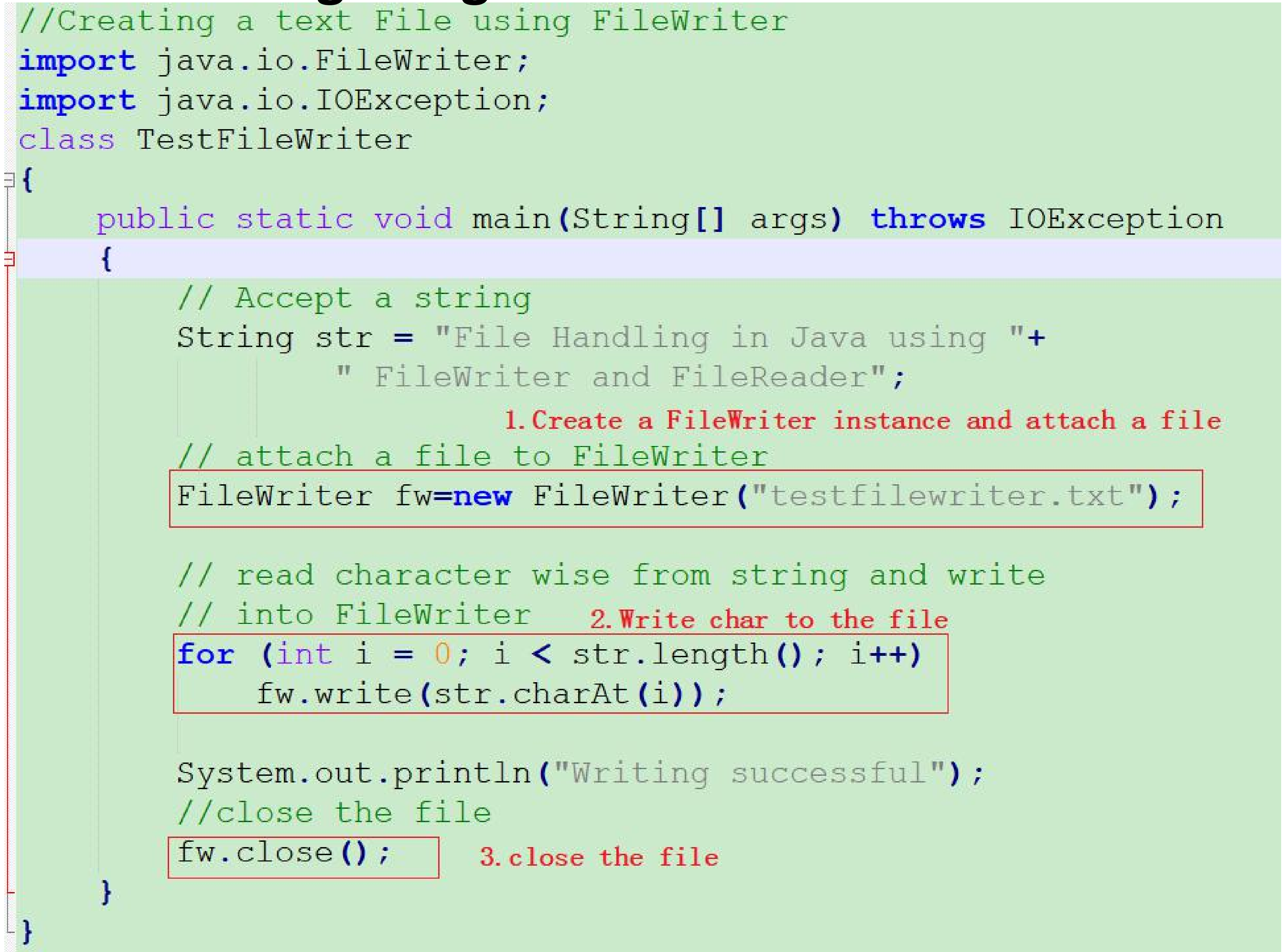
```
Does it exist? true
The file has 707 bytes
Can it be read? true
Can it be written? true
Is it a directory? false
Is it a file? true
Is it absolute? false
Is it hidden? false
Absolute path is F:\JAVAB_2019_Fall\lab\lab11\example\sample.txt
Last modified on Sat Nov 16 17:34:44 CST 2019
```

How to Write a file



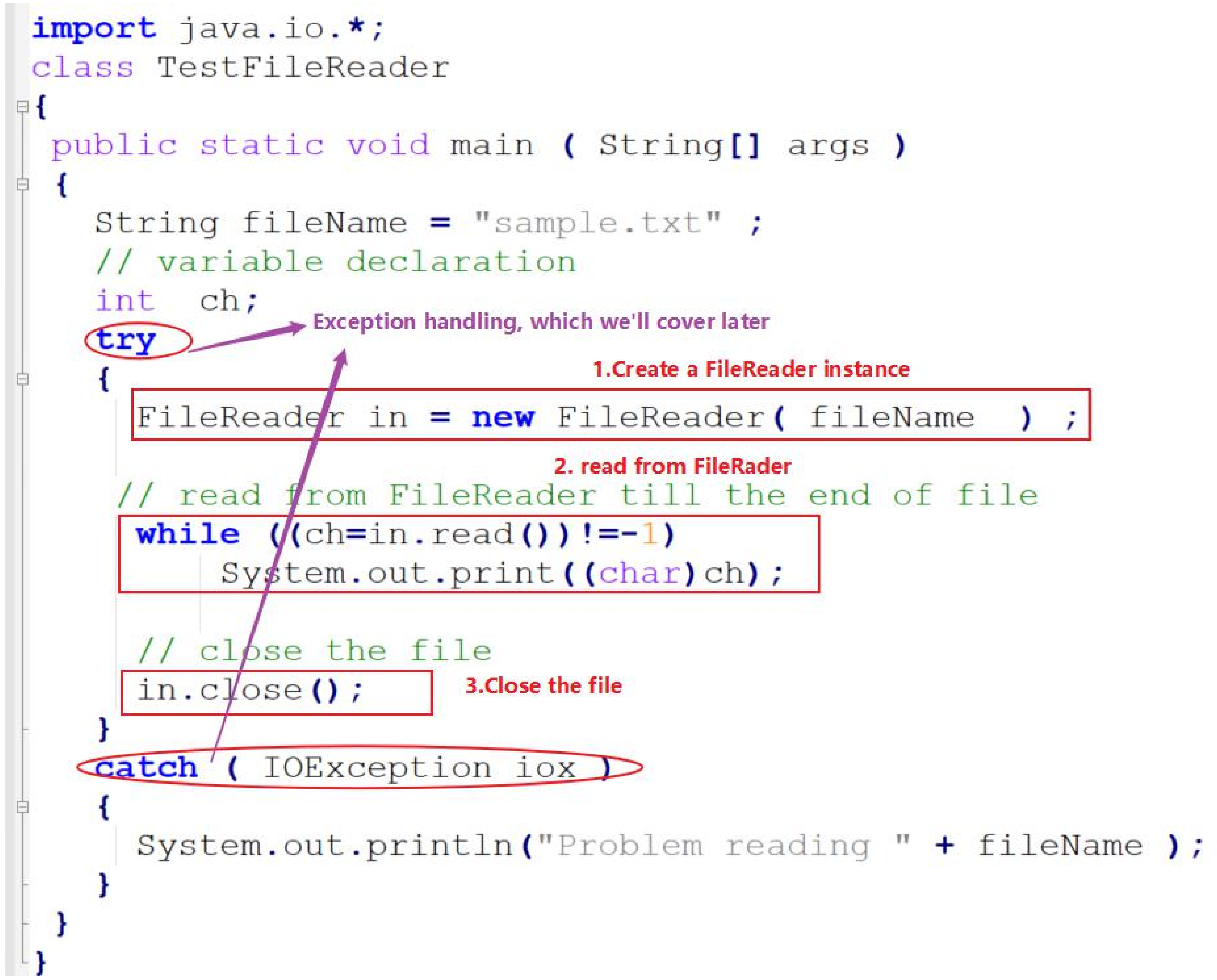


File handling using FileReader and FileWriter



```
//Creating a text File using FileWriter
import java.io.FileWriter;
import java.io.IOException;
class TestFileWriter
{
    public static void main(String[] args) throws IOException
    {
        // Accept a string
        String str = "File Handling in Java using "+
                " FileWriter and FileReader";
                                        1.Create a FileWriter instance and attach a file
        // attach a file to FileWriter
        FileWriter fw=new FileWriter("testfilewriter.txt");

        // read character wise from string and write
        // into FileWriter        2.Write char to the file
        for (int i = 0; i < str.length(); i++)
            fw.write(str.charAt(i));

        System.out.println("Writing successful");
        //close the file
        fw.close();    3.close the file
    }
}
```

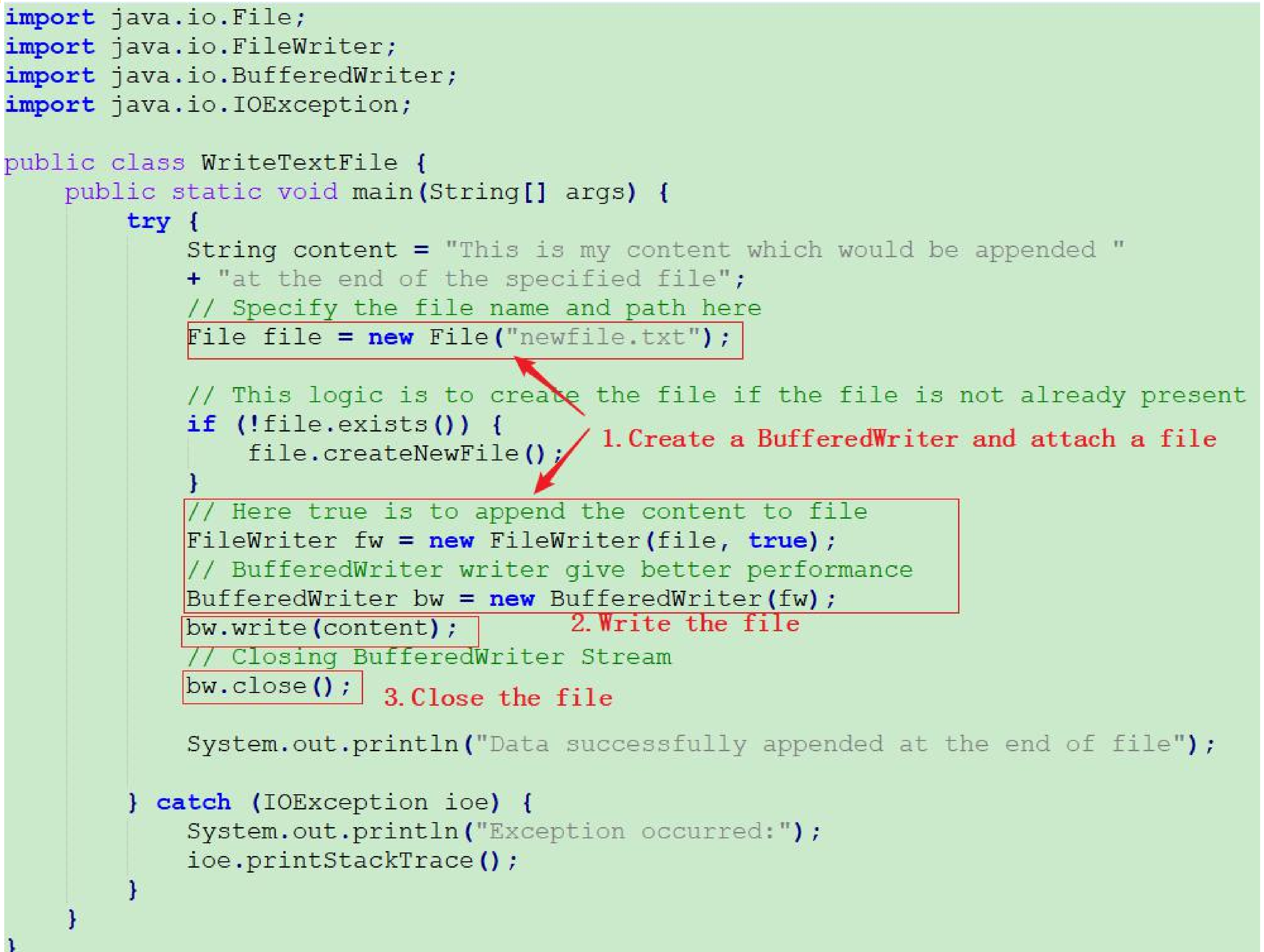
File handling using FileReader and FileWriter



```
import java.io.*;
class TestFileReader
{
 public static void main ( String[] args )
 {
   String fileName = "sample.txt" ;
   // variable declaration
   int  ch;
   try
   {
    FileReader in = new FileReader( fileName  ) ;

    // read from FileReader till the end of file
    while  ((ch=in.read())!=-1)
         System.out.print((char)ch);

    // close the file
    in.close();
   }
   catch ( IOException iox )
   {
    System.out.println("Problem reading " + fileName );
   }
 }
}
```

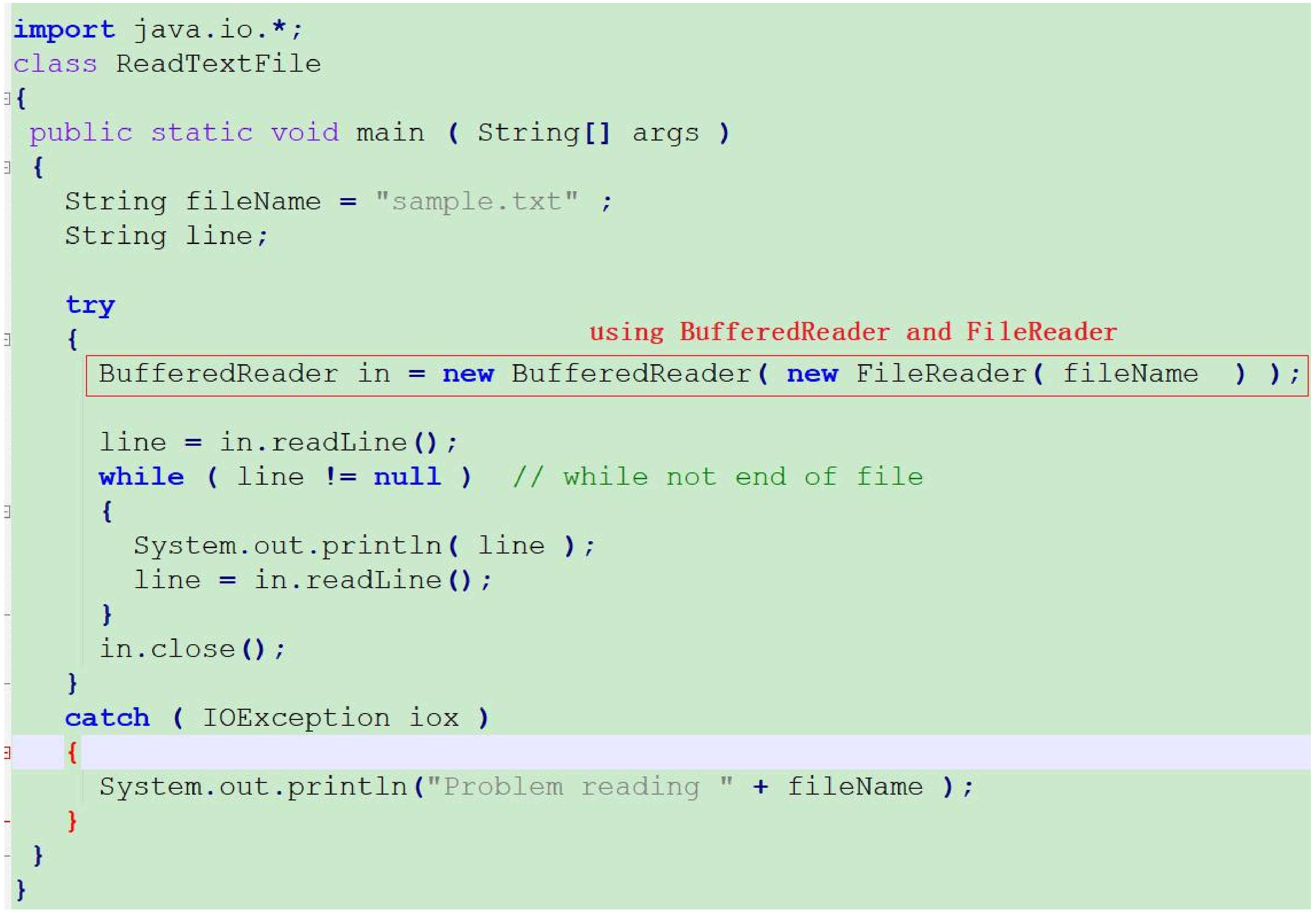
Exception handling, which we'll cover later
1.Create a FileReader instance
2. read from FileRader
3.Close the file

File handling using BufferedReader and BufferedWriter



```
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;

public class WriteTextFile {
    public static void main(String[] args) {
        try {
            String content = "This is my content which would be appended "
            + "at the end of the specified file";
            // Specify the file name and path here
            File file = new File("newfile.txt");

            // This logic is to create the file if the file is not already present
            if (!file.exists()) {
                file.createNewFile();   1. Create a BufferedWriter and attach a file
            }
            // Here true is to append the content to file
            FileWriter fw = new FileWriter(file, true);
            // BufferedWriter writer give better performance
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(content);
                                       2. Write the file
            // Closing BufferedWriter Stream
            bw.close();
                               3. Close the file

            System.out.println("Data successfully appended at the end of file");

        } catch (IOException ioe) {
            System.out.println("Exception occurred:");
            ioe.printStackTrace();
        }
    }
}
```

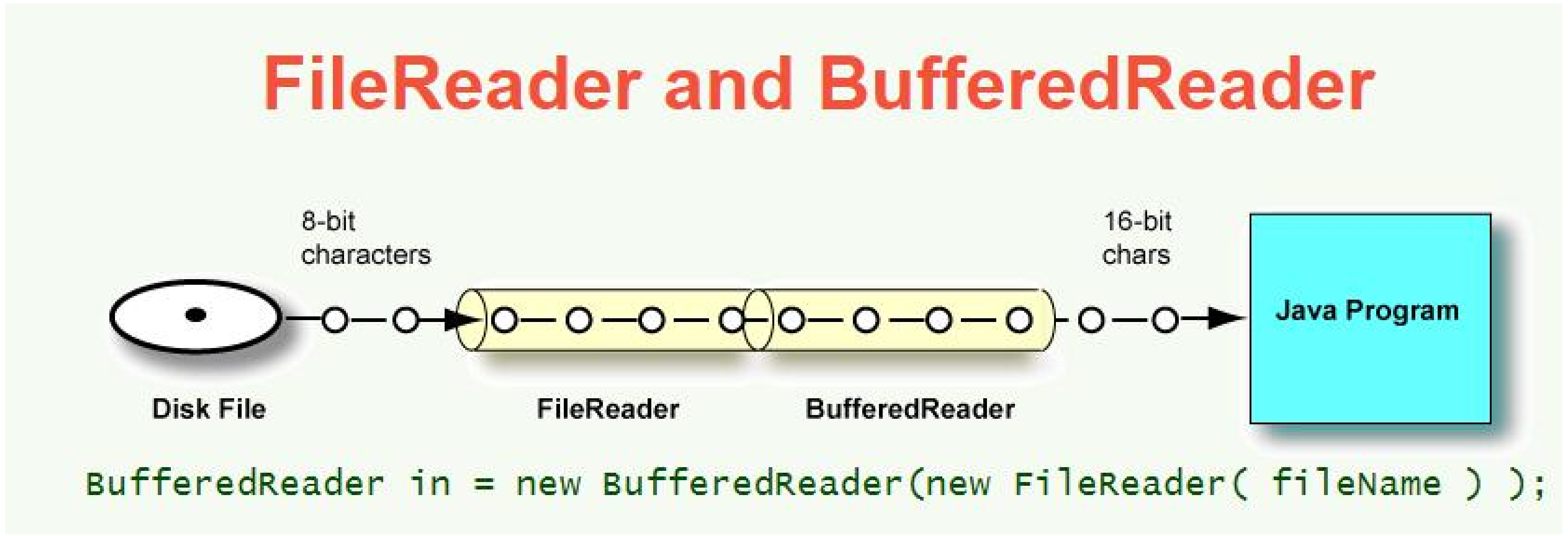
File handling using BufferedReader and BufferedWriter



using BufferedReader and FileReader

```
import java.io.*;
class ReadTextFile
{
 public static void main ( String[] args )
 {
   String fileName = "sample.txt" ;
   String line;

   try
   {
     BufferedReader in = new BufferedReader( new FileReader( fileName  ) );

     line = in.readLine();
     while ( line != null )  // while not end of file
     {
       System.out.println( line );
       line = in.readLine();
     }
     in.close();
   }
   catch ( IOException iox )
   {
     System.out.println("Problem reading " + fileName );
   }
 }
}
```

How to Read a file



```
BufferedReader in = new BufferedReader(new FileReader( fileName ) );
```

Exercises

Complete the exercises in the **2020S-Java-A-Lab-9.pdf** and submit to the blackboard as required.

THANK YOU

贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn