



Chapter 12 Part 2: File I/O

Reading data from/writing data to files

Yepang LIU (刘烨庞)

liuyp1@sustech.edu.cn

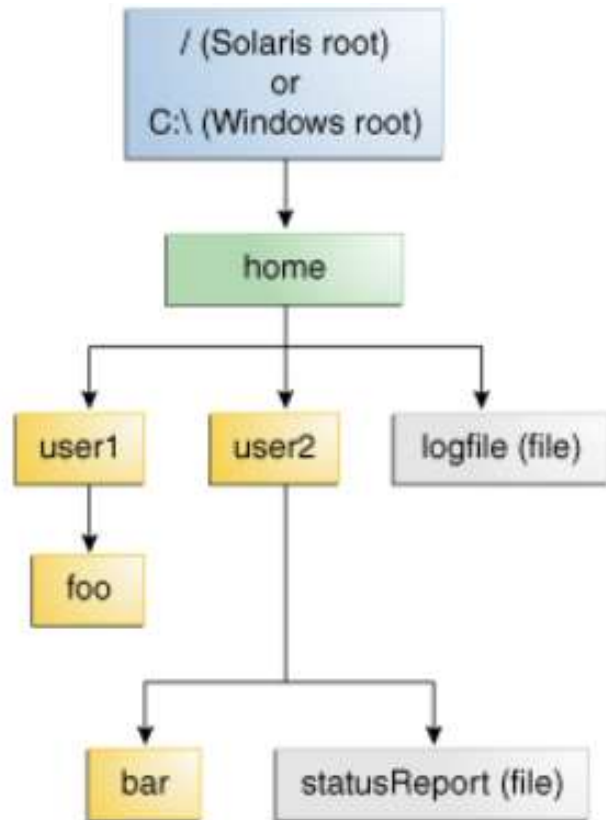
Why File I/O?

- ▶ Objects created in Java programs live in memory; they are removed by the garbage collector once they are not used anymore
- ▶ What if we want to persist the objects?

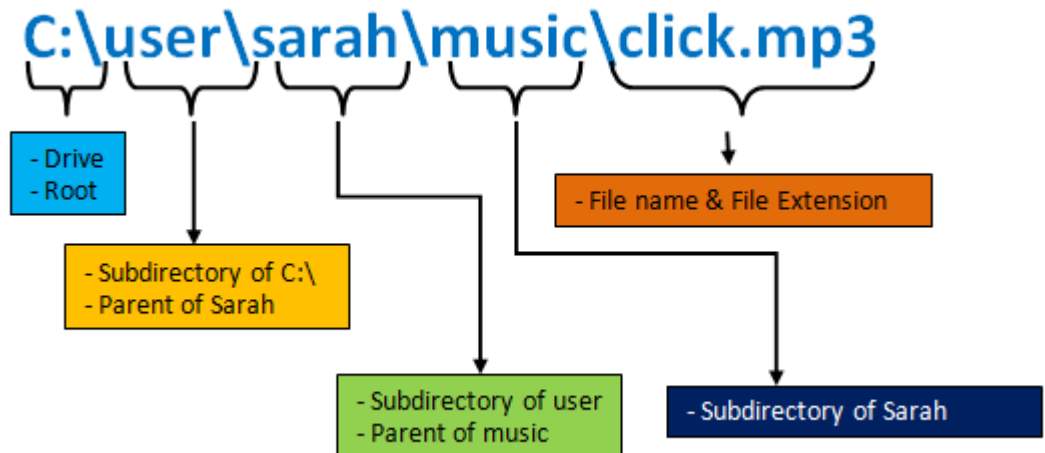


Locating a File

- File paths specify the location of individual files or directories



Sample Directory Structure





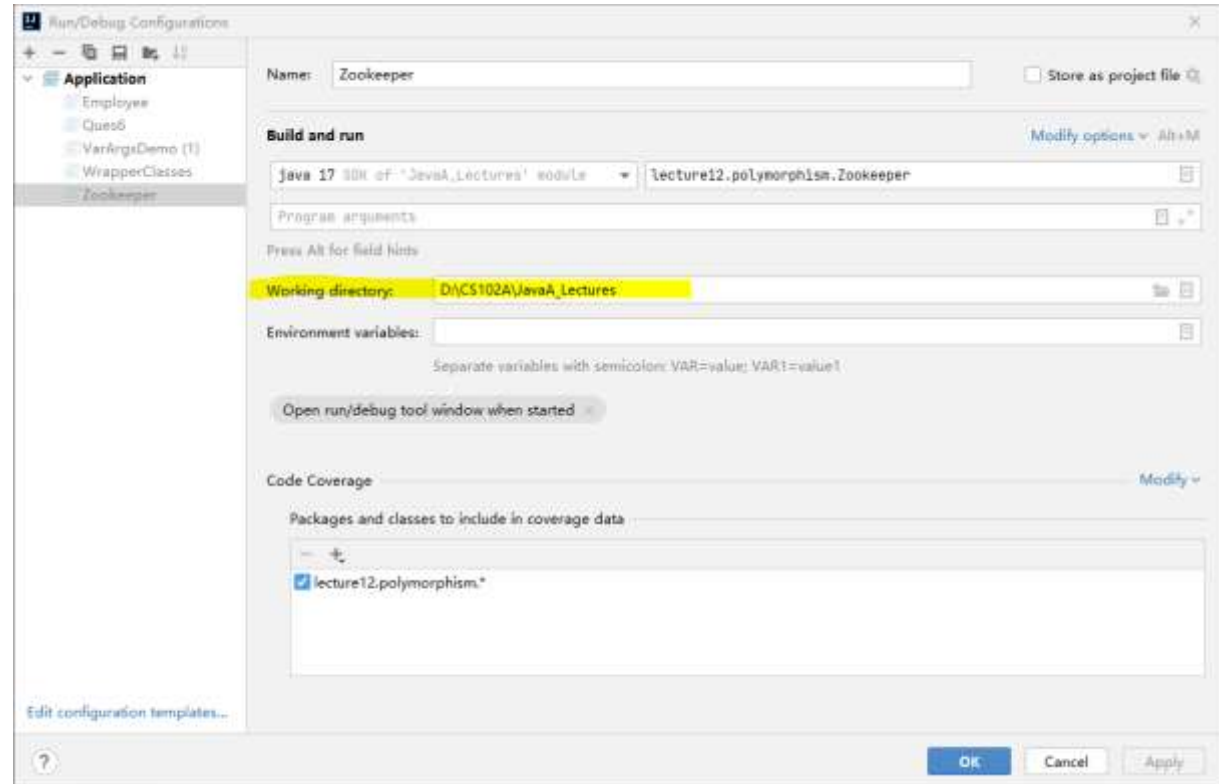
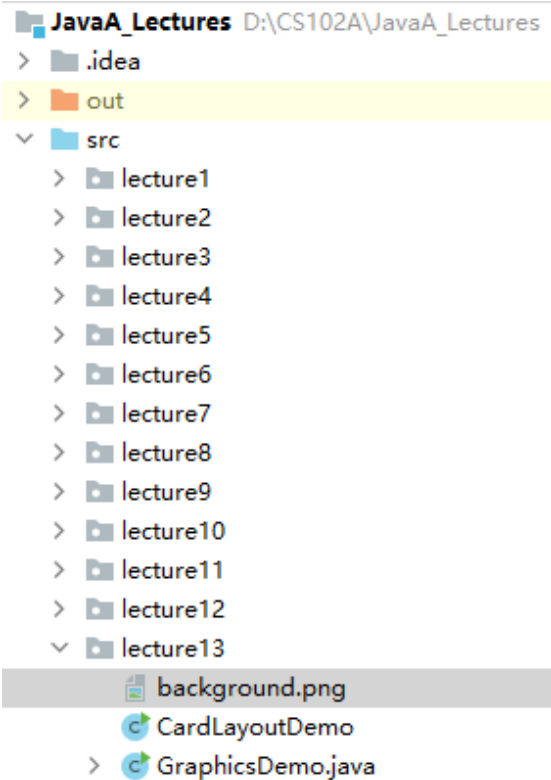
Locating a File

- ▶ Two types of paths
 - **Absolute path**: specifying the location of a file or directory from the root directory, i.e., a complete path (e.g., `D:\CS101\22Fall\lecture1.ppt`)
 - **Relative path**: the path related to the current working directory (e.g., `22Fall\lecture1.ppt` if the **working directory** is `D:\CS101`)

```
java.io.FileNotFoundException: D:\JavaPractice.txt (The system cannot find the path specified)
  at java.io.FileInputStream.open(Native Method)
  at java.io.FileInputStream.<init>(FileInputStream.java:120)
  at Proc.prac1.main(prac1.java:14)
```



Locating a File





Creating a File/Path Object

- ▶ Using `java.io.File` using String as absolute/relative path

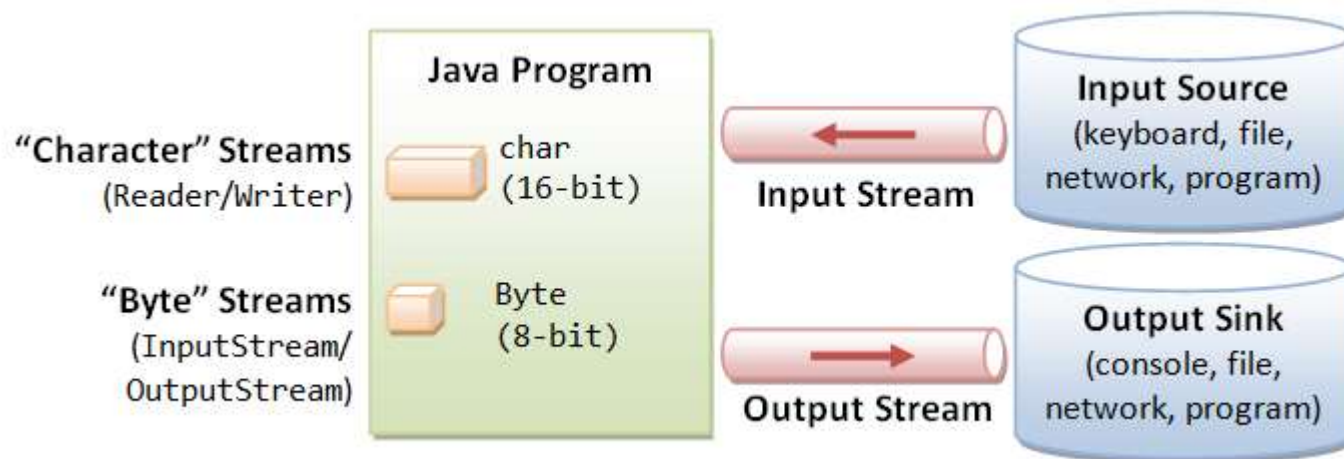
```
File text = new File("C:/temp/test.txt");
```

- ▶ Using `java.nio.Path` and `java.nio.Paths`

```
Path p1 = Paths.get("resources");  
Path p2 = Paths.get(args[0]);
```

Reading/Writing data from/to a file

- ▶ Read/write by bytes
- ▶ Read/write by characters
- ▶ Read/write by lines





Reading/Writing data from/to a file

- ▶ Read/write by bytes

```
java.io.FileInputStream.read()  
java.io.FileOutputStream.write(...)
```

- ▶ Read/write by characters

```
java.io.FileReader.read()  
java.io.FileWriter.write(...)
```

- ▶ Read/write by lines

```
java.io.BufferedReader.readLine()  
java.nio.file.Files.readAllLines()  
java.io.BufferedWriter.write(...)  
java.nio.file.Files.write(...)
```

https://www3.ntu.edu.sg/home/ehchua/programming/java/j5b_io.html



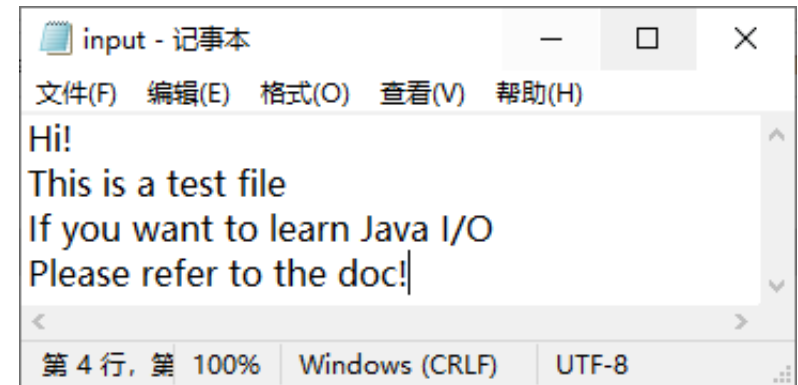
Reading lines from a file

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

public class IO Demo {

    public static void readFileByLines() throws IOException {
        String filename = "src/lecture14/input.txt";
        Path filepath = Paths.get(filename);
        List<String> lines = Files.readAllLines(filepath);

        for(String l: lines){
            System.out.println(l);
        }
    }
}
```



```
filename = "src/lecture14/input.txt"
filepath = {WindowsPath@781} "src\lecture14\input.txt"
lines = {ArrayList@782} size = 4
> 0 = "Hi!"
> 1 = "This is a test file"
> 2 = "If you want to learn Java I/O"
> 3 = "Please refer to the doc!"
```

Writing data to a file

```
public static void writeToFileByLines() throws IOException {  
    List<String> data = new ArrayList<>();  
    data.add("Hello ");  
    data.add("World!");  
    data.add("Let's learn Java!");  
    Path outputPath = Paths.get("src/lecture14/output.txt");  
    Files.write(outputPath, data);  
}
```



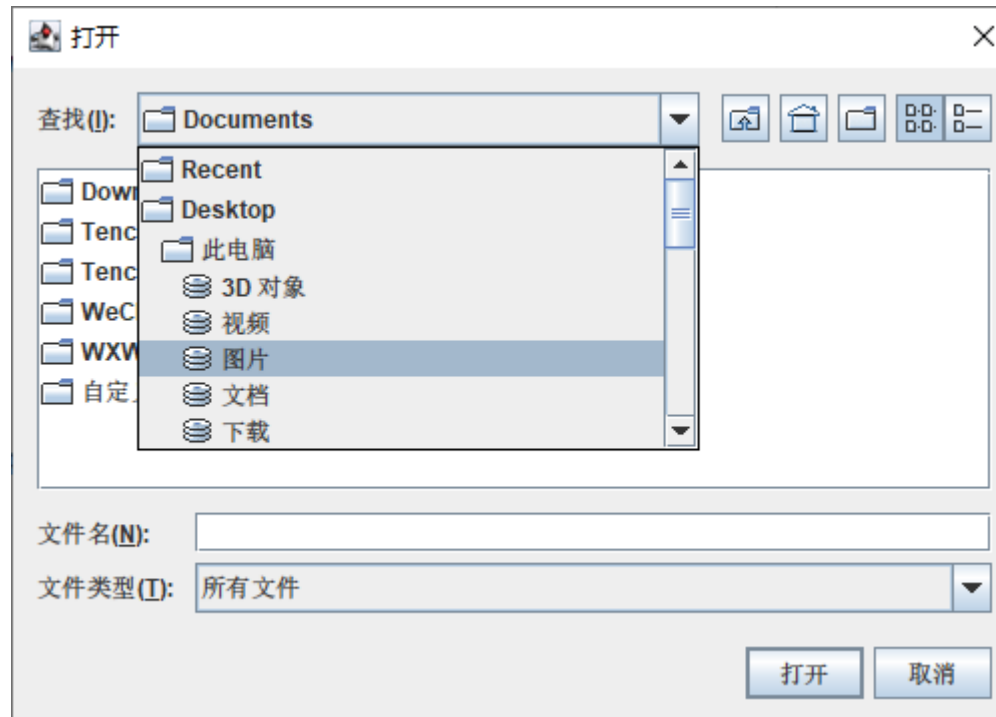


Opening Files with JFileChooser



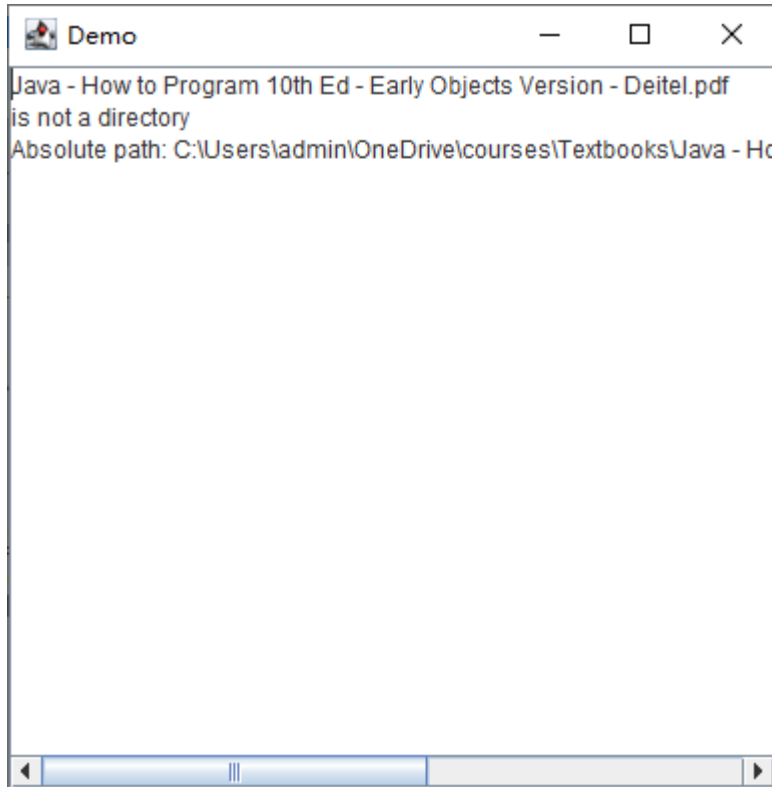
Demo

JFileChooser allows users to browse the file system and select files or directories

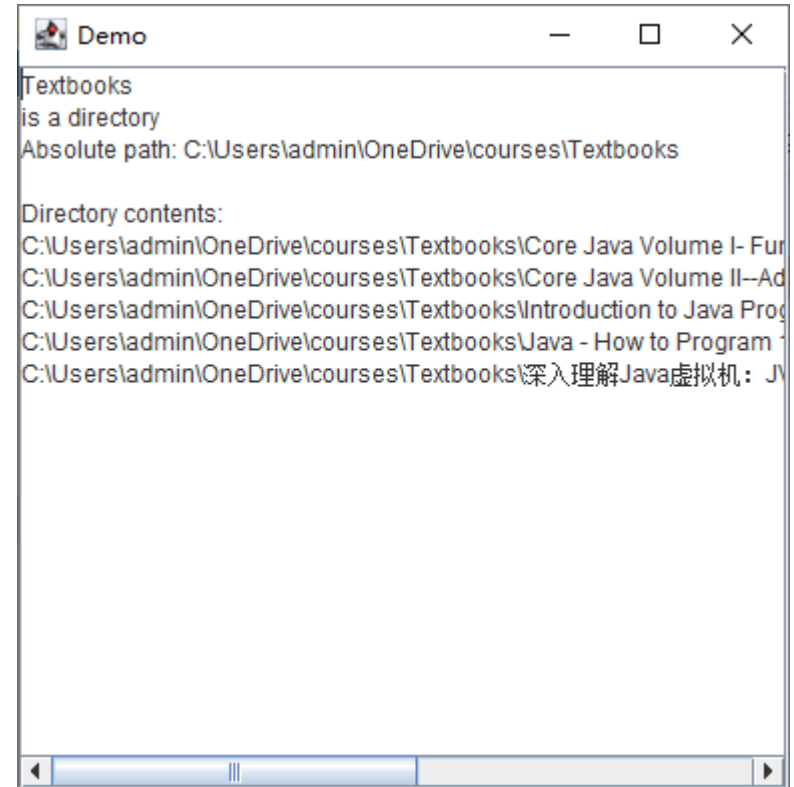




Demo

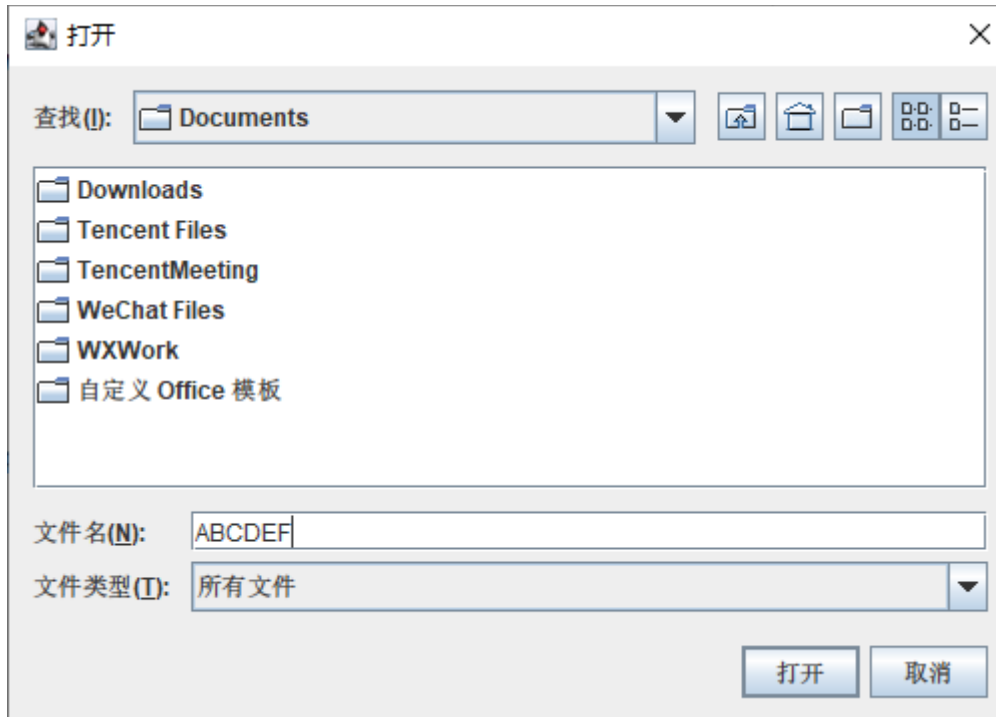


A file is chosen.



A directory is chosen.

Demo



Window

```
public class FileChooserDemo extends JFrame {
    // display file contents
    private final JTextArea outputArea;

    public FileChooserDemo() throws IOException {
        super( title: "Demo");

        outputArea = new JTextArea();
        // output is scrollable
        add(new JScrollPane(outputArea));

        analyzePath();
    }

    public static void main(String[] args) throws IOException {
        FileChooserDemo app = new FileChooserDemo();
        app.setSize( width: 400, height: 400);
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.setVisible(true);
    }
}
```



File Chooser Dialog

```
private Path getFileOrDirectory(){
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

    int result = fileChooser.showOpenDialog( parent: this);

    if(result == JFileChooser.CANCEL_OPTION){
        System.exit( status: 1);
    }

    // return Path representing the selected file
    return fileChooser.getSelectedFile().toPath();
}
```




Display file or directory content in TextArea

```
public void analyzePath() throws IOException {
    Path path = getFileOrDirectory();
    if(Files.exists(path)){
        // gather file or directory information
        StringBuilder builder = new StringBuilder();
        builder.append(String.format("%s\n", path.getFileName()));
        builder.append(String.format("%s a directory\n", Files.isDirectory(path)? "is": "is not"));
        builder.append(String.format("Absolute path: %s\n", path.toAbsolutePath()));

        // output directory listing
        if(Files.isDirectory(path)){
            builder.append(String.format("\nDirectory contents:\n"));

            DirectoryStream<Path> directoryStream = Files.newDirectoryStream(path);
            for(Path p: directoryStream){
                builder.append(String.format("%s\n", p));
            }
        }

        outputArea.setText(builder.toString()); // display
    }
    else{
        JOptionPane.showMessageDialog( parentComponent: this, message: path.getFileName()+" doesn't exist.",
            title: "ERROR", JOptionPane.ERROR_MESSAGE);
    }
}
```