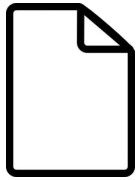
# IO and Exception

Suttipong Meuntaboot

# File System



- **Path** คือ เส้นทางเพื่อบอกตำแหน่งของไฟล์ หรือ ไดเรคทอรี (โฟลเดอร์)
- **Root directory** คือ ไดเรคทอรีหลักที่รวบรวมไดเรคทอรีทั้งหมด
- **Current directory** คือ ไดเรคทอรีปัจจุบัน (แทนด้วย . )
- **Parent directory** คือ ไดเรคทอรีที่บรรจุไดเรคทอรีปัจจุบันเอาไว้ (แทนด้วย .. )
- **File separator** คือ สัญลักษณ์ที่ใช้แบ่งชื่อไฟล์ หรือ ไดเรคทอรี ใน path
  - Linux base ใช้ **/** เช่น `/home`
  - Windows ใช้ **** เช่น `C:\Users\PC708`

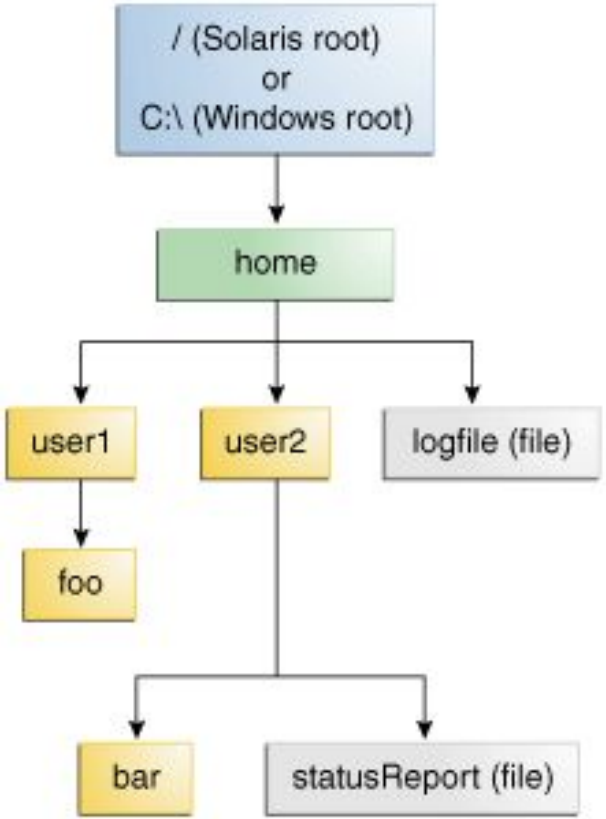
# File Name

- **File Name** คือ ชื่อไฟล์ รวมนามสกุลด้วย
  นามสกุลของไฟล์ จะอยู่หลัง . ทางขวาสุด
  โปรแกรมเมอร์มีสิทธิ์ตั้งนามสกุลของไฟล์ได้อิสระ แต่ควรใช้ตามมาตรฐาน

- บางไฟล์ อาจไม่มีนามสกุล ทำให้ดูคล้ายไดเรคทอรี (นิยมใช้ในตระกูล Linux)

- ถ้าเป็นชื่อไดเรคทอรี โดยทั่วไปนิยมใส่ Separator ต่อท้าย
  ถ้าเขียน Path `C:\Users\PC708`
  จะเข้าใจได้ว่า PC708 เป็นไฟล์
  ถ้าเขียน Path `C:\Users\PC708\`
  จะเข้าใจได้ว่า PC708 เป็นไดเรคทอรี

# Directory Tree




```
/
|
|--- home
       |
       |--- user1
       |      |
       |      |--- foo
       |
       |--- user2
       |      |
       |      |--- bar
       |      |
       |      |--- statusReport
       |
       |--- logfile
```

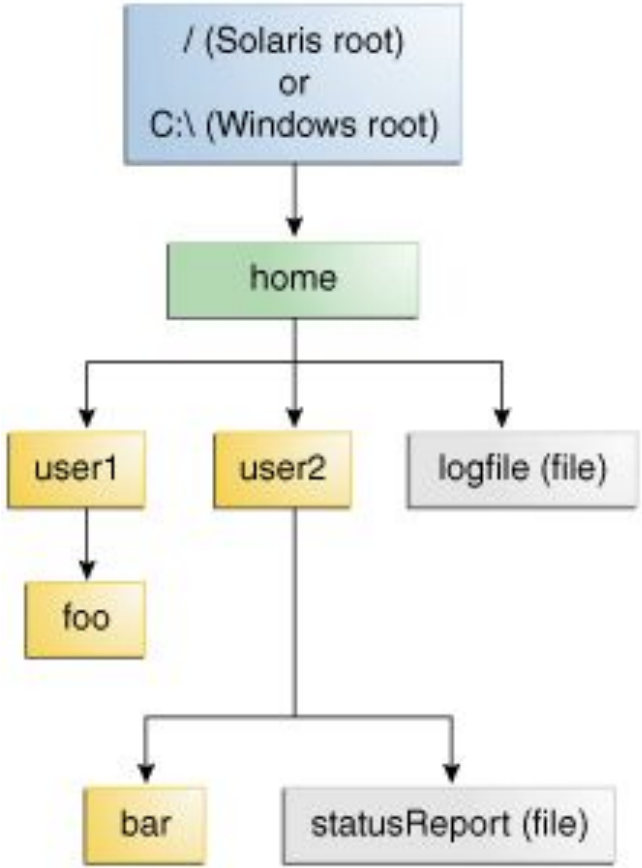
# Absolute Path

- **Absolute Path**
  - การอ้างอิงตำแหน่งจาก root directory
  - Linux base root directory อ้างอิงจาก `/`
  - Windows root directory อ้างอิงจาก drive letter เช่น `C:` , `D:`
- ถ้าต้องการอ้างอิงที่อยู่ไฟล์ `statusReport`

  `/home/user2/statusReport`

  `C:\home\user2\statusReport`

- ข้อควรระวัง การอ้างอิงแบบ absolute path จะต้องมี Directory Tree ที่เหมือนกันเท่านั้น จึงจะอ้างอิงได้ (จาก Root Directory)

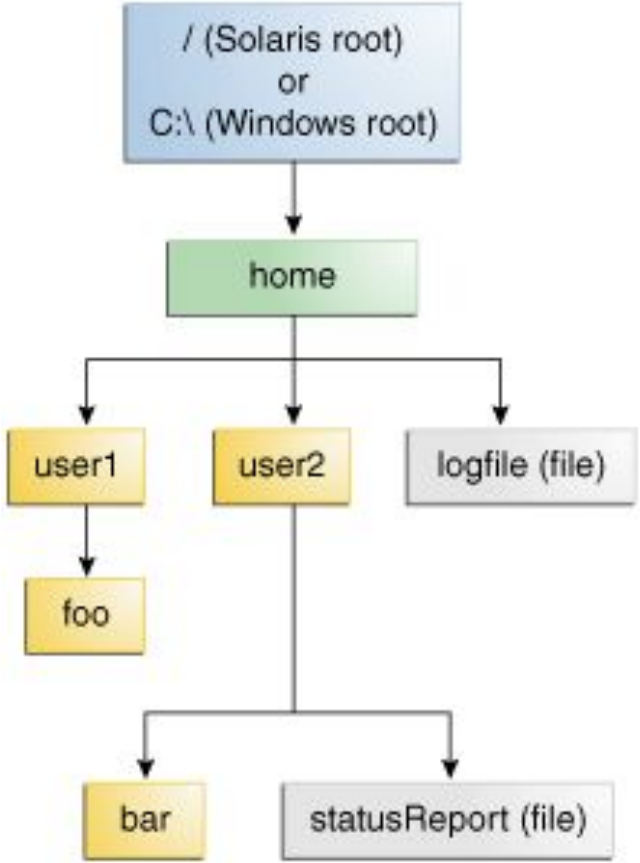


# Relative Path

- **Relative Path**
  - การอ้างอิงตำแหน่งจาก Current Directory
- ถ้าต้องการอ้างอิงที่อยู่ไฟล์ `statusReport` จาก `foo`
  - Linux base

    ```
    ../../user2/statusReport
    ```
  - Windows

    ```
    ..\..\user2\statusReport
    ```
- ข้อควรระวัง การอ้างอิงแบบ Relative Path จะต้องมี Directory Tree ที่เหมือนกันเท่านั้น จึงจะอ้างอิงได้ (เฉพาะส่วนที่จะอ้างอิง)



# Java File API

- `java.io.File`

- Create File

```
File file = new File("test.txt")

file.createNewFile() -> return boolean
```

# Java File API

- File System Property

```
// current working directory
String cwd = System.getProperty("user.dir");

// file system separator
String fileSep = System.getProperty("file.separator");
```

ข้อควรระวัง file separator ของ windows คือ \
เป็นตัวเดียวกันกับ Escape Character
ถ้าจะใส่ใน String โดยตรงต้องใช้ \\ เช่น "C:\\Users"

# Java File API

- File Status
  - `file.exist()`
  - `file.isFile()`
  - `file.isDirectory()`
  - `file.isHidden()`

return boolean

# Read File

- Read File
  - `File file = new file("test.txt")`
  - `FileReader fileReader = new FileReader(file);`

  ใช้ BufferedReader

```
BufferedReader reader = new BufferedReader(fileReader);
String line = "";
while ((line = reader.readLine()) != null) {
      // ดำเนินการทีละบรรทัดกับตัวแปร line
}
reader.close();
```

# Write File

- Write File
  - `File file = new File("test.txt");`
  - `FileWriter fileWriter = new FileWriter(file[, append_parameter]);`
    - append_parameter default is `false` (แปลว่าเขียนทับ)
    - append_parameter is `true` แปลว่า เขียนต่อจากข้อมูลเดิมในไฟล์

    ใช้ `BufferedWriter`

  - `BufferedWriter writer = new BufferedWriter(fileWriter);`
    - `writer.write("test write");`
    - `writer.append("append string");` // append parameter `true`
    - `writer.newLine();`
    - `writer.close();`

# Write File

- Write File
  - `File file = new File("test.txt");`
  - `FileWriter fileWriter = new FileWriter(file[, append_parameter]);`
    - append_parameter default is `false` (แปลว่าเขียนทับ)
    - append_parameter is `true` แปลว่า เขียนต่อจากข้อมูลเดิมในไฟล์

    ใช้ `PrintWriter`

  - `PrintWriter printWriter = new PrintWriter(fileWriter);`
    - `printWriter.print("test");`
    - `printWriter.printf("test %s", "test");`
    - `printWriter.close();`

# IO Exceptions

- เป็น Checked exception ต้องใช้ try/catch เสมอ
- `java.io.IOException`
  - `FileNotFoundException`
  - `EOFException` (End of File)