Warm Up

slido.com
code: #su_cse123

```
public static int recursiveMystery(int n) {
    if (n <= 1) {
        return 1;
    } else {
        return recursiveMystery( n: n - 2)
                + recursiveMystery( n: n - 1);
    }
}
```

What's the output of `recursiveMystery(5)`?

Recursive Programming

Hitesh Boinpally
Summer 2023

Agenda

- Recursion Review
- Practice
- Recursion with Files

Agenda

- Recursion Review 
- Practice
- Recursion with Files

Road Map - Recursion

- Friday
  - Introduce idea of “recursion”
  - Goal: Understand idea of recursion and read recursive code
- Tuesday
  - Practice reading recursive code
- Wednesday (Today)
  - More complex recursive examples
  - Goal: Identify recursive structure in problems and write recursive code
- Thursday
  - Practice writing recursive code

Recursion Intro

- **Recursion:** Defining something in terms of itself
- **Recursive Programming:** Writing methods that call themselves to solve problems
  - Helpful to solve specific problems
  - Equally as powerful as iterative solutions

Recursive Programming

Two cases to always keep in mind:

1. **Base Case:**
   - Stopping point, how to know we're "done"
   - Easiest / smallest thing to calculate
2. **Recursive Case:**
   - Do "one step" of the problem
   - Pass on the work to the next method call

- Some problems may have multiple base / recursive cases!

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

n	5

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

n	5

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

n	

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

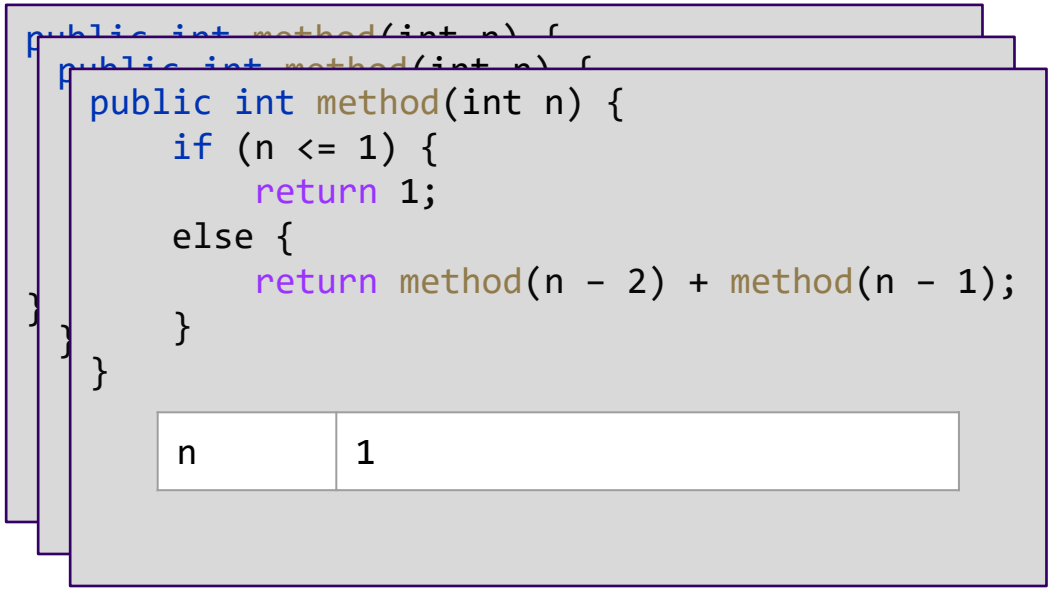
n	3

Warm-Up Visualized

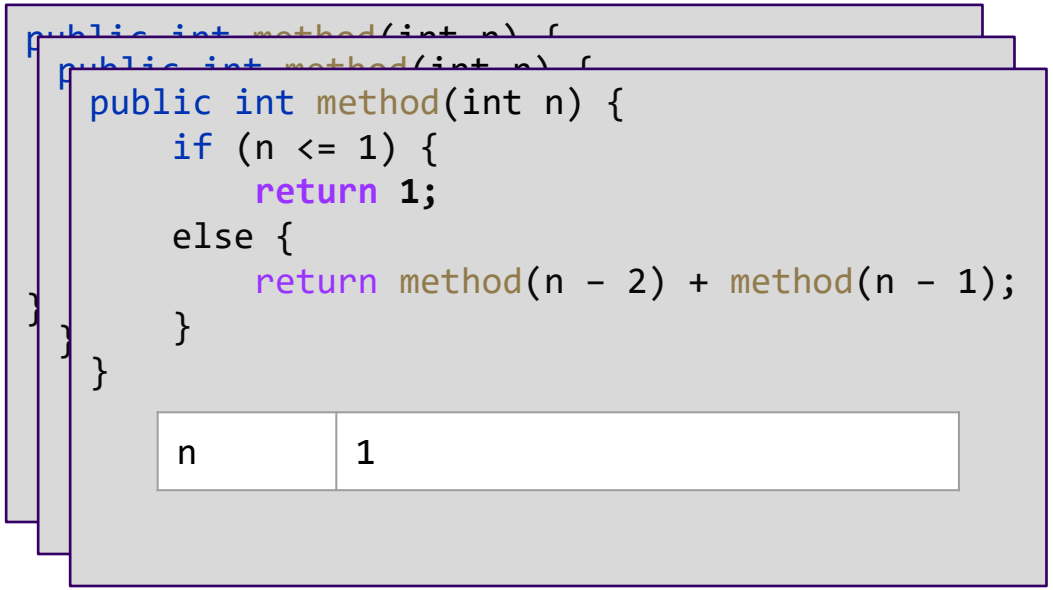
```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

n	3

Warm-Up Visualized



Warm-Up Visualized



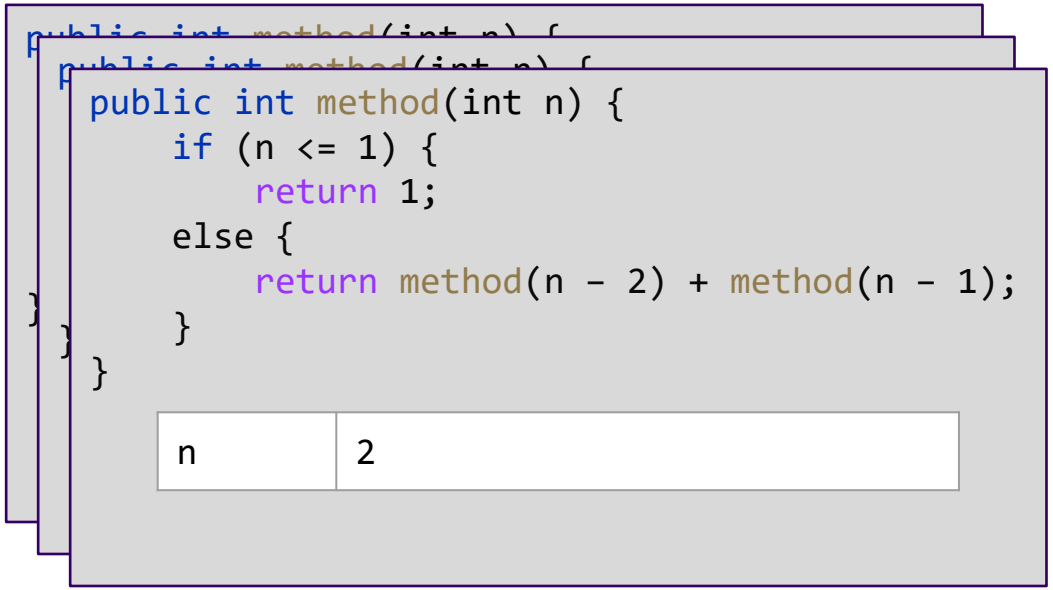
Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

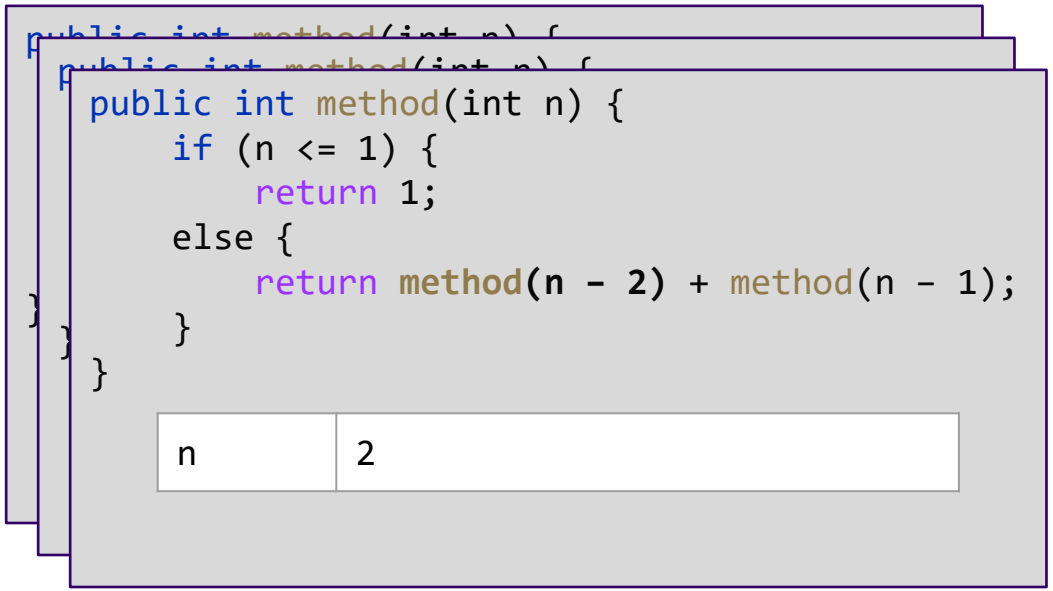
1

n	3

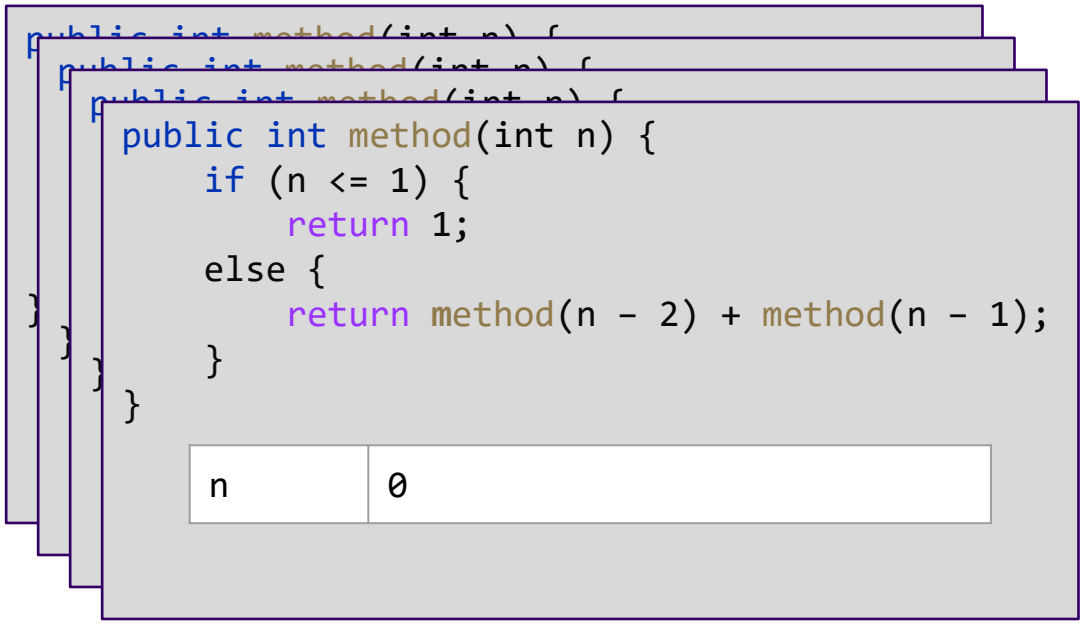
Warm-Up Visualized



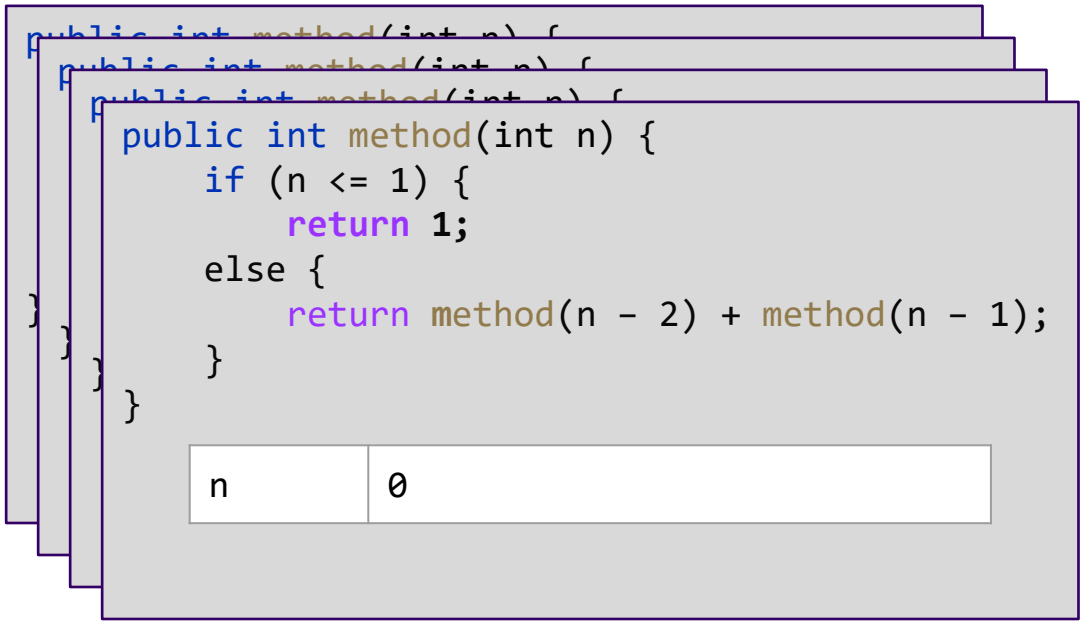
Warm-Up Visualized



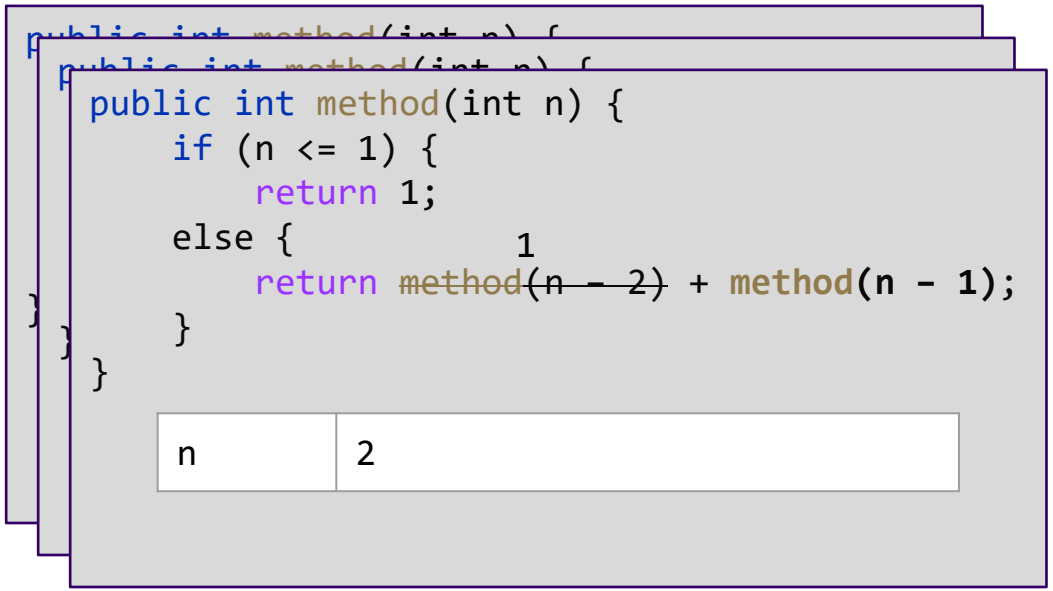
Warm-Up Visualized



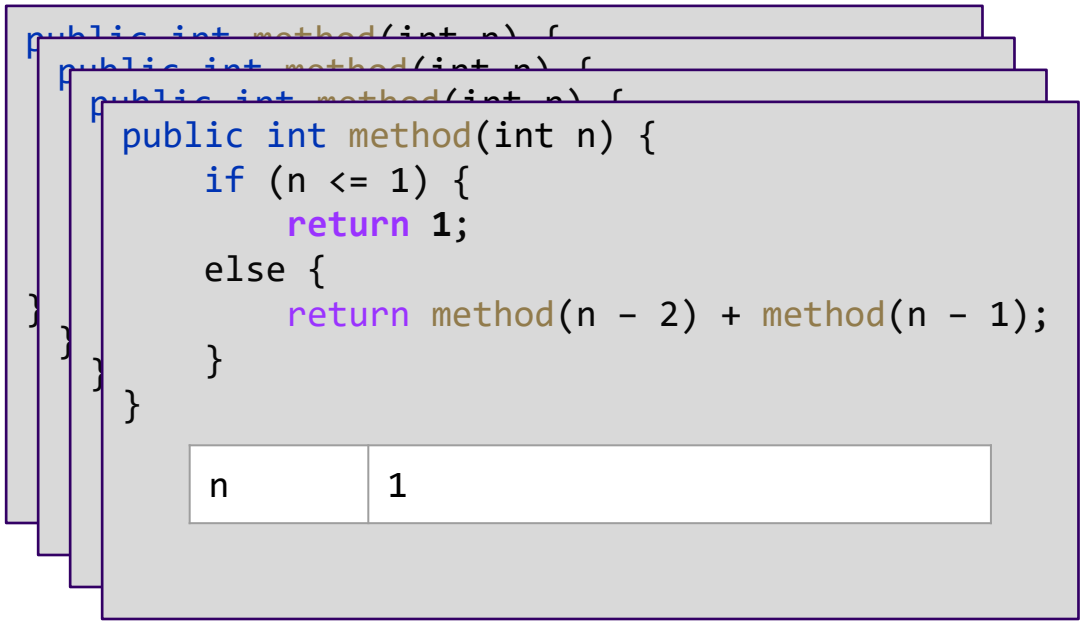
Warm-Up Visualized



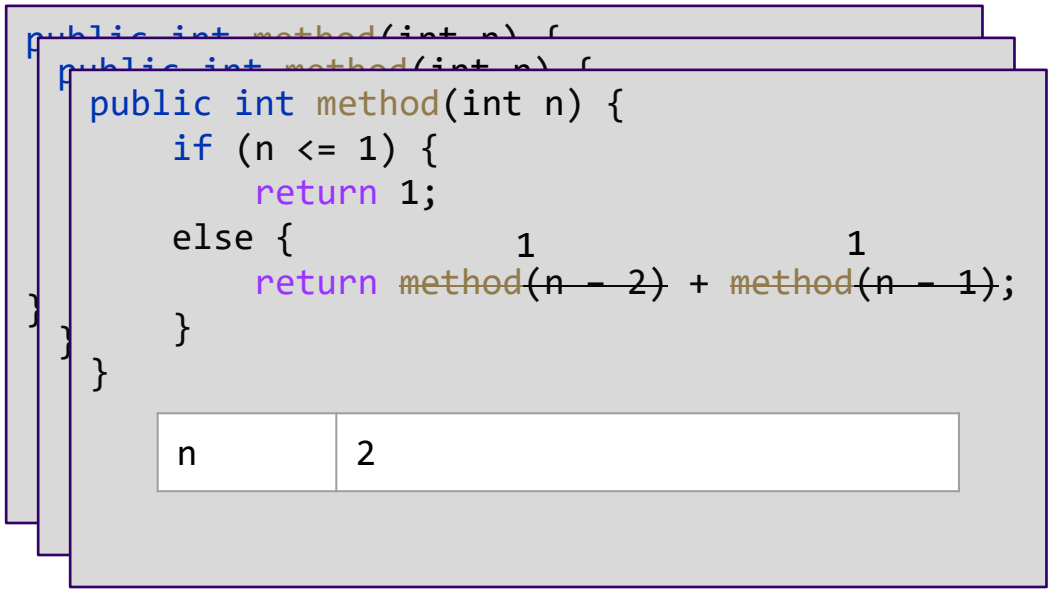
Warm-Up Visualized



Warm-Up Visualized



Warm-Up Visualized



Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {
        return method(n - 2) + method(n - 1);
    }
}
```

1 2

n	3

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {                3
        return method(n - 2) + method(n - 1);
    }
}
```

n	5

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {                3
        return method(n - 2) + method(n - 1);
    }
}
```

n	5

Warm-Up Visualized

```
public int method(int n) {
    if (n <= 1) {
        return 1;
    else {              3                   5
        return method(n - 2) + method(n - 1);
    }
}
```

n	5

Agenda

- Recursion Review
- Practice 
- Recursion with Files

Agenda

- Recursion Review
- Practice
- Recursion with Files 

Recursion with Files

Exercise: Write a recursive method print that takes in a `File` and prints out information about it.

- If the `File` object represents a normal file print its name
- If the `File` object represents a directory, print out its name and information about every file/directory inside of it indented by a level.

Recursion with Files

Exercise: Write a recursive method print that takes in a `File` and prints out information about it.

- If the `File` object represents a normal file print its name
- If the `File` object represents a directory, print out its name and information about every file/directory inside of it indented by a level.

Recursion with Files

Exercise: Write a recursive method print that takes in a `File` and prints out information about it.

- If the `File` object represents a normal file print its name
- If the `File` object represents a directory, print out its name and information about every file/directory inside of it indented by a level.

```
123_folder
    assignments
        warm-up
        mini-git
    practice
        lesson1
        lesson2
    syllabus
    LinkedIntList
```

Definition of a File

- A File is either:
  1. A simple file
  2. A directory containing more files
- A directory can be nested to any depth

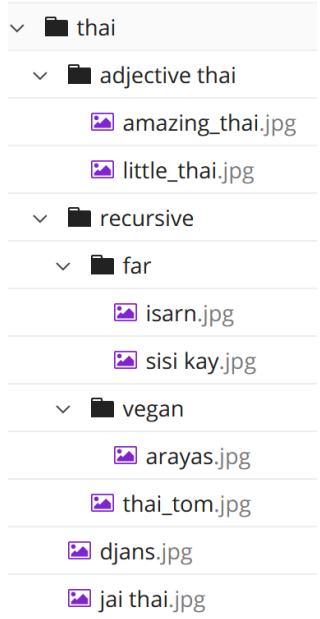
File Methods

- `getName()` – Returns the name of this file
- `isDirectory()` – Returns whether this file is a directory or not
- `listFiles()` – Returns a `File[]` of all subfiles in this directory

`print` Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	thai
indent	0
subFile	

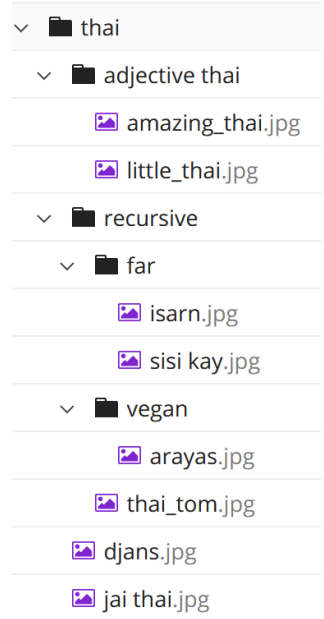


Console output:
thai

`print` Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	thai
indent	0
subFile	adjective thai

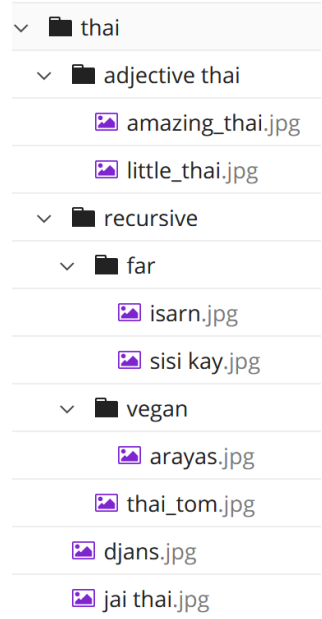


Console output:
thai

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	adjective thai
indent	1
subFile	



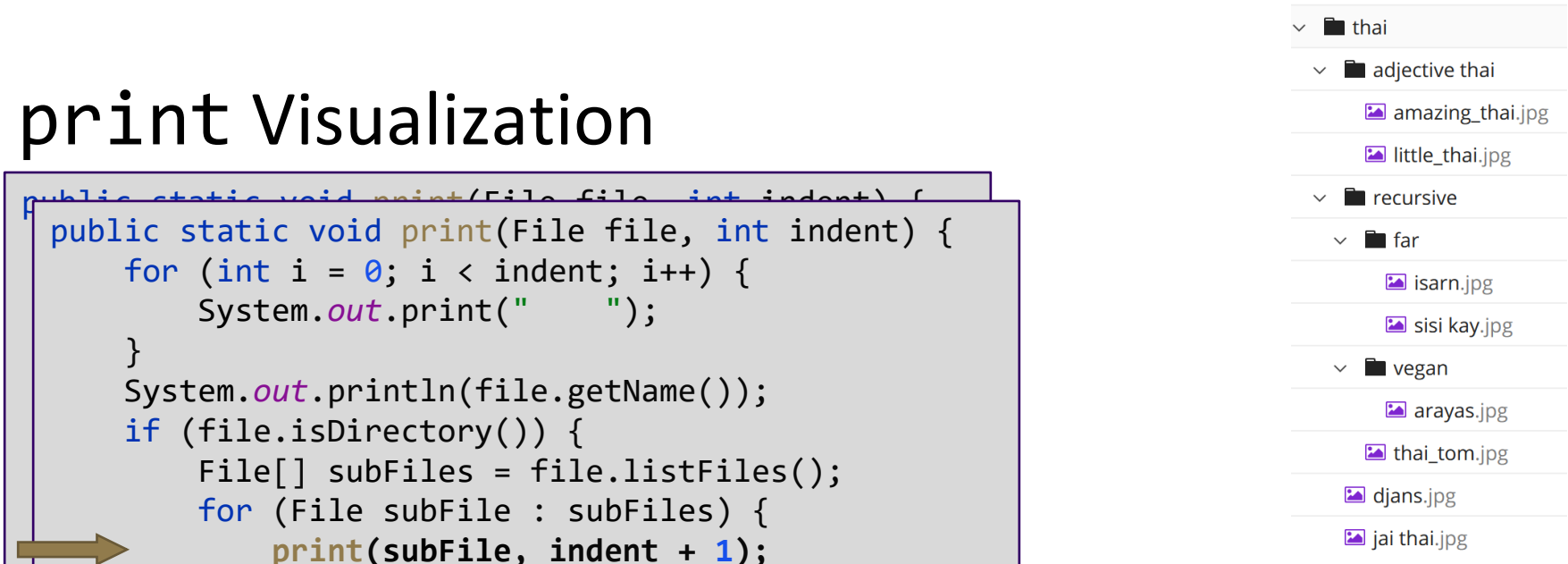
Console output:

```
thai
    adjective thai
```

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	adjective thai
indent	1
subFile	amazing_thai.jpg



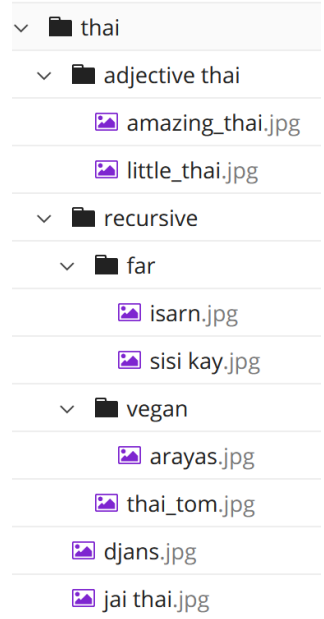
Console output:

```
thai
    adjective thai
```

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	amazing_thai.jpg
indent	2
subFile	



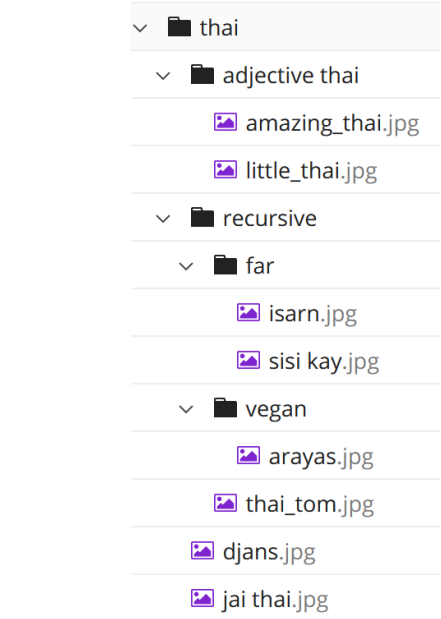
Console output:

```
thai
    adjective thai
        amazing_…
```

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	adjective thai
indent	1
subFile	little_thai.jpg



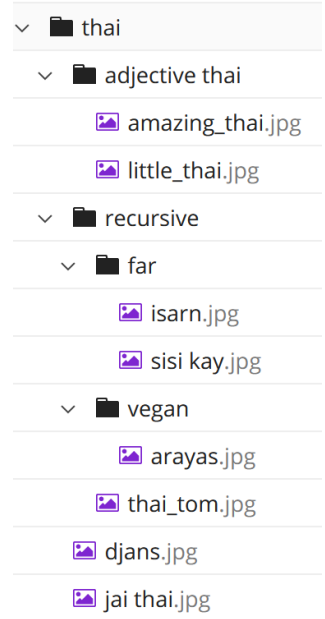
Console output:

```
thai
    adjective thai
        amazing_…
```

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	little_thai.jpg
indent	2
subFile	



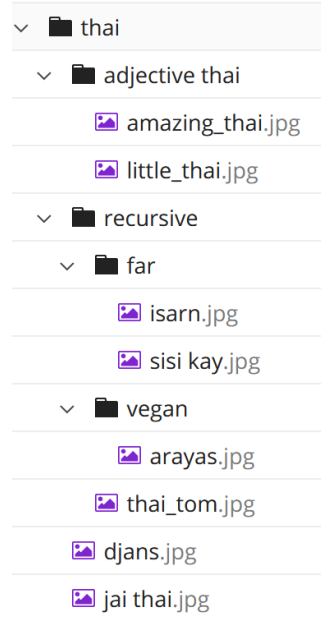
Console output:

```
thai
    adjective thai
        amazing_…
        little_…
```

print Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	adjective thai
indent	1
subFile	



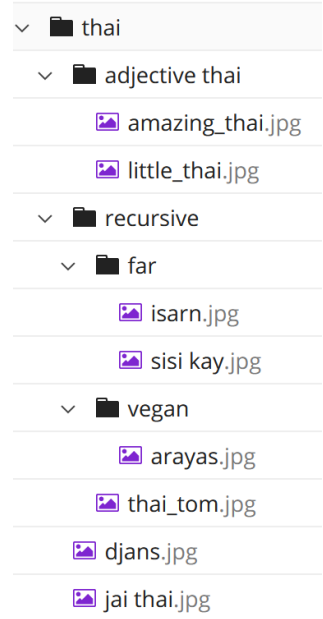
Console output:

```
thai
    adjective thai
        amazing_…
        little_…
```

`print` Visualization

```
public static void print(File file, int indent) {
    for (int i = 0; i < indent; i++) {
        System.out.print("    ");
    }
    System.out.println(file.getName());
    if (file.isDirectory()) {
        File[] subFiles = file.listFiles();
        for (File subFile : subFiles) {
            print(subFile, indent + 1);
        }
    }
}
```

file	thai
indent	0
subFile	recursive



Console output:

```
thai
    adjective thai
        amazing_…
        little_…
```

C2 – Mondrian Art

- This is a difficult assignment conceptually
- It can be challenging to debug / understand
- Make sure to start **early**
- (Possibly) multiple base / recursive cases