Lecture 69: Local Variables and Scope

▼ What is meant by **Scope** of a variable ?

It represents the area within the code where the variable can be accessed

“In Scope” means the particular variable can be accessed by the current executing block

▼ When are variables said to be **in scope** ?

They’re in scope for the block `{}` they are declared in.

They’re available for any nested blocks inside the declared block.

▼ When are variables **out of scope** ?

They’re unavailable for any block outside the block that they were declared in.

▼ What are some scope **best practices** ?

- Declare and initialize the variables at the same place/block if possible
- Declare them in the narrowest scope possible.

▼ How is the scope of variables different for **switch blocks** than for if-else blocks ?

```
switch (value) {
    case 1:
        int i = 10;
        break;
    case 2:
        i = value; // this is accepted!
        System.out.println(i);
        break;
}
```

- So, a variable declared inside a particular `case` is in scope for use for the cases *after that*, not the previous ones.
- Also, the variables declared inside cases are not accessible outside the `switch` block itself