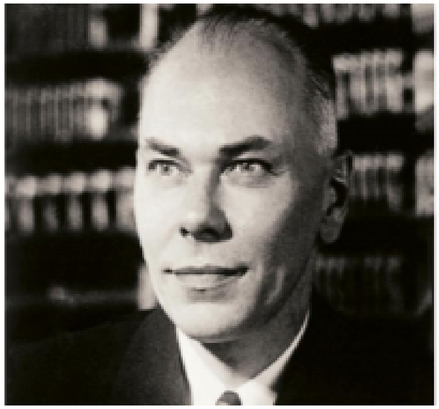
Topic 6
loops, figures, constants

"A fine balance must be maintained between computation time and the ensuing complexity of the coding."

Rear Admiral **Grace Hopper, Ph.D.** and **Howard Aiken, Ph.D.**

"A Manual of Operation for the Automatic Sequence Controlled Calculator"



Class constants and scope

reading: 2.4

Limitations of variables

[redacted]

- Problem: A variable in one method can't be seen in others.

```
public static void main(String[] args) {
     int size = 4;
     topHalf();
     printBottom();
}

public static void topHalf() {
     for (int i = 1; i <= size; i++) {     // ERROR: size not found
          ...
     }
}

public static void bottomHalf() {
     for (int i = size; i >= 1; i--) {     // ERROR: size not found
          ...
     }
}
```

Scope

- **scope**: The part of a program where a variable exists.
  - From its declaration to the end of the `{ }` braces
    - A variable declared in a `for` loop exists only in that loop.
    - A variable declared in a method exists only in that method.

```
public static void example() {
    int x = 3;
    for (int i = 1; i <= 10; i++) {
        System.out.println(x);
    }
    // i no longer exists here
} // x ceases to exist here
```

`i`'s scope

`x`'s scope

Scope implications

- Variables without overlapping scope can have same name.

```
for (int i = 1; i <= 100; i++) {
    System.out.print("/");
}
for (int i = 1; i <= 100; i++) {    // OK
    System.out.print("\\");
}
int i = 5;                          // OK: outside of loop's scope
```

- A variable can't be declared twice or used out of its scope.

```
for (int i = 1; i <= 100 * line; i++) {
    int i = 2;                      // ERROR: overlapping scope
    System.out.print("/");
}
i = 4;                              // ERROR: outside scope
```

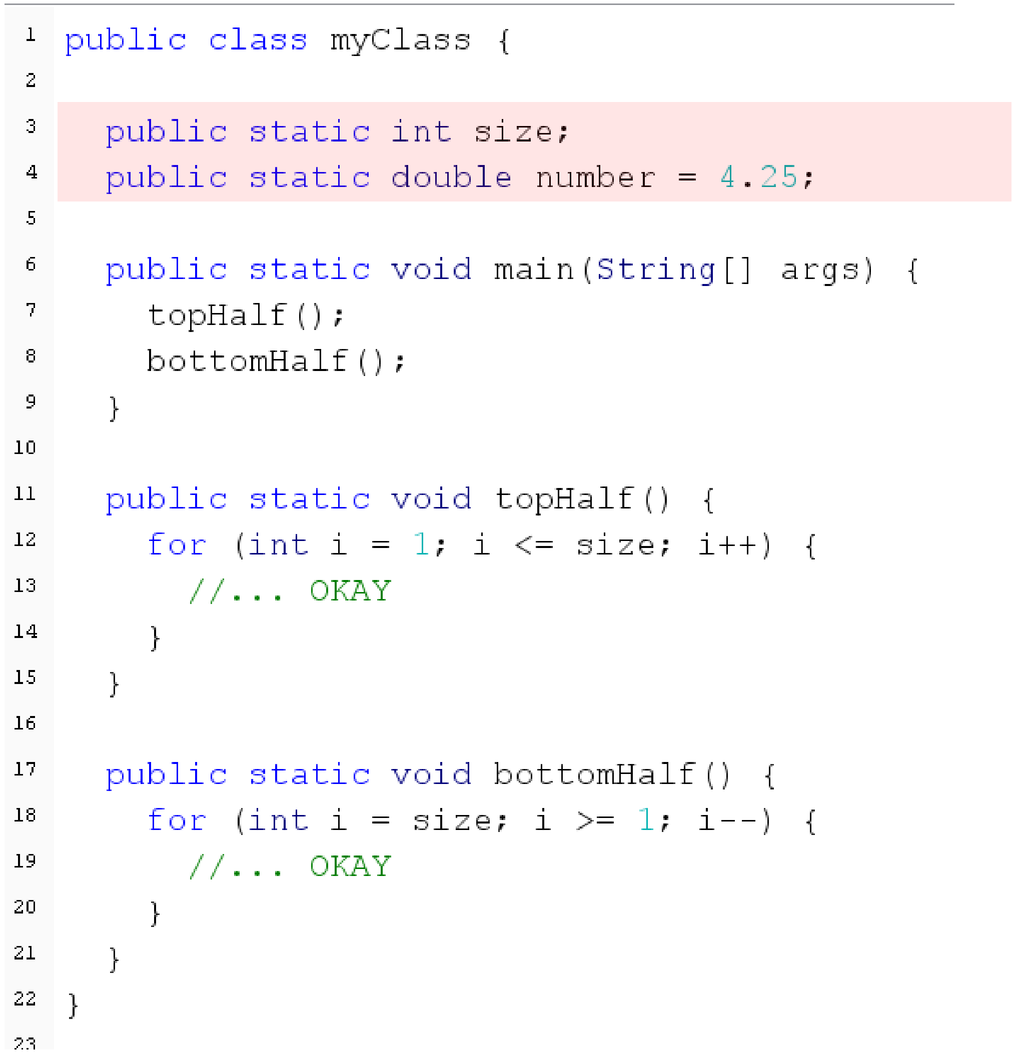
Global Variables

Global variable: A variable visible to the whole program

But, global variables are considered bad style.

It is difficult for programmers to find errors if every method in the class can change the variable!

Global variables = Lose style points!



```java
public class myClass {

  public static int size;
  public static double number = 4.25;

  public static void main(String[] args) {
    topHalf();
    bottomHalf();
  }

  public static void topHalf() {
    for (int i = 1; i <= size; i++) {
      //... OKAY
    }
  }

  public static void bottomHalf() {
    for (int i = size; i >= 1; i--) {
      //... OKAY
    }
  }
}
```

Class constants

- **class constant**: A fixed value visible to the whole program.
  - value can be set only at declaration; cannot be reassigned, hence the name: *constant*

- Syntax:

  `public static final` **type name** `=` **expression**`;`

  - name is usually in ALL_UPPER_CASE

  - Examples:

```
public static final int HOURS_IN_WEEK = 7 * 24;
public static final double INTEREST_RATE = 3.5;
public static final int SSN = 658234569;
```

Adding a constant

```
public class Sign {
    public static final int HEIGHT = 5;

    public static void main(String[] args) {
        drawLine();
        drawBody();
        drawLine();
    }

    public static void drawLine() {
        System.out.print("+");
        for (int i = 1; i <= HEIGHT * 2; i++) {
            System.out.print("/\\");
        }
        System.out.println("+");
    }

    public static void drawBody() {
        for (int line = 1; line <= HEIGHT; line++) {
            System.out.print("|");
            for (int spaces = 1; spaces <= HEIGHT * 4; spaces++) {
                System.out.print(" ");
            }
            System.out.println("|");
        }
    }
}
```

Using a constant

- Constant allows many methods to refer to same value:

```
public static final int SIZE = 4;

public static void main(String[] args) {
    topHalf();
    bottomHalf();
}

public static void topHalf() {
    for (int i = 1; i <= SIZE; i++) {     // OK
        ...
    }
}

public static void bottomHalf() {
    for (int i = SIZE; i >= 1; i--) {     // OK
        ...
    }
}
```

Repetitive figure code

```
public class Sign {

    public static void main(String[] args) {
        drawLine();
        drawBody();
        drawLine();
    }

    public static void drawLine() {
        System.out.print("+");
        for (int i = 1; i <= 10; i++) {
            System.out.print("/\\");
        }
        System.out.println("+");
    }

    public static void drawBody() {
        for (int line = 1; line <= 5; line++) {
            System.out.print("|");
            for (int spaces = 1; spaces <= 20; spaces++) {
                System.out.print(" ");
            }
            System.out.println("|");
        }
    }
}
```

Magic Numbers