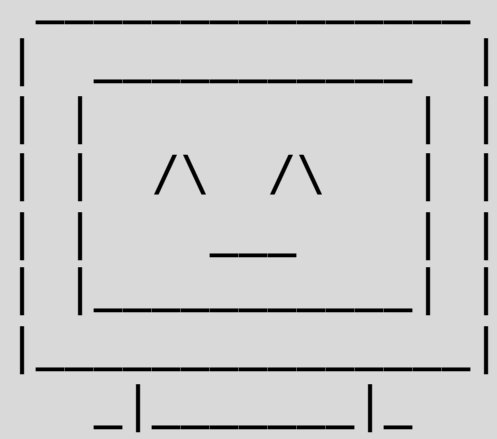
Static Methods
Parameters & Return Values

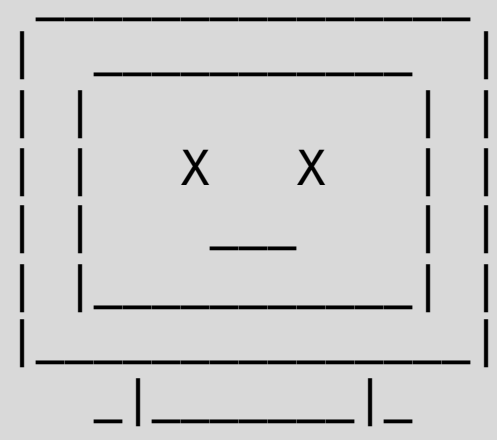
zyBook Chap 3.1, 3.2, 3.3, 3.4

```java
public class PrintFace {
    public static void main(String[] args) {
        System.out.println("  _________________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |   /\\  /\\    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");

        System.out.println("  _________________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |    X   X    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");
    }
}
```

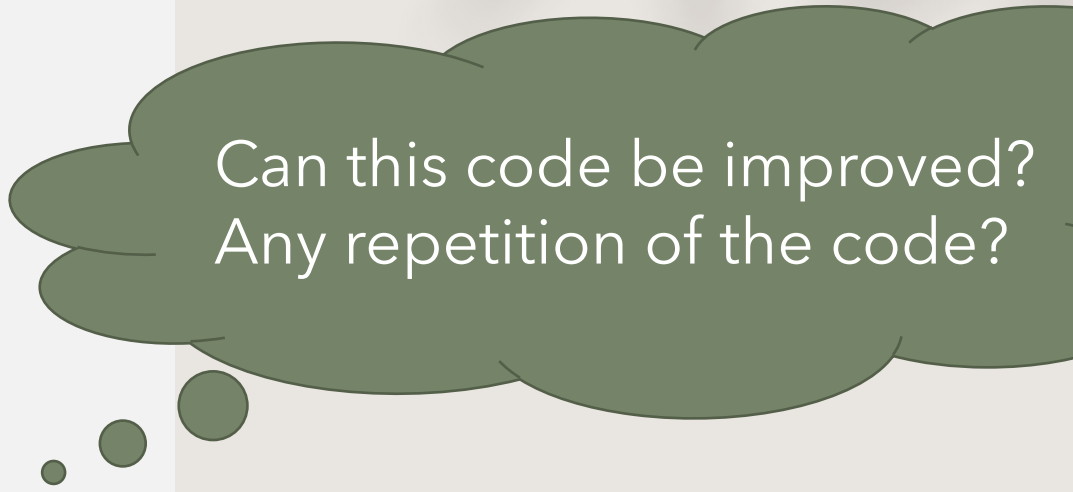
```
$ javac PrintFace.java
$ java PrintFace
```





```
public class PrintFace {
    public static void main(String[] args) {
        System.out.println("  _________________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |   /\\   /\\    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");

        System.out.println("  _________________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |    X   X    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");
    }
}
```



```
public class PrintFace {
    public static void main(String[] args) {
        System.out.println("  _______________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |   /\\   /\\    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");

        System.out.println("  _______________ ");
        System.out.println(" |  _____________  |");
        System.out.println(" | |             | |");
        System.out.println(" | |    X   X    | |");
        System.out.println(" | |     ___     | |");
        System.out.println(" | |_____________| |");
        System.out.println(" |_________________|");
        System.out.println("    _|_______|_\n");
    }
}
```

// Header
// Happy eyes
// Footer

// Header
// Unhappy eyes
// Footer

Can this code be improved?
Any repetition of the code?

Methods

A group of statements with a given name

- Decompose a program into smaller modules that
  - Each module implements a part of the program behavior, and
  - Can be implemented and tested separately
- Eliminates redundancy by allowing code reuse

Methods

Equivalent Implementations

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Have a great day!");
    }
}
```

```
public class HelloWorld {

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

    public static void printMessage() {
        System.out.println("Hello World!");
        System.out.println("Have a great day!");
    }
}
```

Step 2: Call the method
Inside of the main method

<return type> <methodName> (<parameter(s)>)

Step 1: Declare the method
Inside of the class, outside of the main method

Return Type and Return Statement

The **type** of the **output** generated by the method, if any

- Could be an int, a double, a char, a boolean, or a String, ...
- A method can generate only **ONE** value, that is, return one value
- If a method does not generate a value, the return type is **void**
  - E.g., contains print statements only

```
// No parameters; Has a return value
public static <type> <methodName>() {
    <statements>;
    return <expression>;
}

// No parameters; Has no return value
public static void <methodName>() {
    <statements>;
}
```

Parameters of a Method

The **input** into the method

- Each method can have 0, 1, or many parameters.
- Each parameter has a **type** and **name** (similar to variables).
- The **scope** of parameters is the method.

```
// Two parameters; Has a return value
public static <type> <methodName>(<type> <paraName>, <type> <paraName>) {
    <statements>;
    return <expression>;
}
```

Flow of Control

- Flow of Control is the order that statements execute.
- With methods:
  - Control is transferred to the called method
  - When the called method is complete, the control returns to the calling method

```java
public class FlowOfControlDemo {
    public static void main(String[] args) {
        methodOne();
    }

    public static void methodOne() {
        System.out.println("A");
        System.out.println("B");
        methodTwo();
        System.out.println("C");
    }

    public static void methodTwo() {
        System.out.println("X");
        System.out.println("Y");
        System.out.println("Z");
    }
}
```

Output:

A

B

X

Y

Z

C

Q: What's the exact output of the following code?

```
public class MethodExample {

    public static void main(String[] args) {
        m1(4);
        int x = m2(2, 4);
        System.out.println("x is " + x + ".");
    }

    public static void m1(int x) {
        System.out.println("m1 prints its parameter " + x + ".");
    }

    public static int m2(int a, int b) {
        System.out.println("m2 is called.");
        return a + b;
    }
}
```

Since m2 generates an integer result, we declare an integer x to hold its return value

```
$ javac MethodExample.java
$ java MethodExample
m1 prints its parameter 4.
m2 is called.
x is 6.
```

Javadoc a Method

```
public class MethodExample {

    public static void main(String[] args){
        m1(4);
        int x = m2(2, 4);
        System.out.println("x is " + x + ".");
    }

    /**
     * This method takes one parameter and prints it out.
     * @param  x  a value to be printed
     */
    public static void m1(int x) {
        System.out.println("m1 prints its parameter " + x);
    }

    /**
     * This method adds up its two parameters
     * @param  a  the first value to be added
     * @param  b  the second value to be added
     * @return  the sum of a and b
     */
    public static int m2(int a, int b) {
        System.out.println("m2 is called.");
        return a + b;
    }
}
```

- Description of method
- If it takes parameters, use one **@param** tag for **each** parameter. List parameter **name** followed by the **description** of the parameter.
- If it returns a value, use **@return** tag followed by the description of the return value.