Methods

Also called functions, subroutines.

Methods

We've encountered all kinds of methods already such as: print, println, printf, ceil, floor, sqrt and more. A method is a way to organize code and make it reusable.

If you find yourself copying / pasting lots of code, you probably have a good candidate for code to put into a method.

Methods

Many of the methods we've used return a value, however if your method does not return a value, you use the keyword void.

```
while (done == false) {
    displayMenu();
    // Do some other stuff
}


void displayMenu() {
    System.out.println("Add a new item");
    System.out.println("Update an item");
    System.out.println("Delete an item");
}
```

Methods

Here our method uses a parameter and returns a value.

```
System.out.println("Circle radius: ");
double radius = scan.nextDouble();

double area = areaOfCircle(radius);
```

```
double areaOfCircle(double radius) {
    return Math.PI * Math.pow(radius, 2.0);
}
```

Methods

Below is a function that converts Fahrenheit to Celsius.

```
System.out.println("Temperature: ");
double f = scan.nextDouble();

double c = convert(f);
```

```
double convert(double f) {
    return ((f - 32.0)*5.0)/9.0;
}
```

Methods

```
// Before
Random rand = new Random();

int st = rand.nextInt(6) + 1 + rand.nextInt(6) + 1 + rand.nextInt(6) + 1;
int dx = rand.nextInt(6) + 1 + rand.nextInt(6) + 1 + rand.nextInt(6) + 1;
int iq = rand.nextInt(6) + 1 + rand.nextInt(6) + 1 + rand.nextInt(6) + 1;
int wi = rand.nextInt(6) + 1 + rand.nextInt(6) + 1 + rand.nextInt(6) + 1;
```

Methods

```
// After
Random rand = new Random();

int st = diceRoll(rand);
int dx = diceRoll(rand);
int iq = diceRoll(rand);
int wi = diceRoll(rand);



int diceRoll(Random rand) {
    return rand.nextInt(6) + 1 + rand.nextInt(6) + 1 + rand.nextInt(6) + 1;
}
```

Methods

```
// Before
Scanner scan = new Scanner(System.in);

System.out.println("What is your name?");
String name = scan.nextLine();

System.out.println("What is your quest?");
String quest = scan.nextLine();

System.out.println("What is your favorite color?");
String color = scan.nextLine();
```

Methods

```
// After
Scanner scan = new Scanner(System.in);

String name = getString(scan, "What is your name?");
String quest = getString(scan, "What is your quest?");
string color = getString(scan, "What is your favorite color?");




String getString(Scanner scan, String prompt) {
    System.out.println(prompt);
    return scan.nextLine();
}
```

Methods

```
// Before
Scanner scan = new Scanner(System.in);
String name = "", quest = "", color = "";

while (name.length() == 0) {
    System.out.println("What is your name?");
    name = scan.nextLine();
}

while (quest.length() == 0) {
    System.out.println("What is your quest?");
    quest = scan.nextLine();
}

while (color.length() == 0) {
    System.out.println("What is your favorite color?");
    color = scan.nextLine();
}
```

Methods

```
// After
Scanner scan = new Scanner(System.in);

String name = getString(scan, "What is your name?");
String quest = getString(scan, "What is your quest?");
string color = getString(scan, "What is your favorite color?");


String getString(Scanner scan, String prompt) {
    String result = "";

    while (result.length() == 0) {
        System.out.println(prompt);
        result = scan.nextLine();
    }

    return result;
}
```

Methods

When adding methods to our program, for now we need to add public static.

```
import java.util.Scanner;

public class RandomExample
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);

        displayMenu();
        String name = getName(scan);
        displayName(name);
    }

    public static void displayMenu() {
        System.out.println("Welcome to my awesome program!");
        System.out.println("So great to have you here!");
    }

    public static String getName(Scanner scan) {
        System.out.println("What is your name?");
        return scan.nextLine();
    }

    public static void displayName(String name) {
        System.out.println("Hello " + name + ", so nice to meet you!");
    }

}
```

Let’s Code

Don’t Forget!

Check the syllabus / schedule for reading assignments and due dates!

Command-line Arguments

(why can’t we all just get along)

args

You may have noticed all of our programs start with the following line:

```
public static void main(String[] args)
```

`String[] args`

args is short for "arguments"

(notice its an **array** of Strings)

args

If you have run something from the command-line before, you have probably used command-line arguments.

```
cd temp
```

```
args[0] = "temp"
```

```
program.exe in.txt out.txt
```

```
args[0] = "in.txt"
args[1] = "out.txt"
```

```
shutdown -r now
```

```
args[0] = "-r"
args[1] = "now"
```

args

You can use args[ ] just like any other array of strings. Our programs are not executable files (.exe or applications). The compiler turns our code into bytecode which is executed by the JRE (java runtime environment).

```
public static void main(String[] args) {
    System.out.println("Hello, " + args[0] + "!");
}
```

```
java MyProgram Billy
```

```
Hello, Billy!
```

args

Always make sure the user provided the arguments you need!

```
public static void main(String[] args) {
    if (args.length == 0) {
        System.out.println("usage: MyProgram name");
    } else {
        System.out.println("Hello, " + args[0] + "!");
    }
}
```

```
java MyProgram
```

```
usage: MyProgram name
```

Let’s Code

Don’t Forget!

Check the syllabus / schedule for reading assignments and due dates!