CS 302: INTRODUCTION TO PROGRAMMING IN JAVA

Chapter 5: Methods

Lecture 10

PROBLEM

- What if I was using a lot of different arrays and often wanted to print out their contents?
  - I would have to have that same for loop every time I wanted to print out the array contents
- Solution: Use Methods!

THE UNIVERSITY of WISCONSIN MADISON

METHODS

- A method is a named sequence of instructions
- We have already seen many methods
  - main
  - Math.pow()
  - in.next()
  - System.out.println()
  - etc.

Method Man loves methods

METHOD CALLING CONVENTIONS

- [identifier].[methodName]([arguments])
- System.out.println("Hello");
- Math.pow(2, 4);
- System.out.println(x);
- Multiple arguments always seperated by commas (,)
- Identifiers can be class or variable names
  - Variable Names – if calling on a reference variable (like Strings)
  - Class Names – if calling a static method

METHOD BASICS

- Basic Idea – break code down into simple building blocks
- Black Box – when using methods other people have written, we don't need to know how they work
  - TV example
  - Math.pow
- Methods take in input (arguments) and give back output (return values)

THE UNIVERSITY of WISCONSIN MADISON

CALLING METHODS

- Arguments = input
  - Can have 0 or many arguments
  - Go inside the parenthesis
  - Ex.: Math.pow(**2, 4) or rand.nextInt()**
  - Defined by method parameters
- Return Value
  - Can only have 0 or 1 return value
  - Ex. Math.pow can only return 1 value, to do additional powers must call Math.pow multiple times with different parameters
  - Specified by the "return" keyword

THE UNIVERSITY of WISCONSIN MADISON

LUNCH EXAMPLE

```
class Lunch
{
   main
   {
      sandwich = makeSandwich(bread, peanutButter, jelly);
      drink = pourDrink(cup, water);
      turnOnTV();
      eatLunch(sandwhich, drink);
    }
}
```

MATH.POW

```
/**
  Computes a base raised to a positive power
  @param base the base
  @param power the power to raise the base to, must be >= 0
  @return base^power, or 1 if base < 0
*/
public static double pow(double base, double power) {
    double sum = 1;
    for (int i = 1; i <= power; i++) {
        sum *= base;
    }
    return sum;
}
```

IMPLEMENTING METHODS

- Methods go outside the main method but inside the class definition
- Methods have:
  - Method Header
    - public static modifiers like the main method (these will be explained eventually)
    - Return type
    - Name – camelCase convention like variables
    - Type and name for each parameter variable
  - Method Body (code to execute)
- Lets make our own Math.pow method header:

public static double myPow(double base, double exponent)

THE UNIVERSITY of WISCONSIN MADISON

METHOD BODY

- The body of the method is defined by braces and that is the code that will execute when the method is called

```
public static double myPow(double base, double exponent)
{
   //method body
   //must return a double at the end of the method
}
```

THE UNIVERSITY of WISCONSIN MADISON

METHOD COMMENTS

- Use javadoc comments /**...*/ above the method header

```
/**
  Computes a base raised to a positive power
  @param base the base
  @param power the power to raise the base to, must be >= 0
  @return base^power, or 1 if base < 0
*/
public static double myPow(double base, double exponent)
```

THE UNIVERSITY of WISCONSIN MADISON

COMMON ERRORS

```
public static int myDivide(a, b)
{
   return a/b;
}
```

- Forgot the parameter variable types

```
public static myMultiply(int a, int
   b)
{
   return a*b;
}
```

- Forgot the method return type

COMMON ERRORS

```
public static int rectArea(double length, double width)
{
   return length*width;
}
```

Return type doesn't match what the method body returns

PRACTICE 1

- Write a method to compute the area of a cube or a rectangular block
- Input: (validate input type)
  - Height
  - Width
  - Depth
- Output:
  - Volume = height * width * depth

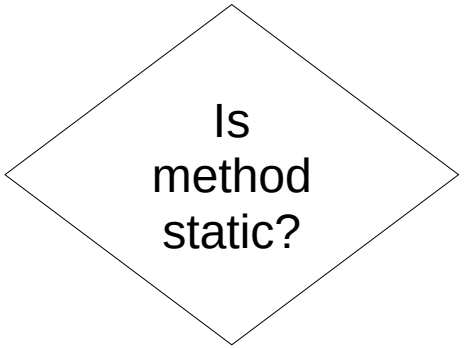
THE UNIVERSITY of WISCONSIN MADISON

WHY DO WE USE METHODS?

- Increase modularity
- Increase readability / maintainability
- Reduce redundancy
- Ex. P1:
  - Multiple Validation loops -> a single validation method
  - Multiple modes -> multiple methods (better style, easier debugging, etc.)

THE UNIVERSITY of WISCONSIN MADISON

CALLING METHODS



Yes

ClassName.methodName(arguments)

ex. Math.pow(2, 4);

No

objectName.method(arguments)

ex. in.nextLine() or rand.nextInt(10)

- For now, we will only create static methods
- If calling static methods that are defined within the same class that they are being called from, the ClassName. identifier can be dropped from the method call (just call the method using its name)

THE UNIVERSITY of WISCONSIN MADISON

ARGUMENTS VS PARAMETERS

- Arguments (book calls them "parameter values") get passed to the method
- Parameters (book calls them "parameter variables") are defined in the method header
- Arguments must match the parameter definitions in type, order, and number
  - Do not need to have the same name
- Argument values get COPIED into the parameter variables
  - Changing the parameter does **NOT change the original argument**

THE UNIVERSITY of WISCONSIN MADISON

ARGS VS PARAMETERS EXAMPLE

```
int a = 4, b = 5;

int area = rectArea(a, b);

...

public static int rectArea(int width, int height)

{

...

}
```

Blue = arguments
Red = parameters

Arguments must match parameters in **number**, **order**, and **type**

- a's value is copied to width
- b's value is copied to height

WHAT IS THE VALUE OF X?

```
int x = 4;
double y = doSomething(x);
...
public static double doSomething(int z)
{
   for (int i = 0; i < 3; i++)
                z = z + i;
   return z;
}
```

VARIABLE SCOPE

- Just like what we talked about in ifs and loops
  - A variable declared within braces is ONLY valid within those braces
  - That means you can't use variables defined in a method outside of that method!!!
- Can use the same variable name in different scopes

Ex.

```
public static double rectArea(int length, int width)

public static double cubeVolumn(int lenght, int width, int depth)
```

THE UNIVERSITY of WISCONSIN MADISON

RETURN STATEMENT

- **Immediatly exits the method**
- Can return
  - Literal – return 4;
  - Variable – return x;
  - Result of an expression – return (x && y || (3+z < 5));
  - Result of another method call – return doSomething(x);
- Return type must match the type in the method header
- If the method returns nothing, it is of type void

THE UNIVERSITY of WISCONSIN MADISON

VOID METHODS

- Ex. public static void main(String[] args)
- Used when the method doesn't return anything
- Often used for displaying things
- Can still use the return statement to exit the method immediatly
  - In this case the statement is simply: return;

```
printStars(3, 4);

...

public static void printStars(int
    width, int height)
{
   for (int i = 0; i < width; i++)
      for (int j = 0; j < width; j ++)
         print("*");
      print("\n");
}
```

THE UNIVERSITY of WISCONSIN MADISON

RETURN VS. BREAK

```
for (int i = 0; i < 10; i++)
{
   if (i == 5) break;
}
```

Vs

```
for (int i = 0; i < 10; i++)
{
   if (i == 5) return;
}
```

- Break simply breaks out of the current loop
  - What would happen in a nested loop?
- Return immediatly exits the method and returns the return value (if any)
  - What would happen in a nested loop?

THE UNIVERSITY of WISCONSIN MADISON

RETURN WITHIN CONDITIONALS

```
public static String getDay(int day)
{
   if (day == 1) return "Sunday";
   if (day == 2) return "Monday";
   if (day == 3) return "Tuesday";
   ...
}
```

- Note we don't need else ifs because the return statement exits the method immediately!
- If we do branch every possible traversal must have a return statement!

Practice 2 (take home)

- Remember we had a practice (refer to HopeAndChange.java on the course website under “In-Class Example Code” tab)

- Let’s do the same thing but now we use a static method to calculate change given coin value (25 for quarter, 10 for dime, 5 for nickel, 1 for penny), coin name (“quarter”, “dime”, “nickel”, “penny”), change (centsLeft defined previously). So the parameter values for this method should be coinValue, coinName, centsLeft with appropriate data types, respectively.

- The method should be able to print out number of coins (quarters, dimes, nickels or pence) AND return the remainder (centsLeft).

- In the main method, call the method for the number of quarters, dimes, nickels, pennies.

THE UNIVERSITY of WISCONSIN MADISON