**What is a method?**

We already use this term for:

* The methods we call from Java API classes
* Where we put our code (*main* method)

A method is code that

We can now create our own!

**Why create methods in our code?**

Allows us to use *divide-and-conquer* to solve problems

So far we’ve used code like this:

public static void main(String[ ] args)

{

System.out.println(“This program determines your name’s length”);

String name = JOptionPane.showInputDialog(null, “What is your name?”);

System.out.println(“Your name has length “ + name.length());

}

Now we can *define* additional methods that we *call* from main:

public class NameLength

{

public static void main(String[ ] args)

{

String name = JOptionPane.showInputDialog(null, “What is your name?”);

System.out.println(“Your name has length “ + name.length());

}

public static void announce( )

{

JOptionPane.showMessageDialog(null, “This program determines your name’s length”);

}

}

***Method Type 1***

You just used a method

* Performs a task and then ends
* Examples:

System.out.println(“basic”);

JOptionPane.showMessageDialog(“Hello there!”);

**Example:** Lab 0 – open the example from ClassExamples

public class AndrewMaggieLab0  
{  
 public static void main(String[] args)  
 {  
 System.out.println("This is Maggie.");  
 System.out.println("I am a junior physics major.");  
 System.out.println("I am on the badminton team.");  
 System.out.println("I am from Baltimore, Maryland.");  
 System.out.println("I can play the piano.");

System.out.println("This is Andrew.");  
 System.out.println("I am a sophomore majoring in math and econ.");  
 System.out.println("I am in the jazz combo and the Greysounds.");  
 System.out.println("I am from Ellicott City, Maryland.");  
 System.out.println("I have a sister who works at Pixar.");  
 }  
}

What tasks is this code performing?

New methods?

**Passing Arguments**

What if we want to give a method access to information that is in another method?

*Scope*

* Local variables:
  + Defined in
  + Exist in
* Why I need to pass arguments:
* Parameter values have scope of
* The arguments sent when calling a method CAN’T be changed by the code in the method *unless they are objects (more on this later)*

*Terminology*

* Arguments: values sent to a method
* Parameters: variable that will hold that value

*Example*

public class Minimum

{

public static void main(String[] args)

{

double a = 10.3;

double b = 20.1;

minimum(a,b);

}

public static voidminimum(double x, double y)

{

if(x < y)

System.out.print(“The smallest value is “ + x);

else

System.out.print(“The smallest value is “ + y);

}

}

**Method Type 2**

Another option is a method

* Performs a task, then sends a value back to the code that called it and ends
* Example:

int number = Integer.parseInt(input);

double answer = Math.sqrt(number);

Why:

How:

* Modify the return type in the header
* end the method body with the keyword **return** followed by the value to return

public static **int** increaseByTwo(int value)

{

int result = value \* 2;

**return** result;

}

*Example*

public static void main(String[] args)

{

double a = 10.3;

double b = 20.1;

}

public static minimum(double x, double y)

{

if(x < y)

else

}

What does this code do?

Why is this useful?

**Example – File Reading Programs**

Previously we had everything in the main method. Instead, we could put user input in the main method, but have the file reading/processing in a different method.

public class ActressAge

{

public static void main(String[] args)

{

//open file for input reading, and exit if it doesn't exit

File inputFile = new File (“BestActress.txt”);

if(!inputFile.exists())

{

JOptionPane.showMessageDialog(null,"Sorry, file doesn't exist.” +

“ Ending program.");

System.exit(1); //use because we can't recover from the error

}

//read from file

Scanner scan = new Scanner (inputFile);

int sum = 0;

int count = 0;

//read each line, adding the ages to a sum

while (scan.hasNext())

{

//read in everything on this one line

short year = scan.nextShort();

String name = scan.next(); //NOT nextLine!

String title = scan.next();

short age = scan.nextShort();

String nationality = scan.next();

//add age to sum

sum += age;

count++;

}

//find average & output to the user with JOptionPane

double average = (double) sum / count;

JOptionPane.showMessageDialog(null,"The average age of winners is " + average);

}

}

public class ActressAge

{

public static void main(String[] args)

{

//open file for input reading, and exit if it doesn't exit

File inputFile = new File (“BestActress.txt”);

if(!inputFile.exists())

{

JOptionPane.showMessageDialog(null,"Sorry, file doesn't exist.” +

“ Ending program.");

System.exit(1); //use because we can't recover from the error

}

//read from file

Scanner scan = new Scanner (inputFile);

JOptionPane.showMessageDialog(null,"The average age of winners is " +

}

{

int sum = 0;

int count = 0;

//read each line, adding the ages to a sum

while (scan.hasNext())

{

//read in everything on this one line

short year = scan.nextShort();

String name = scan.next(); //NOT nextLine!

String title = scan.next();

short age = scan.nextShort();

String nationality = scan.next();

//add age to sum

sum += age;

count++;

}

//find average & output to the user with JOptionPane

double average = (double) sum / count;

}

}

**Summary**

* A method lets you organize code into separate sections
* A method should accomplish 1 goal. If it needs to accomplish more than that, you need more
* You can give a method information (arguments)
* You can ask a method for information (return value)
* You first have to *define* the method, then you can *call* it from a different method