

Pro Dev Session



Pro Dev Session

You may want to write this down.

Agile Software Development is a software development strategy that involves building an application in short iterative bursts. There are several different schools of agile methodology and every team practices agile software development in a slightly different way.

In this class, we'll focus on the general rules that span all agile practices. This morning will practice some core components of the agile methodology, and going forward we'll integrate these practices into every class.

Pro Dev Session

You may want to write this down.

Sprint: a set unit of time in which a team will accomplish a given set of tasks. Usually about 2 weeks.

Agile Ceremonies

- **Sprint Planning :** Where features are selected and broken into tasks for the current sprint.
- **Stand-up:** Where teammates report in daily.
- **Retrospective:** Where teammates work together to provide honest feedback and aid in team growth.

Pro Dev Session

You may want to write this down.

Why are we doing this?

- Agile is a key part of almost every software development team these days. Learning to work in an agile fashion in class will help you hit the ground running on the first day of your job.

Agile Practice

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in GROUPS OF 4 to complete all of the goals below. For this activity, you are working to “build” a social media application. You’ll eventually want to rival the major players so you’ll probably need a chat, messaging, events, posts, pictures, a way to like posts, games, stories, a fundraising feature, photo filters, and more! Instead of days, we’ll be working in 2 minute increments!

Goals:

- Hold a sprint planning meeting where you will choose a feature or 3 and break them into tasks. Teammates should choose tasks they want to implement (draw) (2 min)
- Start drawing! (2 min)
- Hold a scrum. Each member should say what they did in the last 5 min, what they plan to do in the next 5 min, and any blockers they have. (4 min)
- Start drawing! (2 min)
- Hold a retrospective. Did you accomplish all your goals? What went well? What could have gone better? What will you change for next time? (5 min)



**15
minutes!**

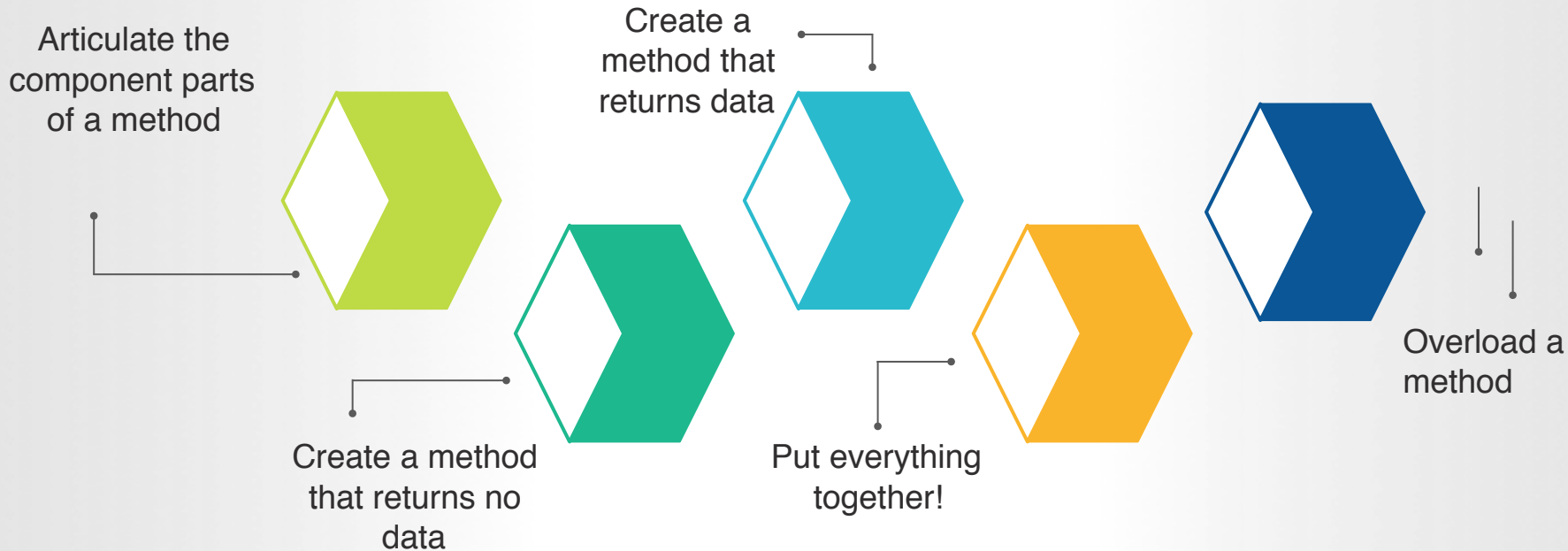
Stand Up!



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

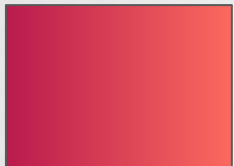


Let's Do This!



A Quick Reminder

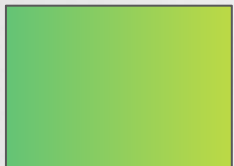
COLOR SYSTEM



Red means stop. When you see a red header, close your laptop.



Yellow means wait. Open your laptop and code along with me.



Green means go. This is independent practice time so get to coding!

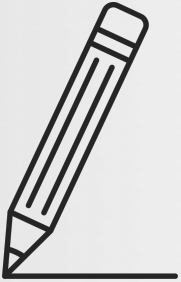


Method Fundamentals



Methods

THE VERBS OF THE CODING WORLD



Notebooks Ready? It's time for a mini lecture.



Methods

THE VERBS OF THE CODING WORLD

So what is a class and what in the heck does `public static void main(String[] args)` mean?

Let's zoom out a little. Right now we are building tiny little programs, but soon we'll be building large applications as part of a team. We'll need some organization system to keep our millions of lines of code straight.

A method is a block of code that performs a single action. The main method is the method that runs automatically when you start your application and calls any other methods you may need to call.

A class is a container (kind of like a folder) that holds a group of related methods.



Methods

THE VERBS OF THE CODING WORLD

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

```
public static void printName(){  
    System.out.println("My name is Pat");  
}
```



Methods

THE VERBS OF THE CODING WORLD

So why use methods?

A lot of reasons. Most notably, methods help to organize our code, help make our code more maintainable, and help us write less code.

How? Let's see any example...



Methods

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(String args[]){  
    System.out.println("My name is Pat");  
    System.out.println("Their name is Patsy");  
    System.out.println("Her name is Pam");  
    System.out.println("My name is Pat");  
    System.out.println("Their name is Patsy");  
    System.out.println("Her name is Pam");  
    System.out.println("My name is Pat");  
    System.out.println("Their name is Patsy");  
    System.out.println("Her name is Pam");  
    System.out.println("My name is Pat");  
    System.out.println("Their name is Patsy");  
    System.out.println("Her name is Pam");  
}
```

Methods

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(String args[]){  
    printNames();  
    printNames();  
    printNames();  
    printNames();  
}  
  
public static void printName(){  
    System.out.println("My name is Pat");  
    System.out.println("Their name is Patsy");  
    System.out.println("Her name is Pam");  
}
```


Methods

THE VERBS OF THE CODING WORLD


Why would I need to print Pat, Patsy, Pam 4 times?

You wouldn't. This was a silly example.

Let's go back to our social media example. A common action is getting a list of your friends. That involves around 5-10 lines of code. How often do you need to do it?

Everytime you click on your friend list, or create an event (because you need a list of people to invite), or go to the home (it needs to know whose posts to show, or click chat (because you need a list of people to message), or any other of a dozen scenarios.

Writing that 5-10 lines over and over is much longer than just calling the getFriends method in a dozen places.



Methods

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Follow along with me as I complete this activity.

Goals:

- Create a method that prints all the numbers from 1 to 5 using a loop.

Methods

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work with your neighbor to INDIVIDUALLY complete all of the goals below.

Goals:

- Create a new project.
- Create a method that prints all the numbers from 5 to 10 using a loop

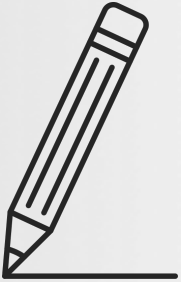
Note: We know you don't know squat about methods yet. We're going to keep lecturing, I promise. This is just a quick pulse check.



5 minutes!

Methods

THE VERBS OF THE CODING WORLD



Notebooks Ready? It's time for a mini lecture.




Methods

COMPONENT PARTS

```
public static void doSomething(a,b,c) {  
    // Code that does something  
}
```

Access Modifier: Can be public, private, or protected. This controls which parts of your application have access to this method. Just use public for now.



Methods

COMPONENT PARTS

```
public static void doSomething(a,b,c) {  
    // Code that does something  
}
```

Static Keyword: We'll get into this next unit. It is not always required but is for the activities this week.



Methods

COMPONENT PARTS

```
public static void doSomething(a,b,c) {  
    // Code that does something  
}
```

Return Type: Methods are like a box. Data can only be passed in through parameters and can only escape the curly braces through the return keyword.



Methods

COMPONENT PARTS

void means nothing is returned

```
public static void print() {  
    System.out.println("I print");  
}
```


Methods

COMPONENT PARTS

int means an integer is returned

```
public static int makeFour() {  
    return 4;  
}
```



Methods

COMPONENT PARTS

String means a String is returned

```
public static String getName() {  
    return "My name is Sammie";  
}
```

Methods

COMPONENT PARTS

... you get the picture.



Methods

COMPONENT PARTS

```
public static void doSomething(a,b,c) {  
    // Code that does something  
}
```

Method Name: The method name is how you call the method when you want it to run.



Methods

COMPONENT PARTS

```
public static void doSomething(a,b,c) {  
    // Code that does something  
}
```

Parameters: Parameters are how we get data into a function.

Let's put it all together...



Methods

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(String args[]) {  
    int sum1 = add(1, 2);  
    int sum2 = add(3, 7); ←The call-site is replaced by the returned value  
  
    System.out.println(sum1);  
    System.out.println(sum2);  
}  
  
public static int add(int a, int b) {  
    return a + b;  
}
```



Check-in Time

- What is the method name used for?
- What are parameters?
- What is a return type?
- What does the return keyword do?



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

Articulate the
component parts
of a method





Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in 10 minutes.



Lab Time

Methods

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

Time for a few more examples to make sure we
are all on the same page.



LabTime

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work together but INDEPENDENTLY write your own code to complete all of the goals below.

Goals:

- Complete all the katas listed in the activity file.

If this is tough, great! You're getting practice.

If this is easy, great! Help your buddies.

We'll be circulating to provide individual help where it's needed.

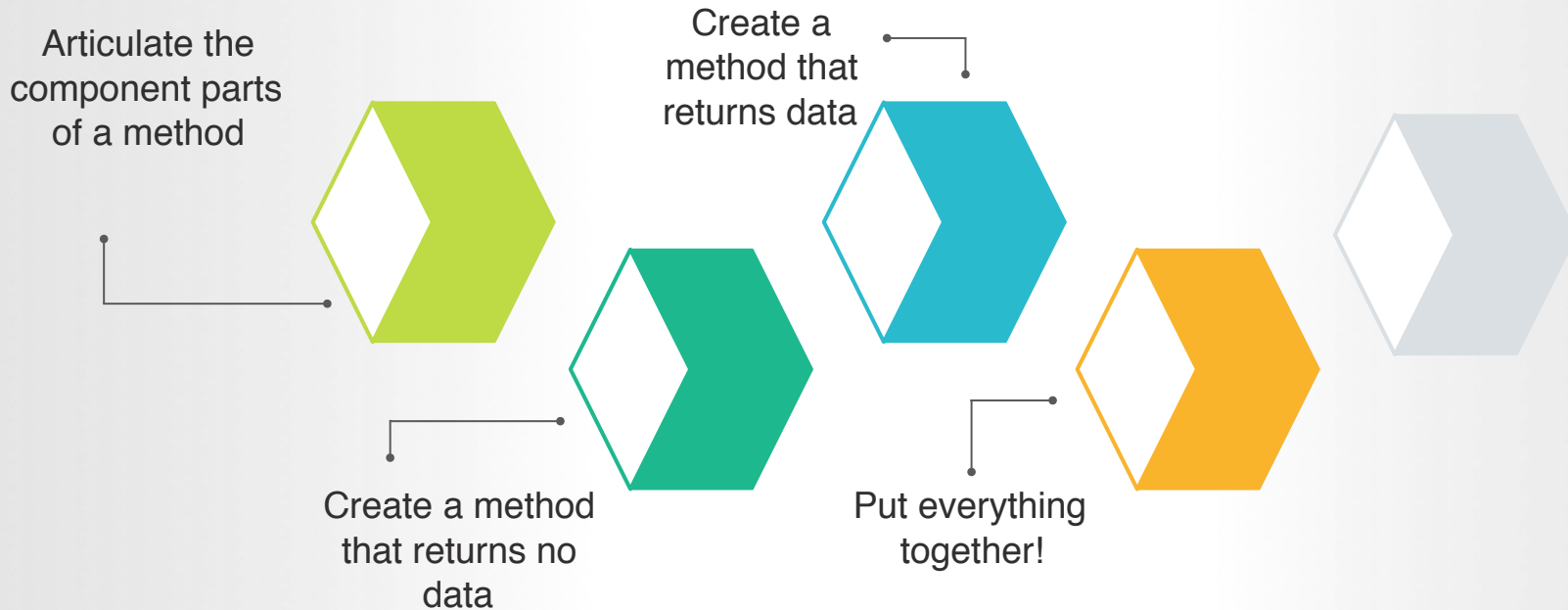


**50
minutes!**

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



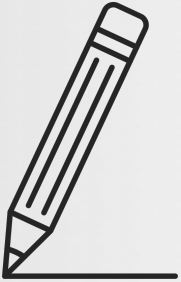
lunch.

Method Overloading



Overloading

SO MANY METHODS, SO LITTLE TIME



Notebooks Ready? It's time for a mini lecture.



Overloading

SO MANY METHODS, SO LITTLE TIME

Method Signature: The name and parameter types and count for a given method. These in combination uniquely identify a method, so that when a method is called, the compiler knows which block of code to execute.

```
public static int add(int a, int b) {  
    return a + b;  
}
```

```
public static void add(int a, int b) {  
    System.out.print(a + b);  
}
```



This is illegal!

Overloading

SO MANY METHODS, SO LITTLE TIME

Method Signature: The name and parameter types and count for a given method. These in combination uniquely identify a method, so that when a method is called, the compiler knows which block of code to execute.

```
public static int add(int a, int b) {  
    return a + b;  
}
```

```
public static int add(double a, int b) {  
    return a + b;  
}
```



A-Okay! Overloading!

Overloading

SO MANY METHODS, SO LITTLE TIME

Method Overloading: Multiple methods in a single class having the same name but different parameter lists.

Why would you do this?

- Sometimes you need methods that do virtually the same thing, but with different inputs.
- Overloading allows you to build more robust classes. Imagine how annoying it would be if you could only get the maximum of two integers but not two doubles!





Check-in Time

- How many add methods would you need to write to cover all pairs of numbers that you might want to add?
- Why can't you have two methods with the same name and parameter list?



Overloading

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work together but INDEPENDENTLY write your own code to complete all of the goals below.

Goals:

- Write a method called subtract that takes 2 integer parameters and a method called subtract that takes 3 integer parameters.
- Write a method called validateInput that takes a String and validates whether the String has more than eight characters (return a boolean). Then write a method called validateInput that validates that an integer is exactly 10 digits.



**10
minutes!**



Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in 10 minutes.

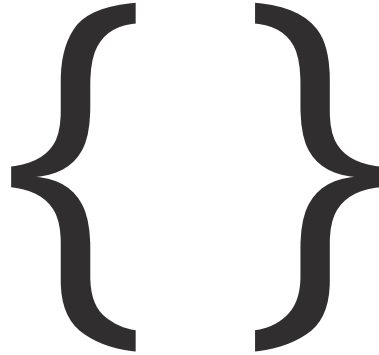


Scope, Value, & Reference

Scope

BLOCK SCOPING

Every variable that is created only exists inside the closest set of curly braces.



Scope

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(string args[]){  
    if (true){  
        int x = 0;  
        System.out.println("inside " + x);  
    }  
  
    System.out.println("after " + x);  
}
```

Scope

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(string args[]){  
    int x = 4;  
    System.out.println("before " + x);  
  
    if (true){  
        x = 0;  
        System.out.println("inside " + x);  
    }  
  
    System.out.println("after " + x);  
}
```

Scope

WATCH & LEARN

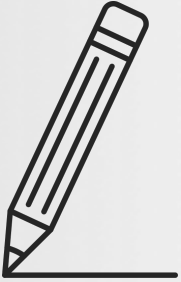
Close your laptop. Eyes on my screen. Pay attention.

```
public static void main(string args[]){
    int x = 4;
    System.out.println("main x " + x);
    System.out.println("method return " +
reassign(x));
    System.out.println("main x after call " + x);
}

public static int reassign(int x){
    System.out.println("x in the method " + x);
    x++;
    System.out.println("x post-increment " + x);
    return x;
}
```

Pass By Value

REFERENCES AND VALUES



Notebooks Ready? It's time for a mini lecture.



Pass By Value

REFERENCES AND VALUES

Pass by value is a term that put simply means that when we call a method, the parameter values are copied to a new variable and then the copied variable is passed into the function.

```
int x = 4;  
doubleInt(x);  
  
public static int doubleInt(int num) {  
    return num * 2;  
}
```

Pass By Value

REFERENCES AND VALUES

Pass by value is a term that put simply means that when we call a method, the parameter values are copied to a new variable and then the copied variable is passed into the function.

```
int x = 4;
```

```
doubleInt(x);
```



x is 4

```
public static int doubleInt(int num){  
    return num * 2;  
}
```

Pass By Value

REFERENCES AND VALUES

Pass by value is a term that put simply means that when we call a method, the parameter values are copied to a new variable and then the copied variable is passed into the function.

```
int x = 4;
```

```
doubleInt(x);
```



therefore..



num is 4

```
public static int doubleInt(int num){  
    return num * 2;  
}
```

Pass By Value

REFERENCES AND VALUES

We have 2 completely independent copies.

x



4

4

num

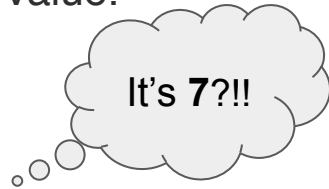


Pass By Reference

REFERENCES AND VALUES

Pass by reference is a bit of a misnomer in Java but people often use it anyway. It means that with some variables - namely objects (for example arrays), you don't get a new copy. Both variables share one copy of the value.

```
int[] x = {4, 5, 3};  
mutate(x);  
System.out.println(x[0]);
```



```
public static void mutate(int[] numArr){  
    numArr[0] = 7;  
}
```

Pass By Reference

REFERENCES AND VALUES

We have references to the same value

x

numArr

[4, 5, 3]



Overloading

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in PAIRS to complete all of the goals below.

Goals:

- Determine what will print at each line of the provided code



**10
minutes!**

Wrap Up

Module 1 Lesson 2

HOMEWORK

You don't have to submit your nightly homework, but you are expected to complete it.

- Read the following articles: <https://books.trinket.io/thinkjava/chapter4.html> and <https://books.trinket.io/thinkjava/chapter6.html>
- Finish katas, if not complete



Daily Assessment

You may leave after a staff member approves your assessment.



Daily Assessment

Work INDIVIDUALLY to complete all of the goals below.

Goals:

- Create a method called `printYear` that prints 1984
- Create a method called `isDog` that takes in a String and returns true if the String is “dog” (case-sensitive)