Pro Dev Session

Pro Dev Session

You may want to write this down.

Going forward when activities say work in PAIRs we're going to practice paired programming. This means one partner will be designated the driver and the other the navigator. The navigator will close their laptop and will NOT write any code.

If I'm the navigator what should I do?

- Plan how to tackle the problem with your partner. Watch for typos or logic errors. Ask questions. Google if needed.

If I'm the driver what should I do?

- Type. Talk. Explain your code as you go. Ask questions. Tell your navigator where they can help.

Pro Dev Session

You may want to write this down.

What do I do if I’m faster than my partner?

- If you’re driving, help them stay engaged. Ask “what should we do next”. Talk through your code as you go.
- If you’re the navigator, offer encouragement. Provide nudges. Do NOT take control of the keyboard.

What do I do if I’m slower than my partner?

- Ask questions. Try things. Don’t be afraid to make mistakes. Learn from your partner.

Pro Dev Session

You may want to write this down.

Why are we doing this?

- Paired programming is a common practice in many companies. As you work with different developers, it's a skill you will need to have. It also reduces bugs, increases team cohesion, and helps you develop better technical communication skills.

What if I don't want to?

- Tough. Paired programming is a necessary skill for developers these days. If you're scared of failure, talk to your pair, they probably are too. You can do this together! If you think a pair will hold you back, lucky you! You have the opportunity to learn by teaching. Learning to mentor is a vital part of moving from junior dev to mid level or senior. What a great opportunity to learn!

Pair Practice

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:

- Assume your instructor is a robot. Write instructions (in plain English) to tell your instructor how to take a plate, a loaf of bread, a jar of peanut butter, a jar of jelly, and a butter knife and make a peanut butter & jelly sandwich.
- Remember robots are quite literal.
- One team member should navigate while the other writes instructions.

Is this silly? Yes indeed. But it's also a good tool to learn to pair and better understand stepwise thinking.

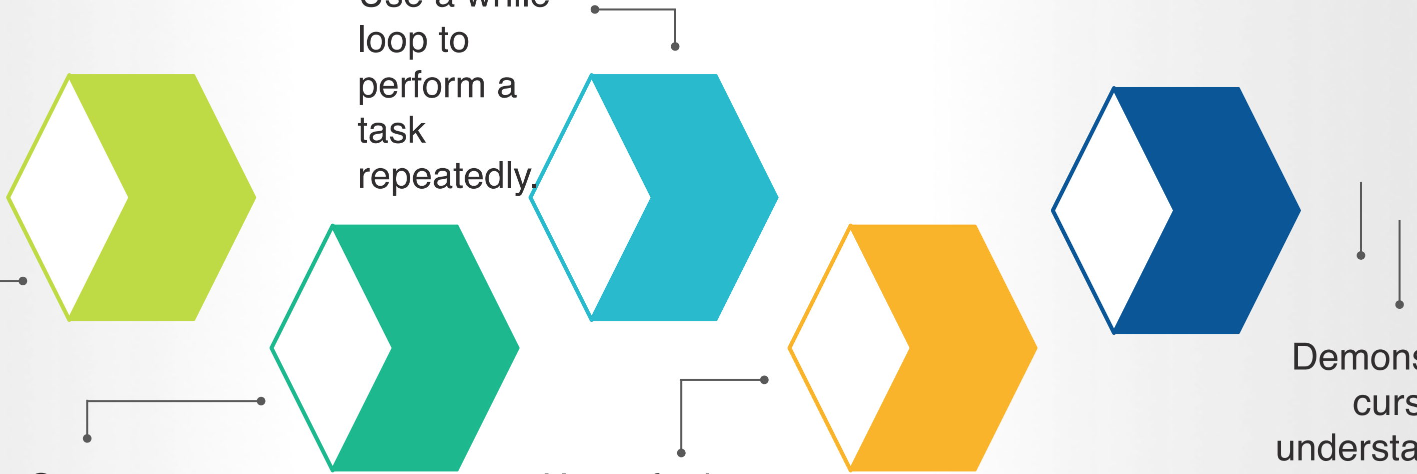
10 minutes!

Class Time.

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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



Use if and switch statements to create conditional execution

Create an array of elements

Use a while loop to perform a task repeatedly.

Use a for loop to perform a task repeatedly.

Demonstrate a cursory understanding of methods.

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!

Conditionals

Conditionals

WATCH & LEARN

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

```
int num1 = 4;
int num2 = 5;

if(num1 == num2){
        System.out.println("These numbers are equal");
}else {
        System.out.println("These numbers are not equal");
}
```

Conditionals

WATCH & LEARN

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

```
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();

if(name.length() > 7){
        System.out.println("Long Name!");
}else {
        System.out.println("Meh. You're average.");
}
```

Operators

LOGICAL OPERATORS

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

Operators

LOGICAL OPERATORS

Boolean values can be combined using logical operators.

&& || ! & | ^

Logical Operators

WATCH & LEARN

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

```
int num1 = 4;
int num2 = 5;
int num3 = 5;


System.out.println(num1 > num2); //false
System.out.println(num2 == num3);//true
System.out.println(num1 > num2 || num2 == num3); //true
System.out.println(num1 > num2 && num2 == num3); //false
```

Comparator Operators

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:

- Create a new project and paste the provided code in the main method. DO NOT RUN IT!
- With your partner, guess what each line will output
- Write down your guesses
- Run the file. Did your guesses match? If not, try to figure out what’s actually happening.

10 minutes!

Check-in Time

- What does `true && true` equal?
- What does `true || true` equal?
- What does `true && false` equal?
- What does `true || false` equal?
- What does `false || true` equal?

Switch Statements

WATCH & LEARN

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

```
int num = 4;
String operator = “+”;
if (operator == “-”){
          System.out.println(num - num);
}else if(operator == “+”){
          System.out.println(num + num);
}else if(operator == “*”){
          System.out.println(num * num);
}else if(operator == “/”){
          System.out.println(num / num);
} else {
          System.out.println(“Invalid operator”);
}
```

Switch Statements

CODE-A-LONG

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

Follow along with me as I complete this activity.

Goals:

- Refactor the calculator code to use a switch statement

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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

Use if and switch statements to create conditional execution

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 shortly.

Conditionals Statements

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:

- Create a new project. Using the provided instructions, complete all of the practice drills.

30 minutes!

lunch.

And Array We Go..

Arrays

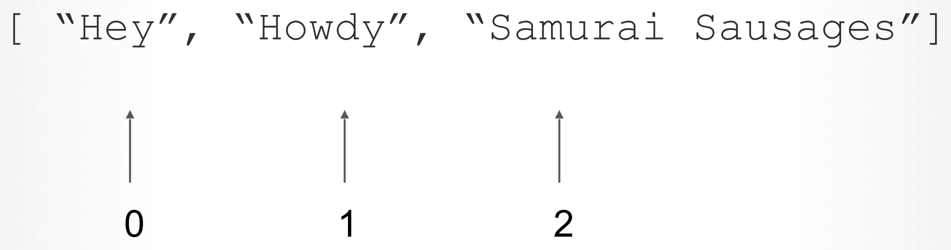
LISTS OF THINGS

Before we start learning about loops, we’re going to take a brief detour into arrays.

Arrays are just lists of things.

Like a list of integers or Strings or doubles.

Arrays are specifically numbered lists that start their numbering at zero. This means we can access elements by their position, or index, in the array.



```
[ “Hey”, “Howdy”, “Samurai Sausages”]
```

Arrays

WATCH & LEARN

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

```
int[] numList = {1, 2, 3, 4};

String[] wordList = {"banana", "pepper", "monkey", "today"};

System.out.println(numList[0]); // 1

System.out.println(numList[1]); // 2

System.out.println(wordList[2]); // "monkey"
```

Arrays

INDEPENDENT PRACTICE

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

Work INDIVIDUALLY to complete all of the goals below.

Goals:

- Create a new project.
- Create an array with the names of 7 of your classmates. Don’t know 7 names? Get up and meet some people!
- Print the name of the second person in the array.

5 minutes!

Arrays

WATCH & LEARN

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

```
int[] arr1 = new int[4];
arr1[0] = 10;
arr1[1] = 20;
arr1[2] = 30;
arr1[3] = 40;


int[] arr2 = {10, 20, 30, 40};
```

Check-in Time

- What is the index of the 3rd element in an array?
- What is an example of something you might store in an array if you were building a social media site?
- How would you create an array of Strings?
- What about an array of doubles?

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 shortly.

Oodles of Loops

Oodles of Loops

WHILE, DO WHILE, & FOR

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

Oodles of Loops

WHILE, DO WHILE, & FOR

Loops do the same thing over and over again until a specified condition is met.

There are 3 primary types of loops:

- While Loops
- Do While Loops
- For Loops

We’ll look at each individually and then you’ll have an opportunity to play with all 3.

Oodles of Loops

WHILE, DO WHILE, & FOR

The While Loop

```
int count = 0;

while (count < 4){
          System.out.println(count);
          count = count + 1;
}
```

What do you think will print?

Oodles of Loops

WHILE, DO WHILE, & FOR

The While Loop

```
int count = 0;

while (count < 4){
        System.out.println(count);
        //count = count + 1;
}
```

Let’s remove 1 line

What do you think will print?

Oodles of Loops

WHILE, DO WHILE, & FOR

The While Loop

```
int count = 0;

while (count < 0){
          System.out.println(count);
          count = count + 1;
}
```

What do you think will print now that the condition changed?

Oodles of Loops

WHILE, DO WHILE, & FOR

The While Loop

```
int count = 0;

do{
          System.out.println(count);
          count = count + 1;
}while (count < 0);
```

What do you think will print now that the loop type changed?

KeyPoint

Unlike a while loop, a do while loop always executes the code once regardless of the condition.

Oodles of Loops

WHILE, DO WHILE, & FOR

This while loop can be converted into a for loop

```
int count = 0;

while (count < 4){
          System.out.println(count);
          count = count + 1;
}
```

```
for (int count = 0; count < 4; count = count + 1){
     System.out.println(count);
}
```

Oodles of Loops

WHILE, DO WHILE, & FOR

This for loop can be written more concisely

```
for (int count = 0; count < 4; count = count + 1){
    System.out.println(count);
}
```

```
for (int i = 0; i < 4; i++){
    System.out.println(i);
}
```

KeyPoint

Use a for loop when you know the number of iterations you want.
Use a while loop when you know the condition that should be met but not the exact number of iterations.

Oodles of Loops

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 while loop
- Create a do while loop
- Create a for loop

Oodles of Loops

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. For each problem decide which loop is best.

Goals:

- Print `“Hello”` 15 times
- Prompt the user for their name. Continue to prompt the user until they input `“Louise”`.

10 minutes!

Oodles of Loops

INDEPENDENT PRACTICE

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

Work in <u>PAIRS</u> to complete all of the goals below.

Goals:

- Think back on everything you know about loops and everything you know about arrays.
- Add in that `array.length` gives you back the number of elements in an array.
- Try to use a loop to print out every element in the array.
- Remember that `arrayName[0]` prints the accesses 1st element and `arrayName[1]` accesses the 2nd element.

No matter what, don’t stop trying. Think hard. Don’t use Google.
Feel your brain stretching and growing!

10 minutes!

A Quick Aside

You don’t need to write this down.

We are nearing the part in the course where someone inevitably asks ‘When am I ever going to need to build a program that prints “Hello” 15 times?’

- You won’t. You also won’t need to print the day of the week based on a number or determine if a number is Hi or Low.

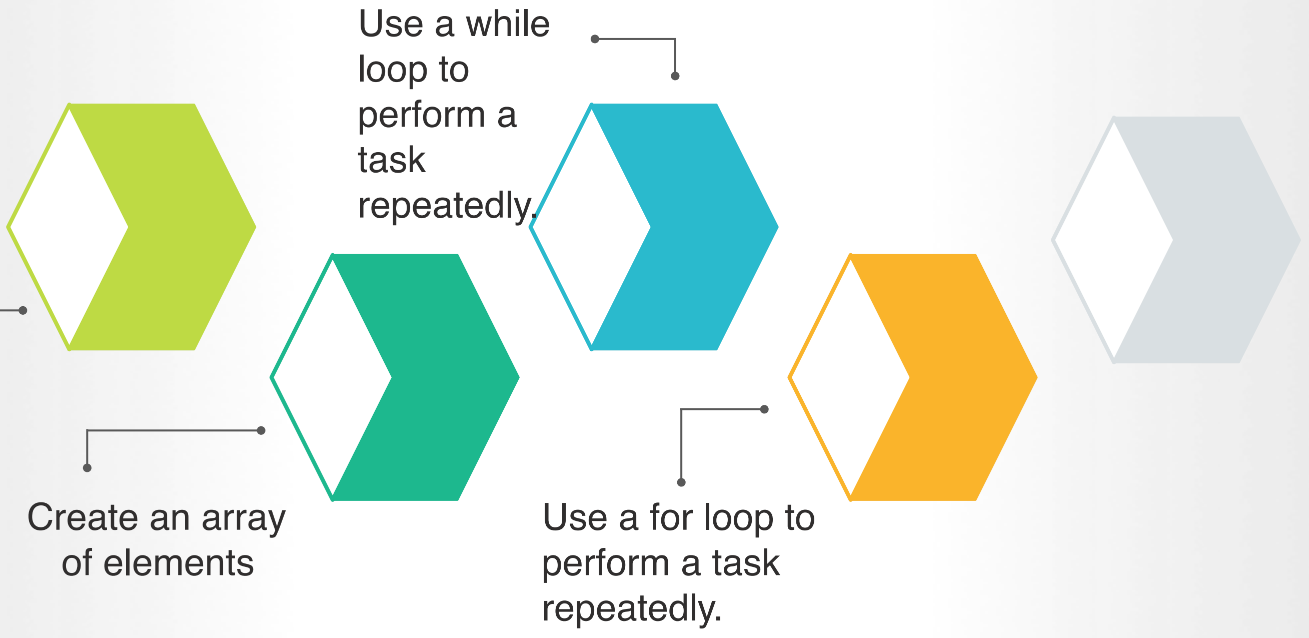
Asking when am I going to do X, is like a football player asking their coach when they’ll have to push a sled down the field in a game. They won’t.

It’s training. It’s conditioning. It’s generating the muscle memory you’ll need for the tough things ahead.

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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


Use if and switch statements to create conditional execution
Create an array of elements
Use a while loop to perform a task repeatedly.
Use a for loop to perform a task repeatedly.

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.

Sneak Peek

Note that next class will start with the sneak peek topic. You are NOT expected to master this today.

Methods

CODE ORGANIZATION + MORE

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

Methods

CODE ORGANIZATION + MORE

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 and calls any other methods you may need to call.

Let’s look at an example to make some sense of this…

Methods

CODE ORGANIZATION + MORE

```
class Arithmetic{
    public static void main(String[] args){
        int favNum = 9;
        int sum = add(favNum, 3);
        int diff = subtract(10, favNum);
        System.out.println(diff);
    }

    public static int add(int num1, int num2){
        return num1 + num2;
    }

    public static int subtract(int num1, int num2){
        return num1 - num2;
    }
}
```

Check-in Time

- What are some other methods that it might make sense for the Arithmetic class to have?

Methods

CODE ORGANIZATION + MORE

Add and subtract are methods. Methods are blocks of code that perform and action. A class is a collection of related methods and properties.

The Arithmetic class could also have a multiply method, a square method, a squareRoot method, etc.

Methods are like black boxes that take in some parameters and output a result. No part of the program outside of the method have to worry about how it gets the result.

Take the add method: It takes in 2 integers and returns an integer that represents the sum. The rest of the program doesn't have to worry about how to add. It can just call the add function, pass it 2 numbers and trust that the right number will come back.

Methods

CODE ORGANIZATION + MORE

An add method may seem trivial. But suppose it was a method that calculated how much weight a steel cable could hold given its diameter, number of strands of steel in the cable, and level of exposure to corrosive elements. That sounds tricky!

With a calculateMaxWeight method, the rest of your program doesn’t need to worry about this complex calculation. Any part of the program that needs to know this data, can just pass in the right variables and expect the correct result back.

Methods

CODE ORGANIZATION + MORE

Reminder: Add and subtract are methods. Methods are blocks of code that perform and action. A class is a collection of related methods and properties.

The Arithmetic class could also have a multiply method, a square method, a squareRoot method, etc.

Methods are derived of a few important parts.

1. The access modifier. This can be public, private, or protected (we'll discuss this one next week. For now just use public)
2. The return type (that is what kind of data will the method hand back when it's done)
3. The name (you can call it anything that you could call a variable)
4. The parameters (the types and names of data that will be passed into the method)

Methods

CODE ORGANIZATION + MORE

Totally lost? That’s okay. Remember this is a sneak peek at what we’ll focus on next class.

Just watch for a bit and see if you can pick up any intuitions.

Methods

WATCH & LEARN

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

```
class Combiner{
     public static void main(String[] args){
          System.out.println(add(1, 4)); // What will print?
          System.out.println(concat(“cat”, “dog”)); // What will print?
          System.out.println(logicalOr(true, false)); // What will print?
     }

     public static int add(int num1, int num2){
          return num1 + num2;
     }

     public static int concat(String str1, String str2){
          return str1 + str2;
     }

     public static int logicalOr(Boolean bool1, Boolean bool2){
          return bool1 || bool2;
     }
}
```

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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

Use if and switch statements to create conditional execution

Create an array of elements

Use a while loop to perform a task repeatedly.

Use a for loop to perform a task repeatedly.

Demonstrate a cursory understanding of methods.

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.

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 article: https://www.programiz.com/java-programming/operators
- Watch the following video: https://www.youtube.com/watch?v=nyIjLIBbgTU

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 new project.
- Create an array that holds the age of your family members (or friends).
- Print all the ages using a loop.
- If the second element is greater than 5, print `“you’re old”`.