Pro Dev Session

Pro Dev Session

You may want to write this down.

At every company, email is a major form of communication - so major, that managing your inbox can be difficult.

Having a good email management strategy is key to success. Even if you don't need it in the first 6 months, there will come a time in your career where you receive so many emails that a management strategy will be vital.

Today, we'll learn to use Gmail's priority inbox system. If you don't use Gmail, explore a similar implementation in your email client of choice.

Pro Dev Session

You may want to write this down.

Priority inboxing allows you to keep a sorted list of emails.

This list essential becomes your todo list each day.

In the next activity you'll set up your priority inbox and calendar reminders that help you keep your inbox at zero.

Practice

In Gmail, click the gear icon  > settings > labels
Click create new label and add a label named #action. Add another label named #waiting.
In the left menu, select the three dots  next to each label.
Choose green as the #action label color and red as the #waiting label color.

In the settings window select inbox. Under inbox type select priority inbox.
Under inbox sections, set the following:

1. Unread
2. More options > #action
3. More options > #waiting
4. Everything else

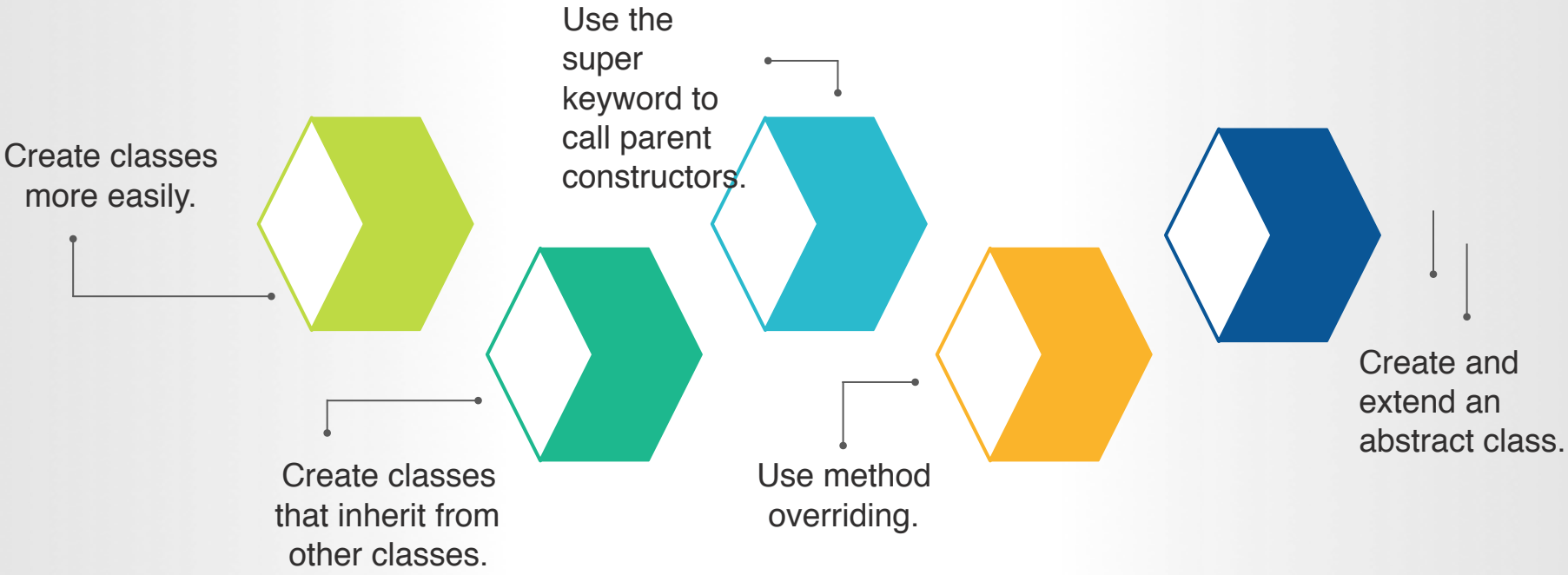
You're all set. As messages come in they will appear at the top of your inbox. Set a calendar reminder to check your inbox 3 times per day. Each time immediately respond if possible, mark with an
#action label if the response requires more than 5 min, or mark with a #waiting label to indicate that you need to take action but are awaiting information from another party first. Your #action list is your task list each day. Don't let #action items linger.

Stand Up

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

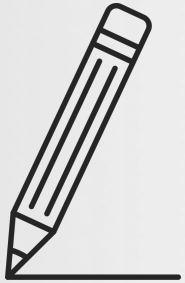
By the end of class today, you will be able to:



Inheritance

Inheritance

RELATIONSHIPS BETWEEN CLASSES



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

Inheritance

RELATIONSHIPS BETWEEN CLASSES

Inheritance refers to when one class extends another.

The extended class is referred to as the parent or superclass.

The extending class is referred to as the child or subclass

Inheritance represents an is-a relationship

Child classes get a copy of all public or protected methods and properties from the parent class.

Inheritance

```
public class Mammal {
        private boolean isWarmBlooded = true;
        private String name;
    private String sound;

        public void speak(){
                System.out.println(this.sound + " " + this.sound);
        }
}
---
public class Dog extends Mammal {
        public Dog(String name){
                this.setName(name);
        }

        public void pant(){
                System.out.println("Hu hu hu hu hu");
        }
}
```

Inheritance

CODE-A-LONG

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

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

        Dog rover = new Dog("Rover");
        rover.speak();
        rover.pant();

        Mammal betti = new Dog("Betti");
        betti.speak();
        betti.pant();
}
```

Classes &Objects

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 your own classes with an is-a relationship.
- Create a superclass and 2 subclasses.

15 minutes!

Inheritance

RELATIONSHIPS BETWEEN CLASSES

The super keyword is used to invoke superclass methods within the subclass. This is especially necessary in the constructor. Learn the following:

If a constructor does not invoke a superclass constructor, no-argument constructor of the superclass is automatically called. If the superclass does not have a no-argument constructor (ie. if you created a constructor in the superclass so that no default exists), you will get a compile-time error.

The superclass in turn calls its superclass's constructor (if it has a superclass).

This process is called constructor chaining.

Inheritance

```
public class Mammal {
    private String name;
    private String sound;
}

---

public class Dog extends Mammal {
    public Dog(String name){
        this.setName(name);
    }
}
```

A-Okay!
Default Constructor
Chaining!

Inheritance

```
public class Mammal {
    private String name;
    private String sound;

    public Mammal(String sound){
        this.sound = sound;
    }
}

---

public class Dog extends Mammal {
    public Dog(String name){
        this.setSound("Ruff");
        this.setName(name);
    }
}
```

This is illegal!

Inheritance

```
public class Mammal {
    private String name;
    private String sound;

    public Mammal(String sound){
        this.sound = sound;
    }
}

---

public class Dog extends Mammal {
    public Dog(String name){
        super("Ruff");
        this.setName(name);
    }
}
```

A-Okay!
Default Constructor Chaining!

Check-in Time

- Do you need to use the super keyword to call the superclass constructor if you have not defined a constructor in the superclass?
- What if you have defined a no args constructor in the superclass?
- What if you have defined a constructor in the superclass with identical parameters to the constructor in the subclass?
- What if you have NOT defined a constructor in the superclass with identical parameters to the constructor in the subclass but you have defined some constructors in the superclass?

Classes &Objects

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:

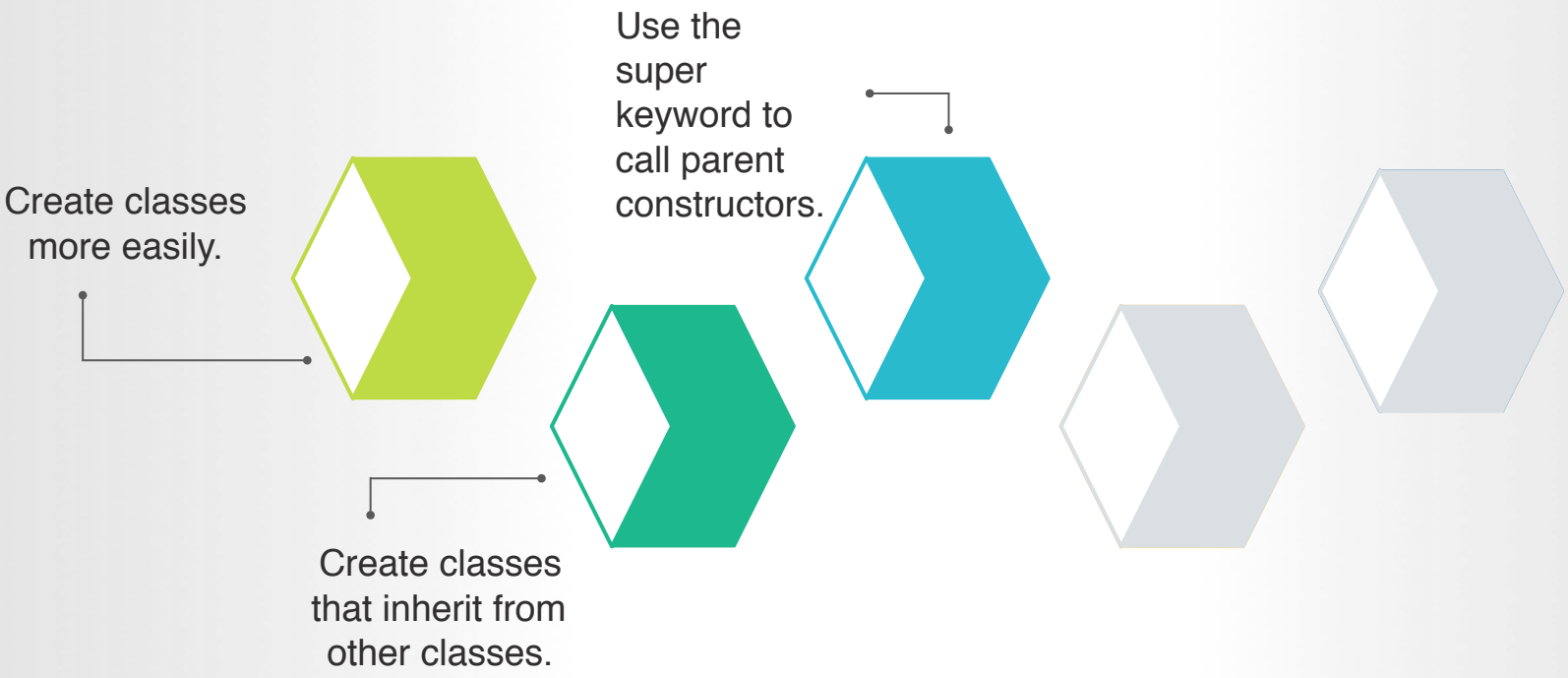
- Create a Car class with numOfDoors, isOn, and color properties. IsOn should default to false. The constructor should accept values for numOfDoors and color.
- Create a Minivan and Sportscar class that inherit from Car
- Create a constructor in each class that uses the super keyword to call the superclass constructor.

15 minutes!

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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



Stay Seated &Take 3 Deep Breaths.

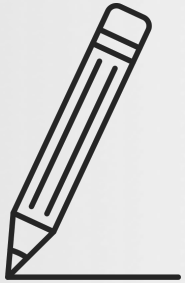
RELAX.

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

Method Overriding

Method Overriding

OBJECT ORIENTED PROGRAMMING CORNERSTONES



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

Method Overriding

OBJECT ORIENTED PROGRAMMING CORNERSTONES

A subclass can provide a specific implementation of a method defined in a superclass.

This means if we have identical method signatures in the subclass and superclass and we instantiate an object from the subclass, the subclass implementation of the method will be used.

Method Overriding

WATCH & LEARN

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

```
public class Developer {
    public void work(){
        System.out.println("Today I will do my work")
    }
}

---

public class TeamLead extends Developer {
    public void work(){
        System.out.println("I will ensure my team completes their their work!");
    }
}
```

Method Overriding

CODE-A-LONG

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

Let's work together complete all of the goals below.

Goals:

- Create a superclass called Building and subclass called Home.
- Building should have length, width, height, and roomCount properties and a constructor.
- Building should have a method called addRooms that increments the roomCount.
- Home should have a bedroomCount and paintColor.
- Create a paint method with no args that prints "ah fresh paint".
- Overload this method with a String argument for a new paint color. Update paintColor and print "ah fresh <color> paint ".
- Override the addRooms method so that it increments bedroomCount.

Method Overriding

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:

In the project you just built as a class

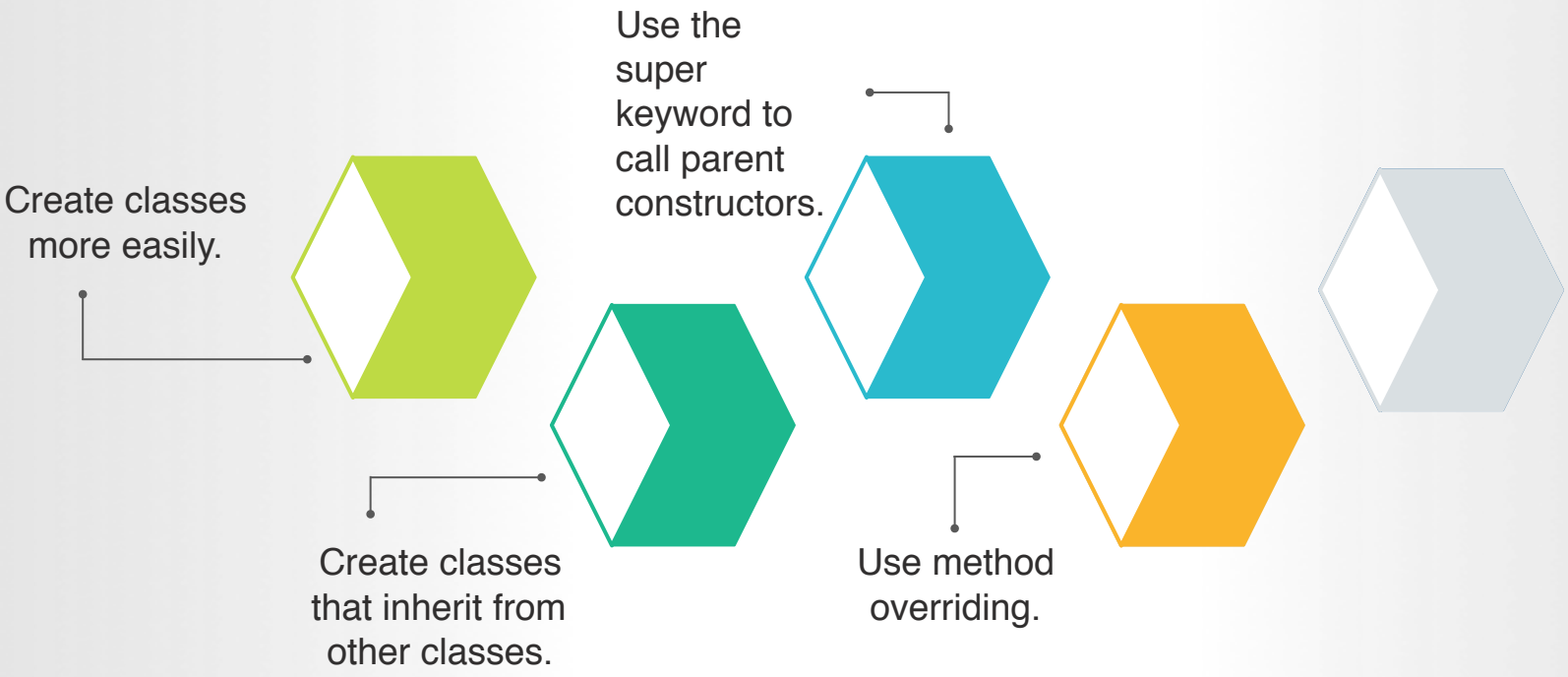
- Create an additional method to override
- Create and additional method to overload

10 minutes!

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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

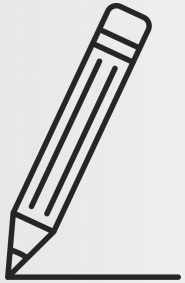


lunch.

Abstract Classes

Abstract Classes

OBJECT ORIENTED PROGRAMMING CORNERSTONES



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

Abstract Classes

OBJECT ORIENTED PROGRAMMING CORNERSTONES

Abstract classes are used when all subclasses should share certain method implementations and should share certain method signatures with different implementations

Abstract Classes

OBJECT ORIENTED PROGRAMMING CORNERSTONES

Know the following characteristics of an abstract base class:

- They cannot be instantiated. Abstract base classes are meant to be extended; they are not meant to stand on their own.

- They can provide implementations of methods (just like regular classes).

- They can have properties (just like regular classes).

- They can define methods with no implementation. Regular classes cannot do this. Any class that extends the abstract base class MUST provide an implementation for each of these methods.

Abstract Classes

WATCH & LEARN

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

```
public abstract class Animal {
        private String name;
        private int numOfLegs;

        public void sleep(){
                System.out.println("zzzzzzz");
        }

        public abstract void eat();
}
```

Abstract Classes

WATCH & LEARN

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

```
public class Dog extends Animal{
        public abstract void eat(){
                System.out.println("mmmmm kibble");
        }
}

-----

public class PolarBear extends Animal{
        public abstract void eat(){
                System.out.println("mmmm seal!");
        }
}
```

Abstract Classes

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:

Using the previous example as a starting point:

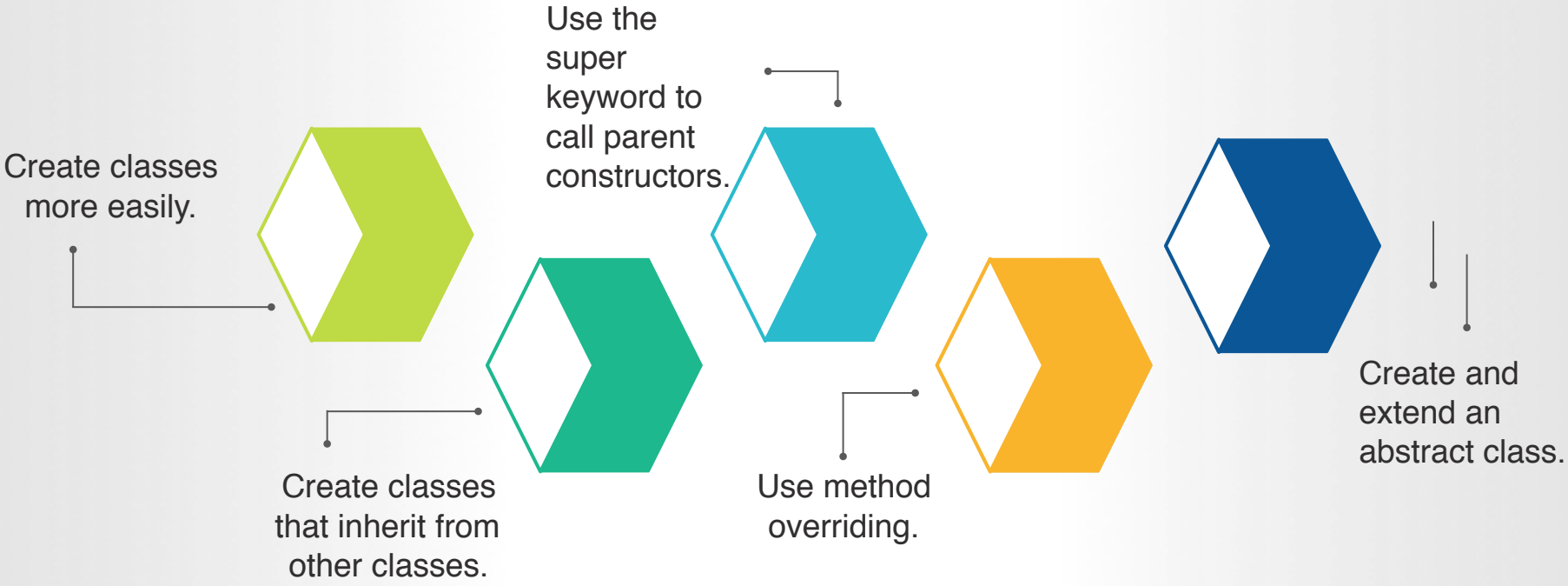
- Create an additional abstract method in the super class
- Implement this method in both subclasses

10 minutes!

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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



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

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.

40 minutes!

Stay Seated &Take 3 Deep Breaths.

RELAX.

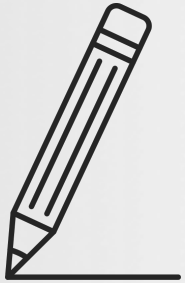
Now take a short walk. Clear your head. After a few minutes break, quickly review your notes. We'll start back in 10 minutes.

Sneak Peek

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

Interfaces

OBJECT ORIENTED PROGRAMMING CORNERSTONES



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

Interfaces

OBJECT ORIENTED PROGRAMMING CORNERSTONES

Interfaces define methods that all classes that implement the interface must implement.

In other words, interfaces are like abstract classes with only abstract methods.

Interfaces serve as a contract between various parts of a system. We'll revisit this point later.

Interfaces

WATCH & LEARN

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

```
public interface Shape{
    public double calculatePerimeter();
    public double calculateArea();
}
----------------------
public class Rectangle implements Shape {
    private double length;
    private double width;

    public double calculatePerimeter(){
        return this.length * 2 + this.width * 2;
    }
    public double calculateArea(){
        return this.length * this.width;
    }
}
```

Wrap Up

Module 2 Lesson 2

HOMEWORK

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

- Complete all 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 new project.
- Create an abstract class called Car
- Car should have the following properties: topSpeed, currentSpeed, color
- currentSpeed should default to 0
- Car should have a constructor that sets the color and topSpeed.
- Car should have an abstract method called accelerate that returns nothing.
- Create a concrete class called Minivan.
- Minivan should have a constructor that sets the color and topSpeed using the superclass constructor.
- Minivan should implement the accelerate method such that it sets the currentSpeed to half the top speed when called.