Pro Dev Session

Pro Dev Session

You may want to write this down.

Effectively using version control is an essential part of being a developer. You need to learn git well.

Today you will be independently learning about feature branching.

Raise your hand for assistance if you get stuck. We will be using this model for the duration of the course.

Agile Practice

INDEPENDENT PRACTICE

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

Goals:

- Read this article Feature Branching Article.
- Working with a partner, create a new repository.
- Each partner should clone the repository.
- Locally, each partner should make a branch, add a new markdown file, and push the new branch.
- Make a pull request against the master branch.
- After these branches are merged, used the pull keyword to update your local repositories

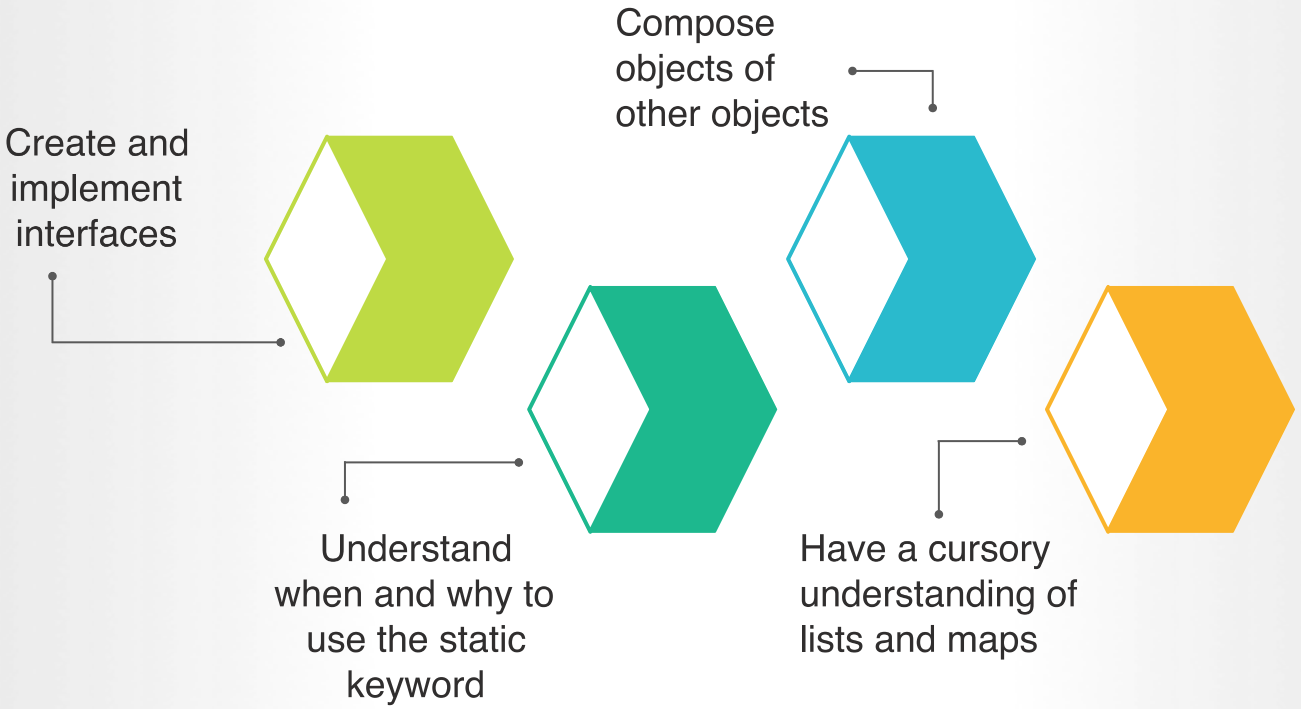
20 minutes!

Stand Up

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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



Create and implement interfaces
Compose objects of other objects
Understand when and why to use the static keyword
Have a cursory understanding of lists and maps

Interfaces

Interfaces

CONTRACTS BETWEEN CLASSES

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

Interfaces

CONTRACTS BETWEEN CLASSES

Interfaces are another tool to help us organize our code and express relationships between objects.

An interface advertises to the world that an implementing class has a particular set of capabilities.

Interfaces are often thought of as contracts in that they guarantee all implementing classes will have certain methods.

Interfaces

CONTRACTS BETWEEN CLASSES

Interfaces cannot have properties.

Interfaces cannot have method implementations (Java 8 does permit default method implementations, which are useful in certain circumstances).

Interfaces

WATCH & LEARN

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

```
public interface Storable {
    public void storeData();
    public void retrieveData();
}
```

Like a class, an interface must be declared. Interface declaration looks a lot like class declaration.

Interfaces

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 interface called Auditable.
- The Auditable interface should define a method called runAudit() that takes no parameters and returns nothing.
- The Auditable interface should also have another method called sendAuditToState() that takes no parameters and returns nothing.

10 minutes!

Interfaces

WATCH & LEARN

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

After an interface is declared, we must create a class to implement the interface.

- We must indicate that the class will be adhering to the interface (we call this implementing the interface).
- The class must provide and implement each of the methods declared in the interface.

```
public class SchoolRecord implements Storable {

    // Properties here
    // Additional methods here

    public void storeData() {
        // method body implementation here
    }

    public void retrieveData() {
        // method body implementation here
    }

}
```

Interfaces

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 class called SchoolRecord that implements Auditable.
- The runAudit() method should simply print the word “auditing” to the console.
- The sendAuditToState() method should simply print a confirmation message to the console.

Note: We'll be implementing actual functionality to the methods runAudit() and sendAuditToState() later. But for now just make sure that you understand how to add code to those individual methods!

10 minutes!

Interfaces

CONTRACTS BETWEEN CLASSES

Classes can implement more than one interface.

The interfaces are listed in a comma-delimited list after the implements keyword.

All methods from all interfaces must be implemented by the class.

Interfaces

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 the interface Storable.
- Storable should define two methods, storeData() and retrieveData(). Both methods should return nothing and take no parameters.
- Then add it to the list of interfaces the SchoolRecord implements.

Hint: If you a compiler error occurs, remember that you need to provide the implementations the Storable methods!

10 minutes!

Interfaces

CONTRACTS BETWEEN CLASSES

Interfaces have a very important role in software engineering. Because they guarantee that any class that implements them has identical method signatures, they allow us to change out one class for another.

Let’s look at an example of why this is powerful...

Interfaces

CONTRACTS BETWEEN CLASSES

Suppose we had several types of lists. They are like arrays in that they store ordered lists of elements. We have ListA, ListB, and ListC. All of them implement a List interface which guarantees that they will have a method called `add`, one called `get`, and one called `remove`.

ListA is the most efficient list for sets of data under 1,000,000 elements that have few duplicates.

ListB is almost always the most efficient when a list has lots of duplicates, unless the list is extremely large.

ListC is the most efficient list for sets of data over 1,000,000 elements that have few duplicates.

You are contracted to build a medical billing application and need to store and manipulate lists of health insurance claims.

Which list do you choose?

Interfaces

CONTRACTS BETWEEN CLASSES

Which list do you choose?

It depends on the expected data set. But data sets can change over time. If you choose ListA today, that could hugely impact the performance of your application in 5 years.

The good news is, because all the lists have identical methods, you can choose ListA today and switch it out for ListC later by changing a single line of code!

claimList.remove(42) does the same thing regardless of which type of list you choose so you need only switch the instantiation line later to change which type of list you use.

This is the power of interfaces.

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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

Create and implement interfaces

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.

Static Keyword

Static Keyword

THE NON-ACCESS MODIFIER

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

Static Keyword

THE NON-ACCESS MODIFIER

When a variable is declared as static, a single instance of that variable exists for ALL objects instantiated from the class.

Static variables are like global variable.

Methods can be static too. Static methods can only call other static methods and directly access static properties.

Static members of a class can be accessed before any objects are instantiated from the class.

Static Keyword

WATCH & LEARN

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

```
public class Student {
    private int id;
    private static stuCount = 0;

    public Student(){
        this.id = incrementStuCount();
    }

    private static int incrementStuCount(){
        return ++stuCount;
    }
}
```

Static Keyword

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:

A common use case for static methods (other than creating unique IDs and keeping object count) is in utility classes that need not be instantiated.

- Create a Math Class
- Create the following static methods: add, subtract, divide, multiply

10 minutes!

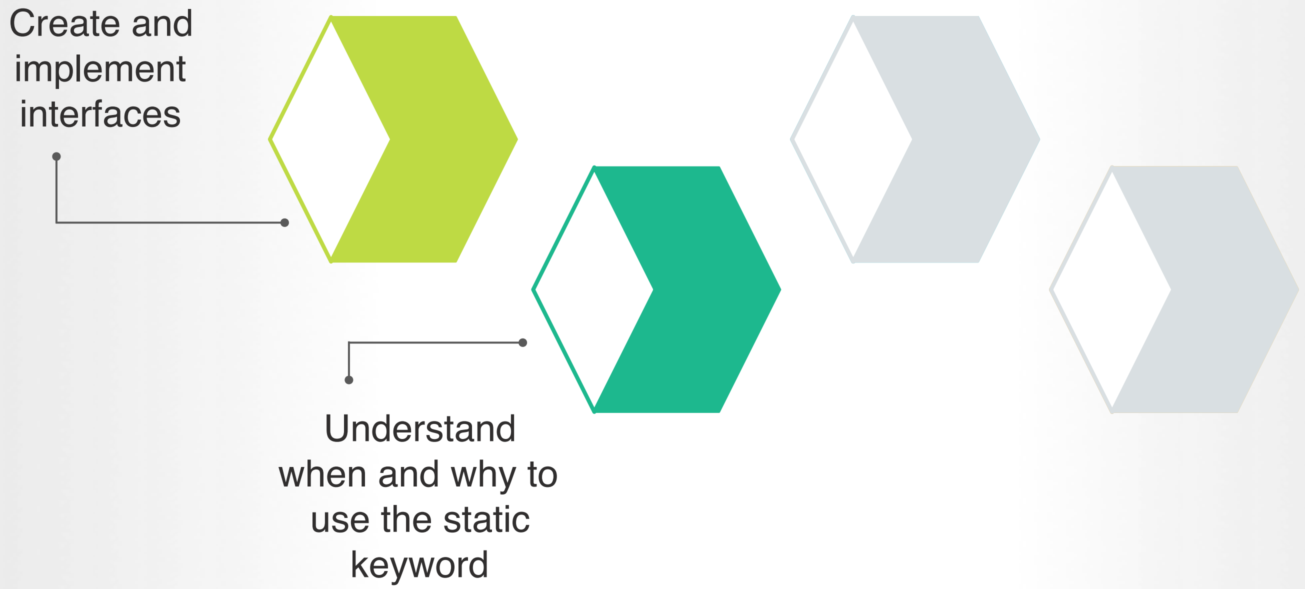
Check-in Time

- Why do you think the main method is static?
- Can the `this` keyword be used in a static method? If so, what does it refer to?

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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


Create and implement interfaces
Understand when and why to use the static keyword

lunch.

Composition

Composition

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

Composition

OBJECTS MADE OF OBJECTS

Composition is a technique that allows reuse of existing code and reduces the amount of code we have to write.

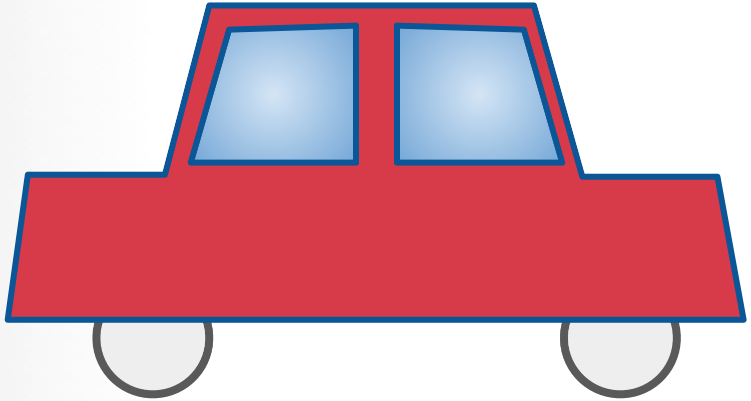
You shouldn't reinvent the wheel if you don't have to!

Composition

OBJECTS MADE OF OBJECTS

Imagine a car class. A really comprehensive car class that covers all the functions of a car.

We'll start with the doors.



Composition

```
public class Car{
    private boolean isFpDoorOpen = false; //front passenger door state
    private boolean isFdDoorOpen = false; //front driver door state
    private boolean isRpDoorOpen = false; //rear passenger door state
    private boolean isRdDoorOpen = false; //rear driver door state

    public void openFpDoor(){
        this.isFpDoorOpen = true;
    }

    public void closeFpDoor(){
        this.isFpDoorOpen = false;
    }

    public void openRpDoor(){
        this.isRpDoorOpen = true;
    }

    public void closeRpDoor(){
        this.isRpDoorOpen = false;
    }

    //and then some ...
}
```

Composition

```
public class Door {
    private boolean isOpen = false;

    public void open(){
        this.isOpen = true;
    }

    public void close(){
        this.isOpen = false;
    }
}

---

public class Car{
    private Door fpDoor = new Door();
    private Door fdDoor = new Door();
    private Door rpDoor = new Door();
    private Door rdDoor = new Door();
}
```

Composition

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 Person class.
- Use composition to include 2 hands and control of their state.
- The state should be a String and have a method to set the state.
- The states that a hand can be in should be:
    - Open
    - Grabbing
    - Waving
    - Pointing
    - Typing

*These should be the only possible states

15 minutes!

Objectives &Key Outcomes

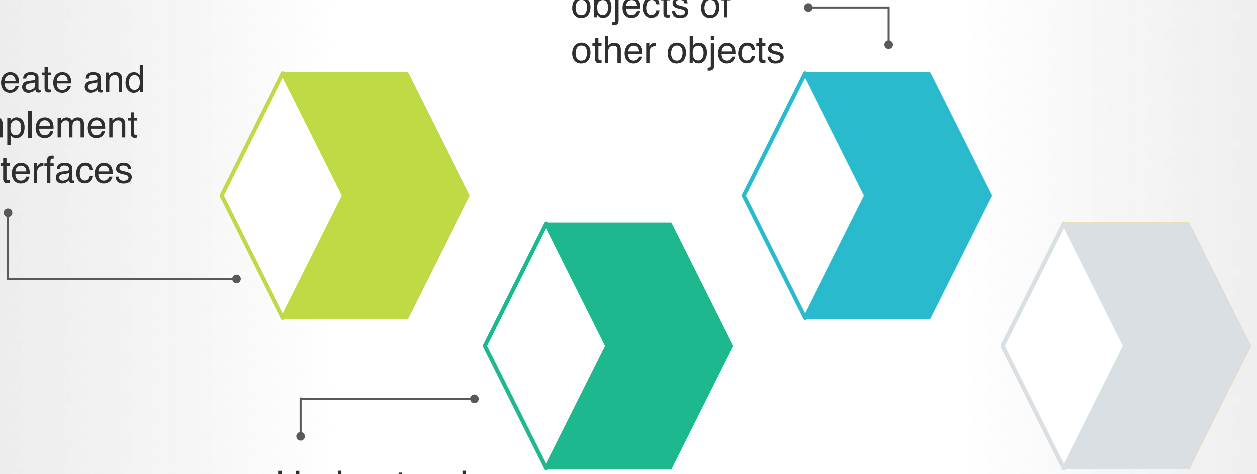
THE TAKEAWAYS FROM THIS CLASS

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

Create and implement interfaces

Understand when and why to use the static keyword

Compose objects of other objects



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 <u>INDEPENDENTLY</u> write your own code to complete all of the goals below.

Goals:

- Complete all the katas listed in the activity file.
- Complete the shapes-perimeters activity

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!

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.

Lists &Maps

CONTRACTS BETWEEN CLASSES

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

Lists &Maps

INTRODUCTORY DATA STRUCTURES

Java has pre-built data structures for commonly data storage needs.

Lists are like variable length arrays.

Maps are key/value pairs.

List and Map are interfaces with several concrete classes that implement them.

Sneak Peek

WATCH & LEARN

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

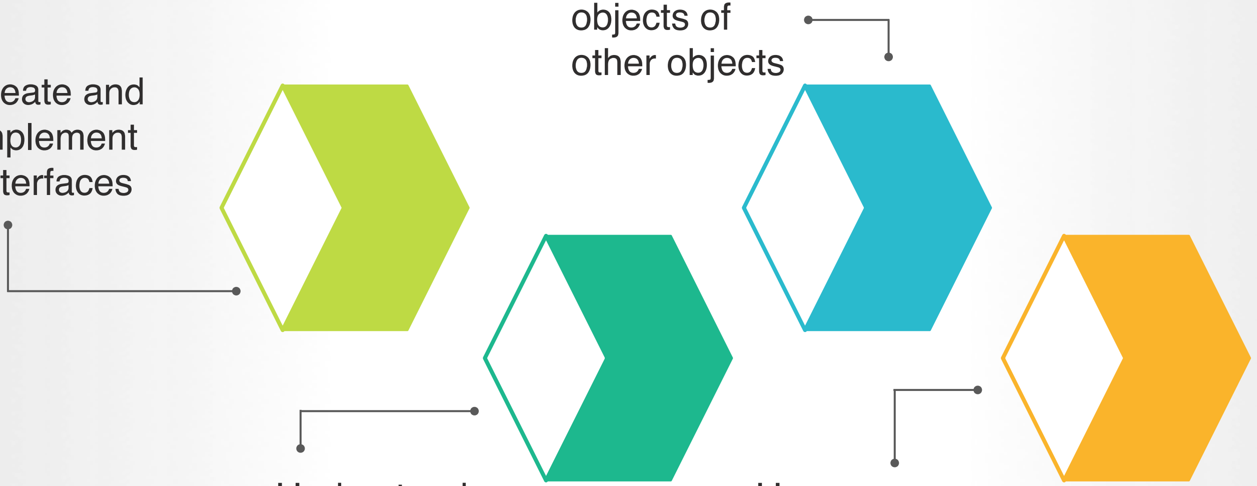
```
public static void main(String[] args){
    HashMap<String, String> nickNames = new HashMap<String, String>();
    nickNames.put("William", “Bill");
    nickNames.put("Robert", “Bob");

    ArrayList<String> roster = new ArrayList<String>();
    roster.add("Keisha");
    roster.add("Dawn");
}
```

Objectives &Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

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



Create and implement interfaces

Understand when and why to use the static keyword

Compose objects of other objects

Have a cursory understanding of lists and maps

Wrap Up

Module 2 Lesson 3

HOMEWORK

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

- 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 new project.
- Create an interface called Robot with the following methods:
  - public boolean turnOn()
  - public boolean turnOff()
- Create a class that implements the interface called RosieBot
- RosieBot should have a uniqueID and static rosieCount variable and associated methods.