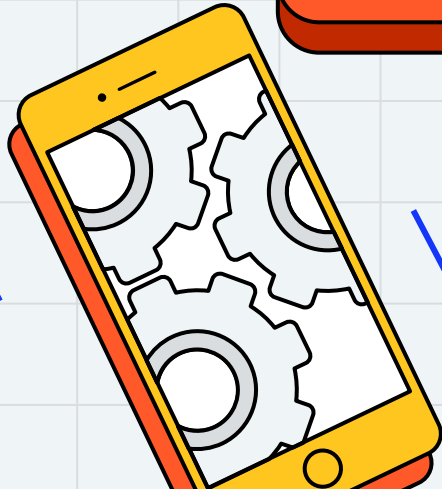


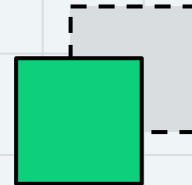


Application

Programming



Hend Alkittawi





OOP Concepts

Introduction to Java Interfaces

INTERFACES

- Java interfaces are particularly useful for assigning common functionality to possibly unrelated classes
- Java interfaces offer a capability requiring that unrelated classes implement a set of common methods
- A Java interface describes a set of methods that can be called on an object to tell it to perform some tasks or return some piece of information
- An interface should be used in place of an abstract class when there is no default implementation to inherit, that is, no fields and no concrete methods implementations
 - this allows objects of unrelated classes to be processed polymorphically

INTERFACES

- An interface declaration begins with the keywords `interface` and contains only constants and abstract methods
 - all methods declared in an interface are implicitly `public` abstract methods
 - all fields are implicitly `public`, `static`, and `final`

```
public interface InterfaceName {  
    public static final dataType varName;  
    public abstract returnType interfaceMethod();  
}
```

INTERFACES

- To use an interface, a concrete class must specify that it implements the interface and must declare each method in the interface with the signature specified in the interface declaration
- Java does not allow subclasses to inherit from more than one superclass, but it allows a class to inherit from one superclass and implement as many interfaces as it needs

```
public class ClassName implements InterfaceName
```

or

```
public class ClassName extends SuperClass implements InterfaceName
```

where **InterfaceName** maybe a comma-separated list of interface names

```
public interface Drawable {
    public void draw();
}
```

```
public class Tree implements Drawable {
    private String type;
    private double height;

    public Tree(String type, double height) {
        this.type = type;
        this.height = height; }

    // getters and setters are omitted

    @Override
    public void draw() {
        System.out.println("Drawing a tree with height " + getHeight() + " meters"); }
}
```

```
public class Rectangle implements Drawable{
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width; }

    // getters and setters are omitted

    public void draw() {
        System.out.println("Drawing a rectangle with length " + length + " and width " + width); }
}
```

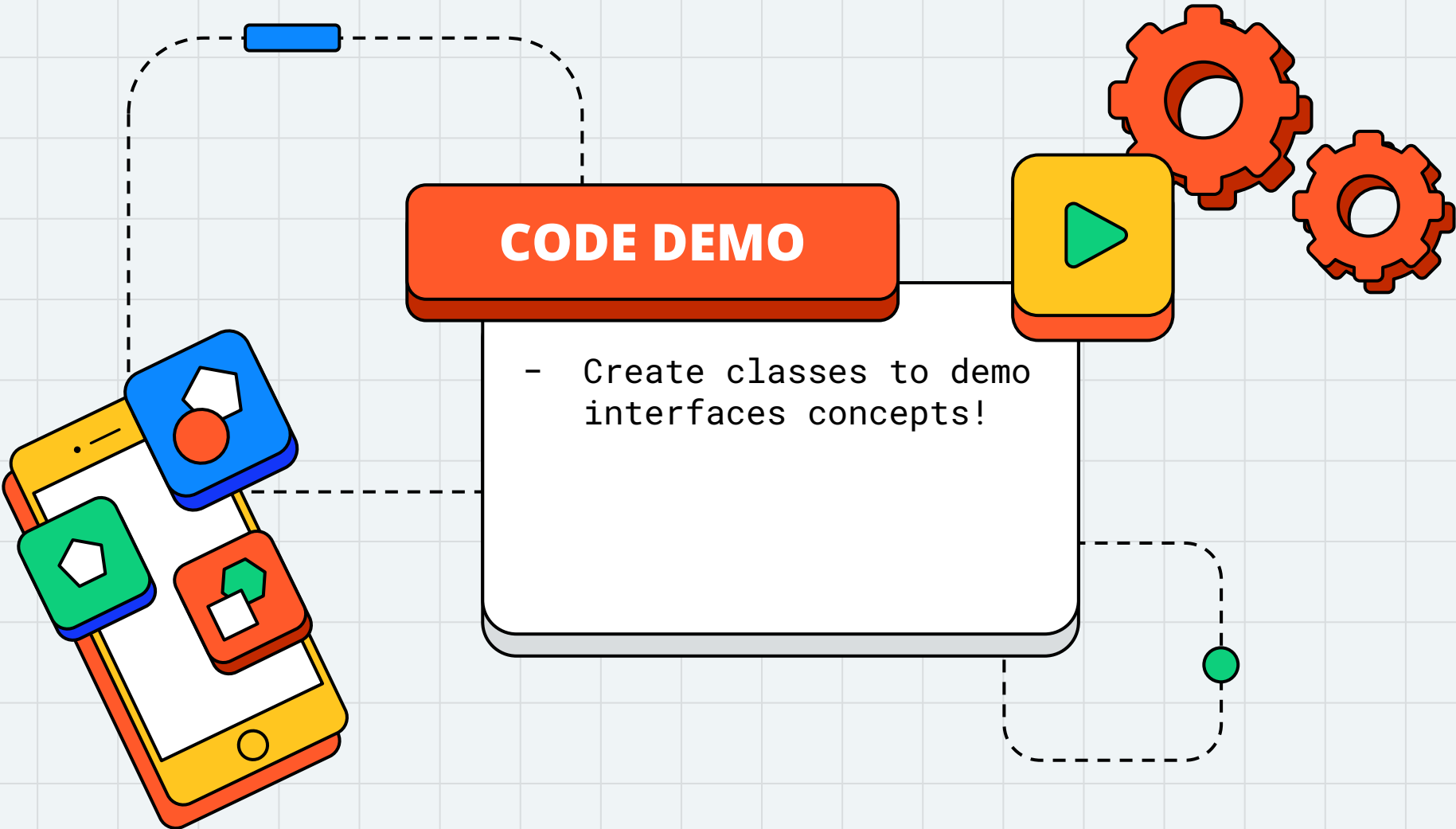
```
public class InterfaceDemo {

    public static void main(String[] args) {

        // Creating objects of different
        // classes that implement Drawable
        Drawable rectangle = new Rectangle(2.5, 7.2);
        Drawable tree = new Tree("Oak", 5.5);
        Drawable person = new Person("John", 30);

        // Array of Drawable objects
        Drawable[] drawables = {rectangle, tree, person};

        // Drawing all drawable objects
        for (Drawable drawable : drawables) {
            drawable.draw();
        }
    }
}
```





THANK

YOU!



DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online