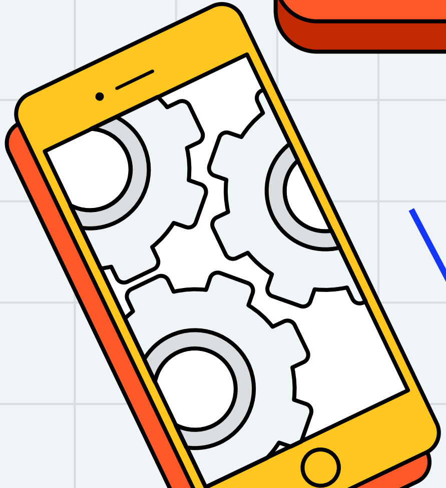
Application Programming

Hend Alkittawi



OOP Concepts

Java Abstract Classes

ABSTRACT CLASSES

- Sometimes it's useful to declare classes for which you never intend to create objects. These classes are called **abstract classes**.
- The purpose of an abstract class is to provide an appropriate superclass from which other classes can inherit and thus share common design.
- Abstract classes are used only as superclasses in inheritance hierarchies.
- Abstract classes are incomplete, and cannot be used to instantiate objects.

ABSTRACT CLASSES

- To make a class abstract, declare it with the keyword **abstract**

```
public abstract class MyClass { … }
```

- Abstract classes normally contain one or more **abstract methods**
- An abstract method is an instance method with the keyword abstract and its declaration
- An Abstract method **do not provide implementations**

```
public abstract myMethod();
```

- Constructors and static methods cannot be declared as abstract.

ABSTRACT CLASSES

- Subclasses must declare the missing pieces to become concrete classes from which objects can be instantiated.
- Each concrete subclass of an abstract superclass must provide concrete implementations of each of the superclass abstract methods.

```java
public abstract class Shape {

  private String color;

  public String getColor() {
    return color;
  }
  public void setColor(String color) {
    this.color = color;
  }
 public void display() {
   System.out.println("This is a " + getType() + " shape. Its color is: " +  getColor());
 }
 public abstract String getType();
 public abstract void draw();
}
```

```java
public class Circle extends Shape{

   private double radius;
   public double getRadius() {
     return radius;
   }
   public void setRadius(double radius) {
     this.radius = radius;
   }
   public void draw() {
     System.out.println("Drawing a " + getColor() + " circle with radius " + getRadius() + ".");
   }
   @Override
   public String getType() {
     return "Circle";
   }
}
```

```java
public class AbstractClassDemo {

  public static void main(String[] args) {
  Circle circle = new Circle();
  circle.setColor("red");
  circle.setRadius(5.0);
  circle.draw(); // implemented abstract method
  circle.display(); // concrete method from
                                       abstract class
  }
}
```

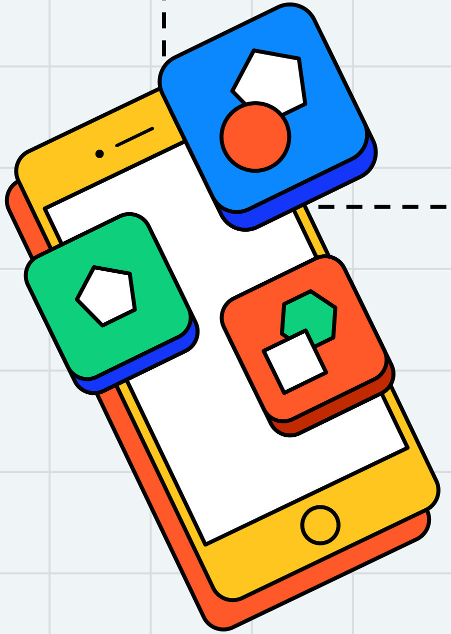
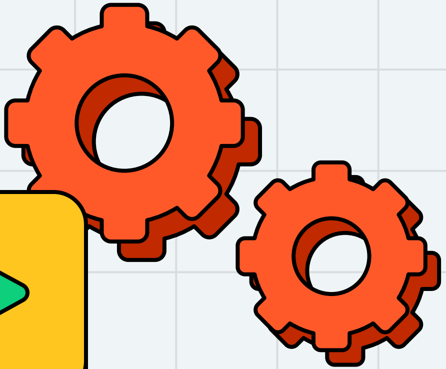
```java
public abstract class Shape {

  private String color;

  public Shape(String color) { this.color = color; }

  public String getColor() { return color; }
  public void setColor(String color) { this.color = color; }

  public void display(){ System.out.println("This is a " + getType() + " shape. Its color is: " +  getColor()); }
  public abstract String getType();
  public abstract void draw();
}
```

```java
public class Circle extends Shape{

 private double radius;

 public Circle(String color, double radius) {
   super(color);
   this.radius = radius; }

 public double getRadius() { return radius; }

 public void setRadius(double radius) { this.radius = radius; }

 @Override
 public void draw() {
   System.out.println("Drawing a " + getColor() + " circle with radius " + getRadius() + ".");
 }

 @Override
 public String getType() {
   return "Circle"; }
}
```

```java
public class AbstractClassDemo {

  public static void main(String[] args) {

//Shape shape = new Shape("blue"); // invalid

  Circle circle = new Circle("red", 5.0);
  circle.draw(); // implemented abstract method
  circle.display(); // concrete method from
                                abstract class
  }
}
```

CODE DEMO

- Create classes to demo abstract classes concepts!

THANK YOU!

DO YOU HAVE ANY QUESTIONS?

hend.alkittawi@utsa.edu

By Appointment

Online