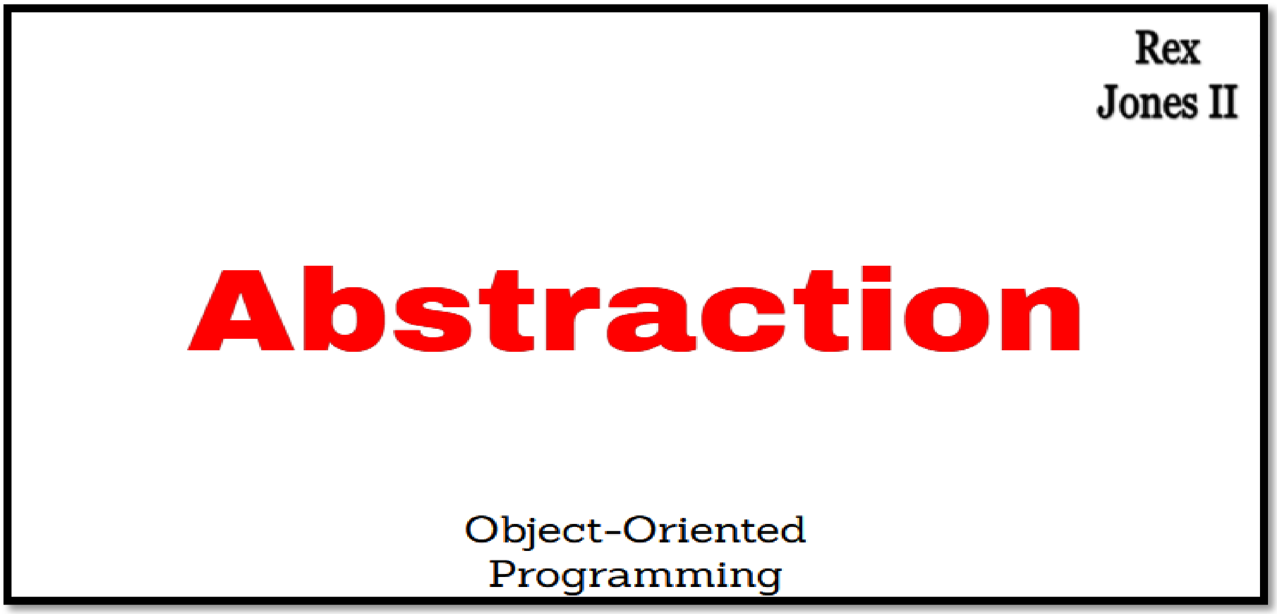
Rex
Jones II

(Transcript)
Abstraction Introduction



Rex
Jones II

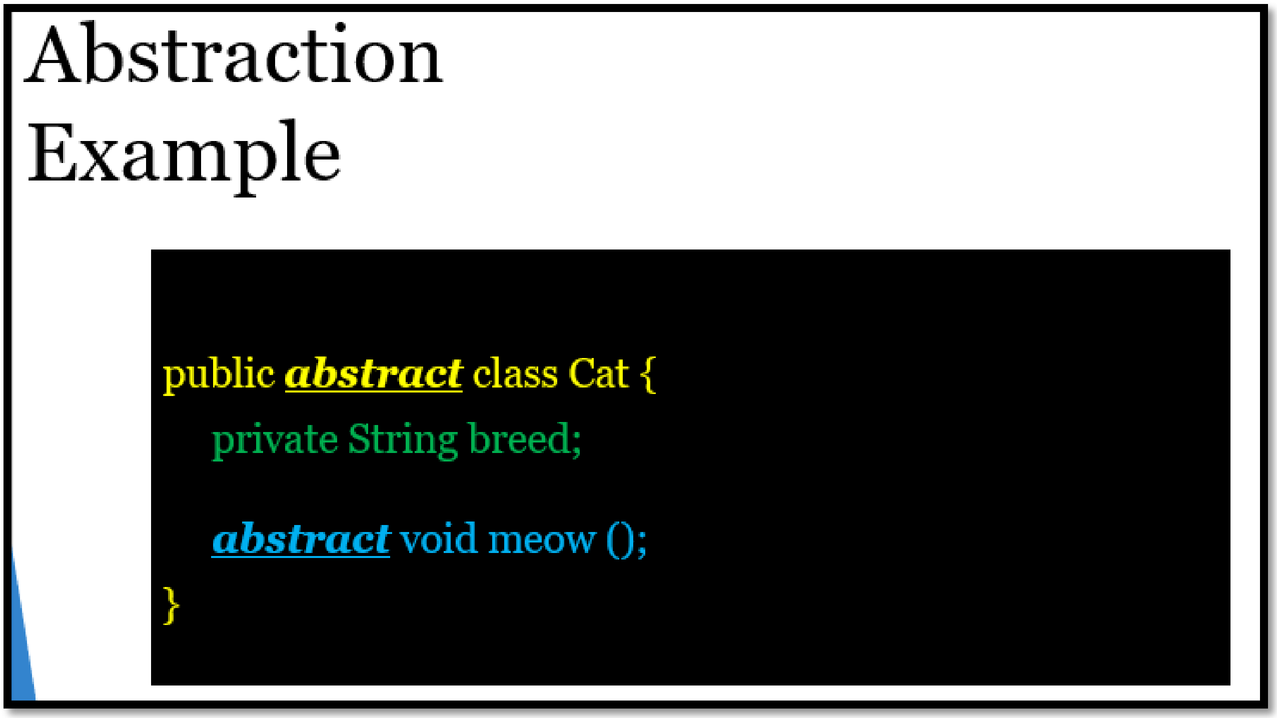
Abstraction

Object-Oriented
Programming

Introduction

In general, abstract indicates a feature apart from a particular object. When it comes to Object Oriented Programming, abstraction serves as a representative for real things. For example, we have a class named Cat that's abstracted into a type. Next, we declare a field named breed in the Cat class. Breed is a real characteristic that we can abstract from a cat. Last, is the behavior action of a cat. We abstract meow as a real action for a cat.

With this example, abstraction shows the necessary features of an object. Cat is an abstract concept because it displays a general characteristic and a general action. As a result, the type is a Cat with a breed characteristic and meow as an action. Did you notice the keyword abstract?



Abstraction Example

```
public abstract class Cat {
    private String breed;

    abstract void meow ();
}
```

Abstract is a non-access modifier for classes and methods but not for a variable like breed. The purpose of Abstraction is to define a template for a class or define a template for a method. Abstract classes are generic and operate as a superclass. Therefore, it can be extended which allows a subclass to fill in the details.

Abstract methods are also generic. They are designed to be overridden in the subclass. That's why we only see the signature of the method: abstract, void, meow, parenthesis, semi-colon, and no body within the curly braces. The details are implemented in the subclass which overrides the abstract method defined in the superclass. Static methods and constructors cannot be abstract.

Java also provide an interface which is similar to an abstract class. One of the differences between an abstract class and interface is: An abstract class can have abstract methods and fully defined methods. The fully defined methods are instance methods which provide a default action.

Traditionally, with an interface, there was not a way to supply a default action. All of the methods were abstract before Java Development Kit JDK 8. Now, we can define a default method to provide a default action. The primary motivation for default methods was to deliver a way to enhance interfaces without breaking existing code. Next, I'm going demo Abstract Classes and Abstract Methods.