Abstraction and Interface

Learning Objectives

- Learn and use **Java Abstraction**
- Learn and use **Java Interface**

Abstraction

Data **abstraction** is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either **abstract classes** or **interfaces**.

Abstraction

The **abstract keyword** is a non-access modifier, used for classes and methods:

- **Abstract class (restricted class)**

  Cannot be used to create objects

  Must be inherited from another class

- **Abstract method**

  Can only be used in an abstract class

  Does not have a body. The body is provided by the subclass (inherited from).

Example

```
abstract class Animal {
  public abstract void animalSound();
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

```
 Animal myAnimal = new Animal();

 // will generate an error
```

$\longrightarrow$ ERROR

```java
// Abstract class
abstract class Animal {
  // Abstract method
  //(does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

```java
// Subclass (inherit from Animal)
class Dog extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The dog says: woof woof");
  }
}
```

```java
class Main {
  public static void main(String[] args) {
    Dog myDog = new Dog(); // Create a Dog object
    myDog.animalSound();
    myDog.sleep();
  }
}
```

Interfaces

An **interface** is a completely "abstract class" that is used to group related methods with empty bodies:

```
// interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)

 }
```

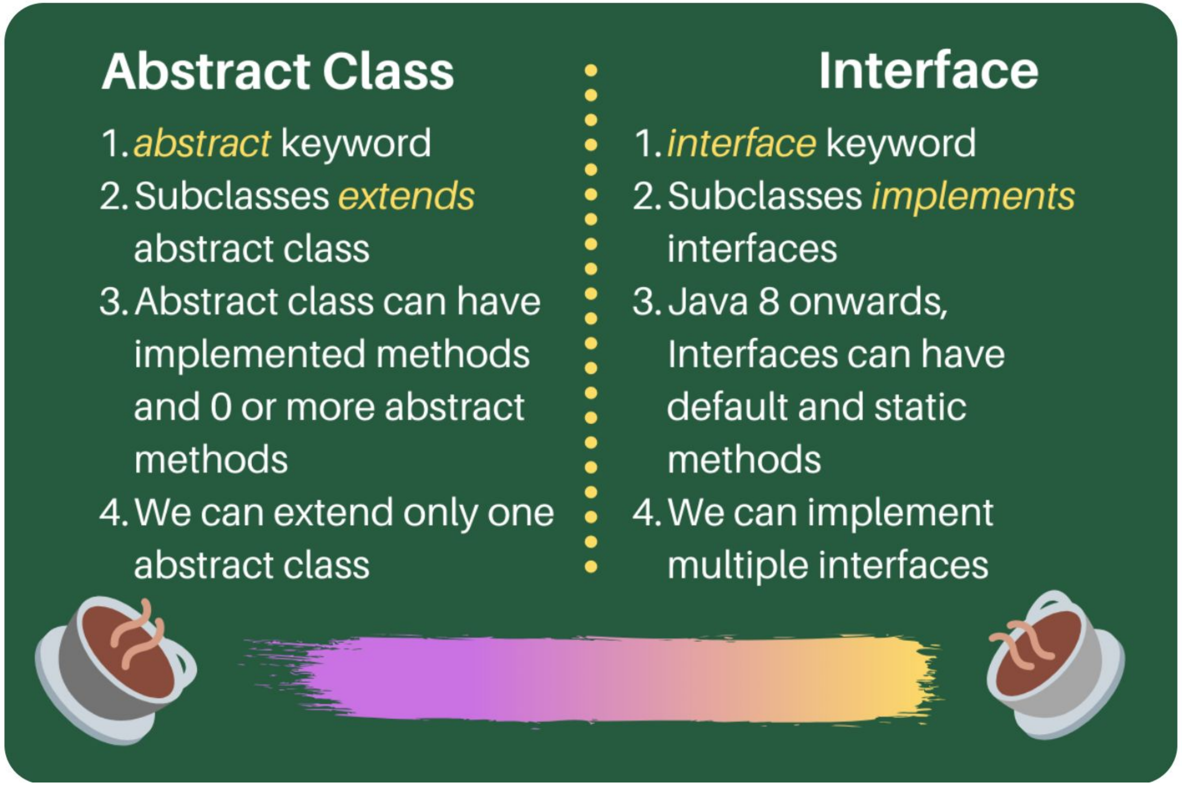
To access the interface methods, the interface must be "implemented" by another class with the `implements` keyword (instead of `extends`). The body of the interface method is provided by the "implement" class

```java
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
}
```

```java
// Dog "implements" the Animal interface
class Dog implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided
here
    System.out.println("The dog says: woof
woof");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}
```

```java
class Main {
  public static void main(String[] args)
{
    // Create a Dog object
    Dog myDog = new Dog();
    myDog.animalSound();
    myDog.sleep();
  }

}
```



Abstract Class	Interface
1. *abstract* keyword	1. *interface* keyword
2. Subclasses *extends* abstract class	2. Subclasses *implements* interfaces
3. Abstract class can have implemented methods and 0 or more abstract methods	3. Java 8 onwards, Interfaces can have default and static methods
4. We can extend only one abstract class	4. We can implement multiple interfaces