Object Oriented Programming (OOP)

Lecture6: Interfaces

Prepared by:
Mohamed Mabrouk

Those slides are based on slides by:
Dr. Sherin Mousa and Dr. Sally Saad

Lecture Outline

- What interfaces are
- How to implement interfaces
- How to define interfaces
- Interface data members
- Interfaces vs abstract classes

Interfaces

Interfaces

- An interface is a *PURE ABSTRACT CLASS*
- *Interfaces cannot be instantiated, like in abstract classes*
- *An interface constitutes a contract between a class and the outside world, and it is imposed at compile-time by the compiler*
- *Thus, if a class implements an interface, then it has to override all of the interface methods*

Interfaces

- It is like a *CONTRACT* that a class must adhere to if it implements it
- A class can implement *AS MANY INTERFACES AS REQUIRED*, while this is not the case for inheritance
- A class can extend at most one class but can implement many interfaces

Interface Components

- An interface can contain both fields and methods. However, those members are special
  - All data members are `public static final`, and there is no need to define them explicitly as `public static final`
  - All methods are `public abstract`, and there is no need to define them explicitly as `public abstract`

Interface Example

```
public interface Calculator {
	public static final int MIN_NUMBER = 0;
	public static final int MAX_NUMBER = 100;
	public abstract int add(int num1, int num2);
	public abstract int subtract(int num1, int num2);
}
```

- `public static final` for data members and `public static` for methods are not required and can be removed

Interface Example

- Same interface can be rewritten as

```
public interface Calculator {
	int MIN_NUMBER = 0;
	int MAX_NUMBER = 100;
	int add(int num1, int num2);
	int subtract(int num1, int num2);
}
```

Implementing an Interface

- That interface can be implemented as follows:

```
public class SimpleCalculator implements Calculator{
	@Override
	public int add(int num1, int num2) {
		return num1 + num2;
	}

	@Override
	public int subtract(int num1, int num2) {
		return num1 - num2;
	}
}
```

Implementation Details

- Use the `interface` keyword rather than `class`
- No constructors can be defined as the interface *CANNOT* be instantiated by itself
- To implement an interface use `implements` keyword

Implementation Details

- `public` and `private` access modifiers are not allowed
- Fields are `public static final` by default
- Methods are `public abstract` by default
- An interface can inherit (extend) another interface

Interface Inheritance Example

- Same interface can be rewritten as

```
public interface ScientificCalculator extends Calculator{
        float sin(int angle);
        float cos(int angle);
}
```

Implementation Details

- A class can extend at most one class but it an implement as many interfaces as required
- *JAVA SUPPORTS MULTIPLE INHERITANCE VIA INTERFACES ONLY*

Implementation Details

- When a class implements an interface, it has to override all its methods unless it is declared abstract
- In other words, if a class implements and interface and it is declared abstract it does not have to implement all interface methods
- If a superclass implements an interface → all its subclasses also implement it.

Why Interface

- How to use interfaces???
- Can you think of some interfaces???

Interfaces

- Following is just a couple of the most common interfaces:
  - `Comparable`
  - `Comparator`
  - `Serializable`
  - `Cloneable`

Usages of Interfaces

- Once an interface is implemented, then Java knows that some functionality is supported
- The most common and intuitive interface example is interface `Comparable`
- It has a single method `compare`

Usages of Interfaces

- It can be used for comparing two objects of the same type, e.g., two Student or Vehicle objects
- Comparing two or more objects helps in sorting them either ascendingly or descending
- It has form `int compareTo(Object o)`
- It returns one of three possible values, -ve, 0, or +ve

Usages of Interfaces

- -ve if the current object is smaller than the passed object
- 0 if both objects are equal
- +ve if the current object is larger than the passed object
- All common types, e.g., Integer, Float, String, implement `comparable` interface → This is why we can compare and ultimately sort an array of that type
- Therefore, we can sort an array of strings using `Arrays.sort()` or `Collections.sort()` for lists

Comparable Interface Example

```
public class Student implements Comparable{
	private int id;
	private String name;
	public Student(int id, String name){
		this.id = id;
		this.name = name;
	}
```

Comparable Interface Example

```
	@Override
	public int compareTo(Object obj) {
		Student otherStudent = (Student) obj;
		if(this.id < otherStudent.id){
			return -1;
		} else if(this.id > otherStudent.id) {
			return 1;
		} else {
			return 0;
		}
	}
}
```

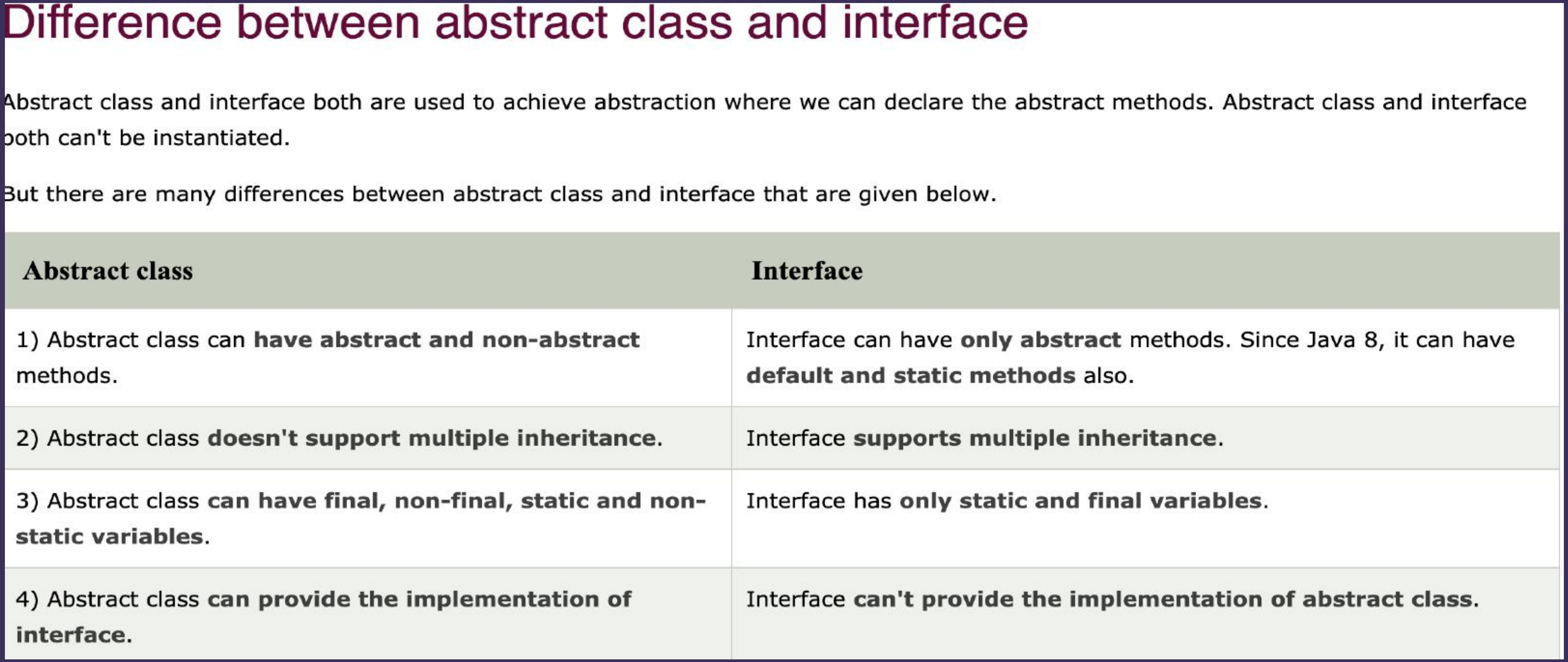
Comparable Interface Example

```
public static void main(String[] args) {
	Student[] arr = new Student[3];

	arr[0] = new Student(7, "Ahmed");
	arr[1] = new Student(1, "Mona");
	arr[2] = new Student(5, "Ashraf");

	Arrays.sort(arr);
}
```

Interfaces vs Abstract Classes



Difference between abstract class and interface

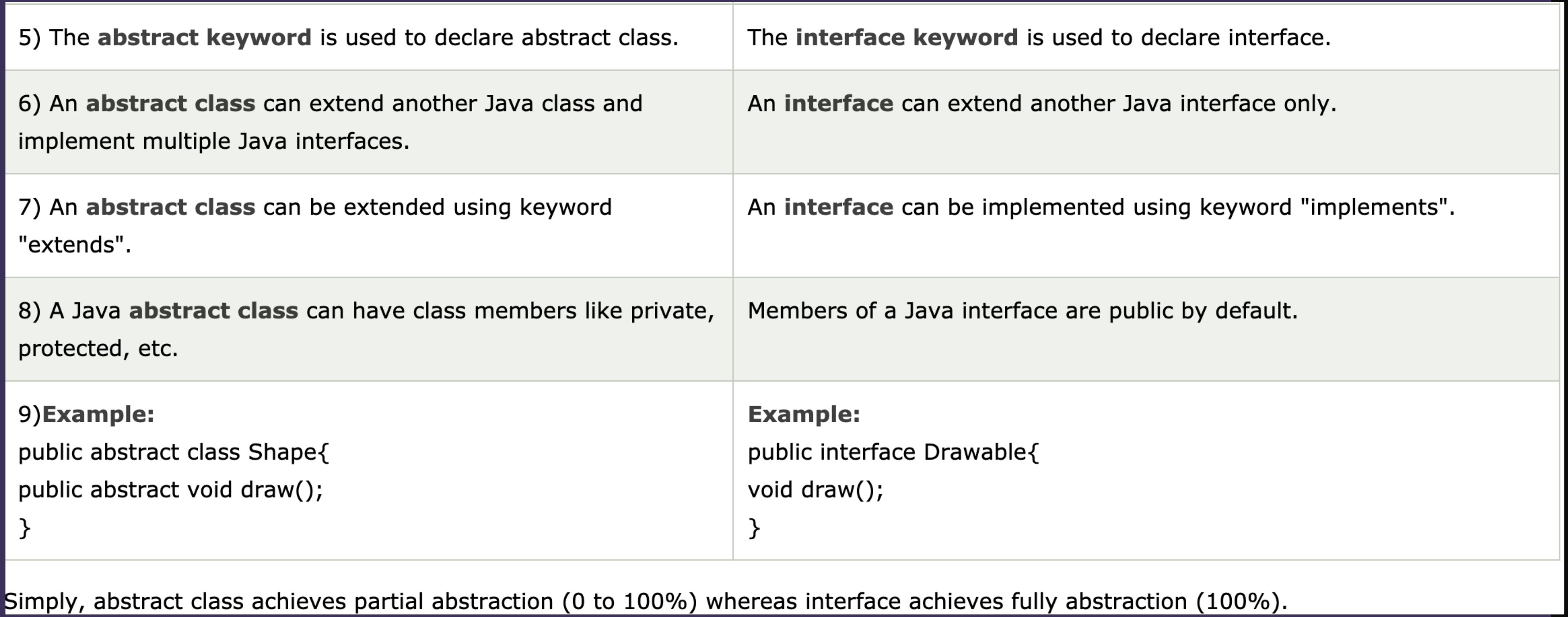
Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

But there are many differences between abstract class and interface that are given below.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance.**	Interface **supports multiple inheritance.**
3) Abstract class **can have final, non-final, static and non-static variables.**	Interface has **only static and final variables.**
4) Abstract class **can provide the implementation of interface.**	Interface **can't provide the implementation of abstract class.**

Source of this image is https://www.javatpoint.com/difference-between-abstract-class-and-interface

Interfaces vs Abstract Classes



5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

Source of this image is https://www.javatpoint.com/difference-between-abstract-class-and-interface

Thank You!