PROGRAMMING

Lecture 17

Sushil Paudel

PREVIOUS TOPIC

- Polymorphism
- Type of polymorphism
- Type Casting

TODAY'S TOPIC

- Abstraction
- Abstract Class
- Interface

ABSTRACTION

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- Another way, it shows only important things to the user and hides the internal details
- For example sending SMS, you just type the text and send the message. You don't know the internal processing about the message delivery.
- Abstraction lets you focus on what the object does instead of how it does it.

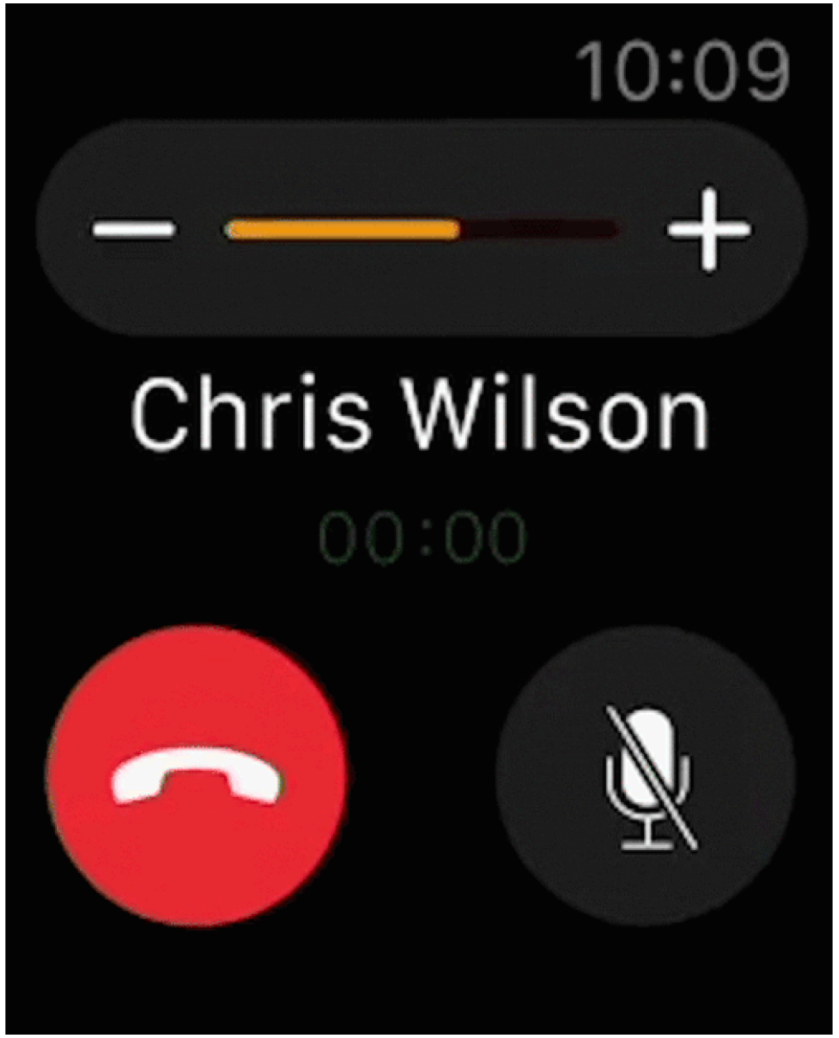
REAL TIME EXAMPLE

- Abstraction shows only important things to the user and hides the internal details.
- For example, when we ride a bike/car, we only know about how to ride bikes/car but can not know about how it work? And also we do not know the internal functionality of a bike/car.

EXAMPLE

- Another real life example of Abstraction is ATM Machine.
- All are performing operations on the ATM machine like cash withdrawal, money transfer, retrieve mini-statement…etc. but we can't know internal details about ATM.

EXAMPLE



EXAMPLE

- If you look at the above gif, you can see when you get a call, we get an option to either pick it up or just reject it.
- But in reality, there is a lot of code that runs in the background.
- So here, you don’t know the internal processing of how a call is generated, that’s the beauty of abstraction.

WAYS TO ACHIEVE ABSTRACTION

Ways to achieve Abstraction

There are two ways to achieve abstraction in java:

1. Abstract class
2. Interface

ABSTRACT METHOD

- Method without body (no implementation) is known as abstract method.
- A method must always be declared in an abstract class, or in other words you can say that if a class has an abstract method, it should be declared abstract as well.
- It is an incomplete method.

SYNTAX

```
abstract void abstractMethod();
```

RULES OF ABSTRACT METHOD

- Abstract methods don't have body, they just have method signature as shown above.
- If a class has an abstract method it should be declared abstract, the vice versa is not true, which means an abstract class doesn't need to have an abstract method compulsory.
- If a regular class extends an abstract class, then the class must have to implement all the abstract methods of abstract parent class or it has to be declared abstract as well.

ABSTRACT CLASS

- An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be sub classed.
- It is an incomplete class.
- *Note:* ***Abstraction*** *is a process of hiding the implementation details and showing only functionality to the user.*

ABSTRACT CLASS

- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (**public void get();**)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

SYNTAX

```
abstract class ClassName {

}
```

EXAMPLE – ABSTRACT CLASS & METHOD

```
abstract class JavaStudent {

	public void learnJava() {
		System.out.println("I am learning Java");
	}

	public abstract void name();
	public abstract void location();
}
```

EXAMPLE – IMPLEMENTED METHOD

```
public class Ram extends JavaStudent {

	@Override
	public void name() {
		System.out.println("My name is Ram");
	}

	@Override
	public void location() {
		System.out.println("Kathmandu");
	}

	public static void main(String[] args) {
		Ram student = new Ram();
		student.learnJava();
		student.name();
		student.location();
	}
}
```

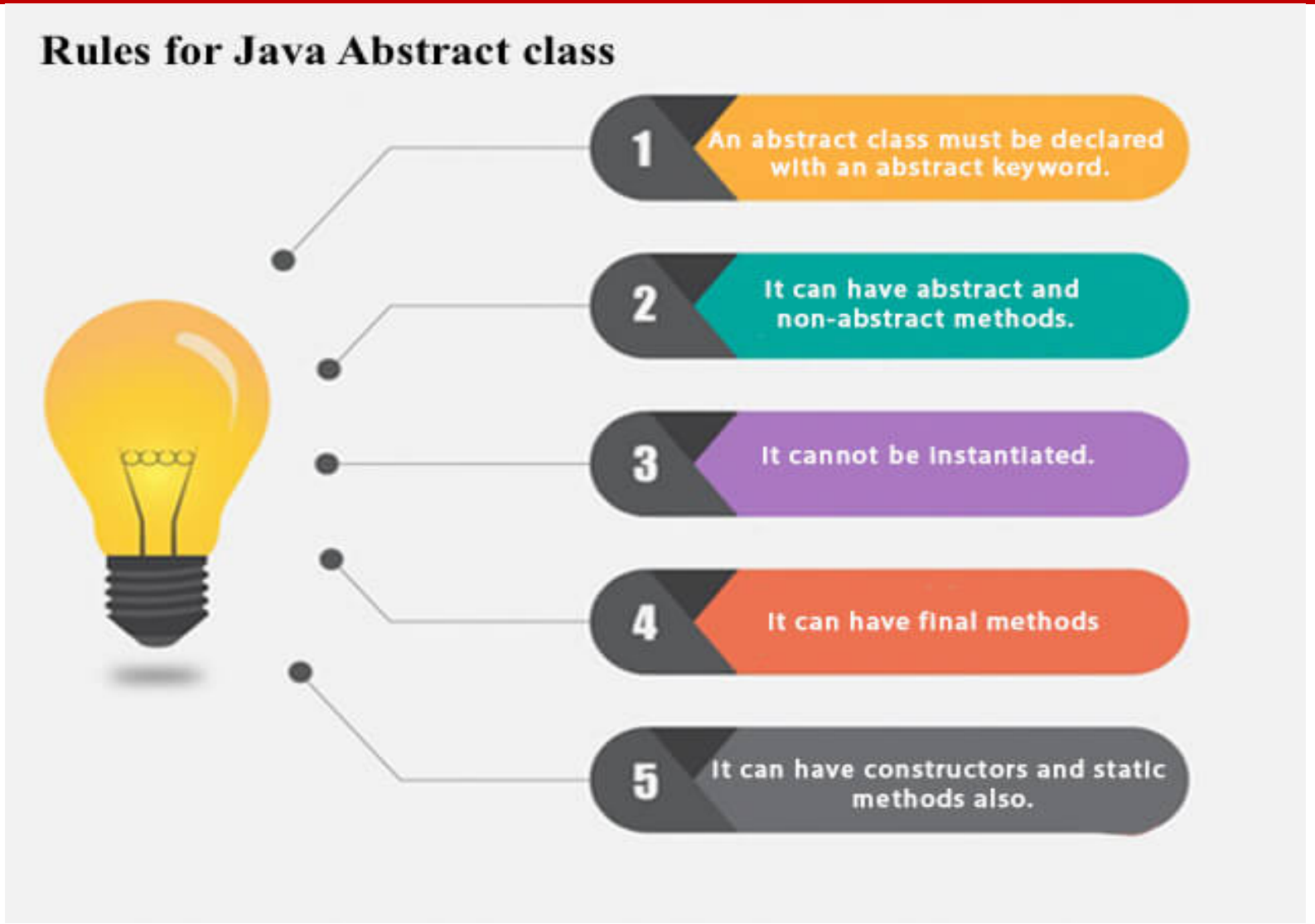
OUTPUT

I am learning Java

My name is Ram

Kathmandu

RULES OF ABSTRACT CLASS



INTERFACE

- An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.
- The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

INTERFACE

- An interface is different from a class in several ways, including –
  - You cannot instantiate an interface.
  - An interface does not contain any constructors.
  - All of the methods in an interface are abstract. (From Java 8, interface can contain default methods)
  - An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
  - An interface is not extended by a class; it is implemented by a class.
  - An interface can extend multiple interfaces.

SYNTAX

```
public interface InterfaceName {

      public void method1();
      public void method2();
}
```

EXAMPLE

```
public interface Chat {

	public void message();
	public void send();
	default void testConnection(){
		System.out.println("Connected.");

	}


}
```

EXAMPLE

```
public class ChatSystem implements Chat{

	@Override
	public void message() {
		System.out.println("Hi, I am learning Java.");
	}

	@Override
	public void send() {
		System.out.println("The message has been sent.");
	}

	public static void main(String[] args) {
		Chat chat = new ChatSystem();
		chat.testConnection ();
		chat.message();
		chat.send();
	}
}
```

OUTPUT

Connected.

Hi, I am learning Java.

The message has been sent.

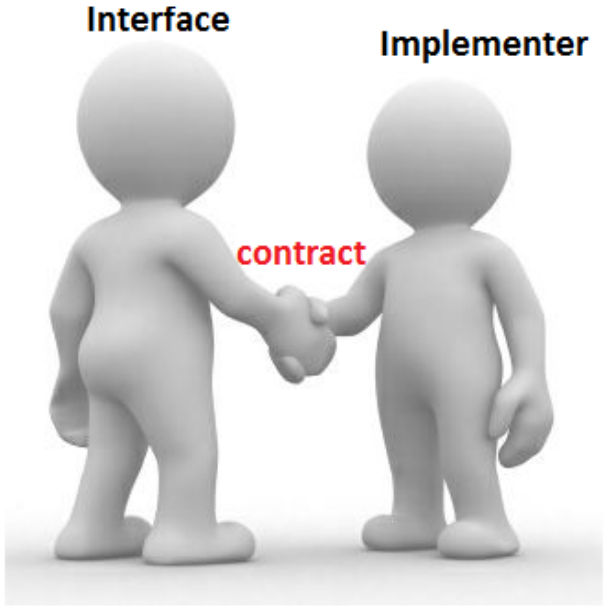
ABSTRACT CLASS VS INTERFACE

Interface

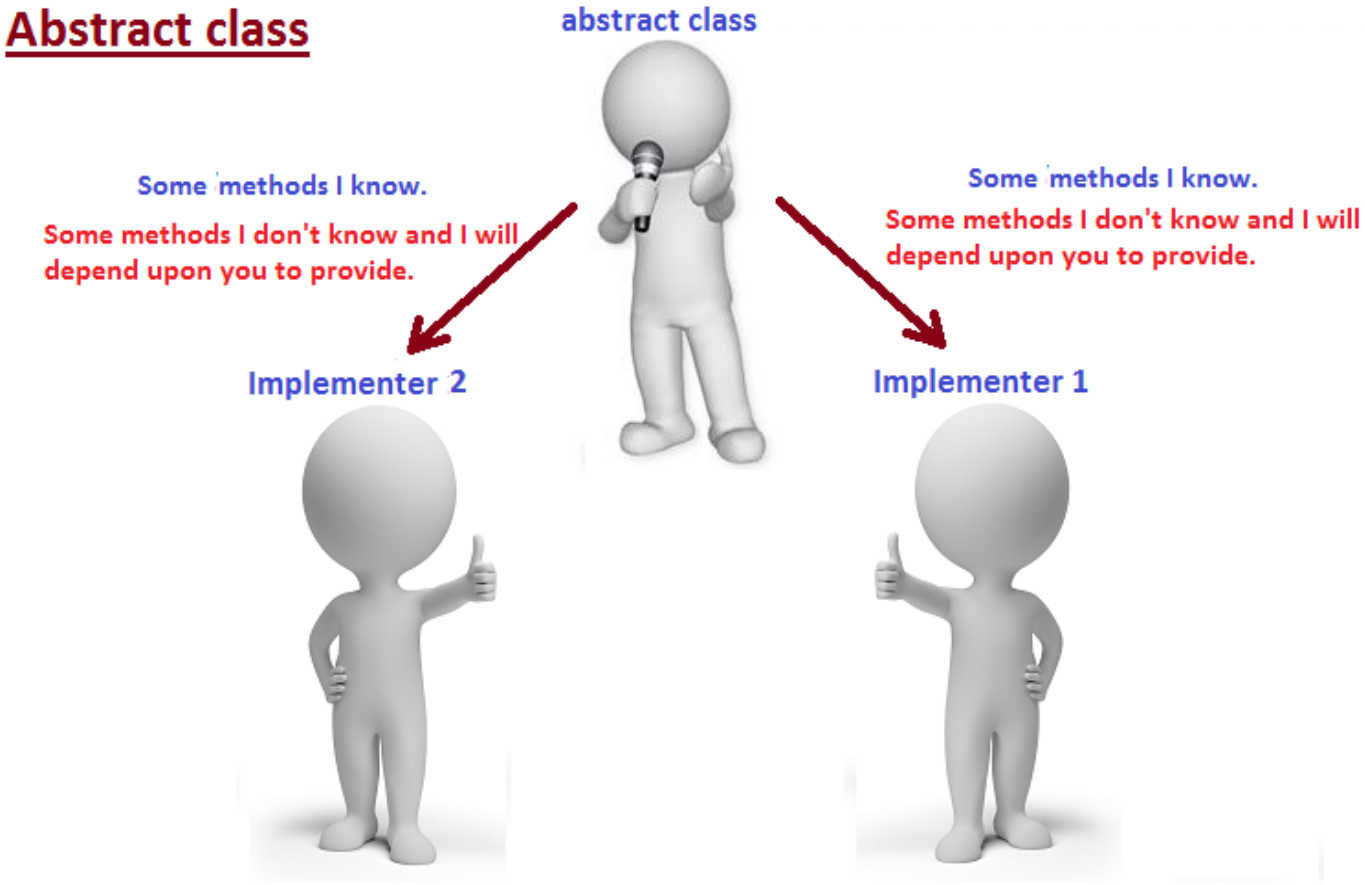
I only know method names that I will require for my job to be done.

You have to provide body for those methods.

Sure, I will definitely provide body to all your methods but in my way.



Abstract class



THANK YOU!

Any questions?