**Abstract class and interface**

In the Animal example of the previous polymorphism session, the Animal class can "eat" and its subclasses can decide what to eat, eg. lion eats meat and cow eats grass ...It's quite simple and clear unless one thing unsatisfactory: The Animal class has the implementaion of eating. For the Animal class, eating is a general sense and we don't need it to implement the eating action. Is there any way to hide the implementaion methods of a parent class and let its subclasses to implement? To realise this goal, we should learn about abstract class and interface.

**Abstract class**

To understand abstract class, we should first get to know abstract methods.

**Abstract methods:**

*Definition: Abstract method is a special method with only declaration and no implementation and is modified by the keyword of abstract.*

*Grammar:*

**abstract** returnType **methodName**();

*For example:*

**abstract** **void** **eat**();

We can see that this method only has method name but doesn't have any specific implementation.

**Abstract class**

*Definition: An abstract class is a class which includes abstract methods. It is also modified by "abstract". And like other classes, it can aso include member variable and general member methods except abstract methods.*

*Grammar:*

**abstract** **class** **ClassName** {

**abstract** returnType **methodName**();

returnType **methodName**(){

*// other method*

}

}

*The example in the previous session can be changed to below:*

**public** **abstract** **class** **Animal** {

**public** **abstract** **void** **eat**();

}

**public** **class** **Cow** **extends** **Animal** {

**@Override**

**public** **void** **eat**() {

System.out.println("I am a vegetarian!");

}

}

**public** **class** **Lion** **extends** **Animal** {

**@Override**

**public** **void** **eat**() {

System.out.println("I am a carnivore!");

}

}

**public** **class** **Test** {

**public** **static** **void** **main**(String[] args) {

Animal cow = **new** Cow();

cow.eat();

Animal lion = **new** Lion();

lion.eat();

}

}

**The differences between an abstract class and a general class are as follows:**

* Abstract class must be public or protected. It is public by default.
* Abstract class cannot be used to create object: there is no sense to create objects for abstract classes.
* If a subclass inherits an abstract class, this subclass should implement the abstract methods of its parent class, If not, this subclass should also be defined as abstract class.

**Interface**

Up to now, we have learnt abstract class and abstract methods and know that they can hide implementation details and only defines actions. But sometimes we may need a set containing only abstract methods and no member variables, or the methods inside are just descriptions instead of implementation. Interface can help us do that.

*Definition: Interface is a set of declarations and the standards on declaration, which consists of many abstract methods.*

**Notes:**

* Interface can be implemented by other classes and declared using "interface".
* Though similar in statements, interface is not equivalent to class. They belong to different concepts. Class describes the attributes and methods of an object, while interface consists of the methods to be implemented by a classs.
* If the class to implement interface is other classes than an abstract class, this class should define all the methods in the interface.
* Interface cannot be instantiated but can be implemented using "implement".

*Grammar:*

**public** **interface** **interfaceName** {

returnType **method**();

...

}

**public** **class** **className** **implement** **iterfaceName** {

**public** returnType **method**() {

*// code*

}

...

}

*Above example on abstract class can be changed to below:*

**public** **interface** **AnimalInterface** {

**void** **eat**();

}

**public** **class** **LionImplement** **implements** **AnimalInterface** {

**@Override**

**public** **void** **eat**() {

System.out.println("I am a carnivore!");

}

}

**public** **class** **Test** {

**public** **static** **void** **main**(String[] args) {

AnimalInterface animalInterface = **new** LionImplement();

animalInterface.eat();

}

}

**The differences between abstract class and interface**

1. The methods in abstract class can have method body, namely specific functions to implement the methods, which doesn't apply to the methods in interface.
2. The member variables in abstract classes are of different types, but in interface they must be the type of public static final.
3. An interface shouldn't have static codes block and static methods (methods modified by static), while abstract classes can have them both.
4. One class can only inherit one abstract class, but can implement multiple interfaces.

**Reference materials:**

1. [Java basics - interface and abstract class](https://www.jianshu.com/p/eb77a2e64fda)
2. [Abstract Methods and Classes](https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html)
3. [Interfaces](https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html)