

JAVA HAS A BETTER WAY TO MAKE CONTRACT BETTER THROUGH ABSTRACT CLASSES AND INTERFACES



"**Abstract class** and **interfaces** are two very vital pivots in the Object Oriented Programming concept."

-Lincoln W. Daniel



Why Abstract Classes and Interface important to be learn?

A regular class defines the fields, implements fully functioning methods and can be instantiated; such a class is referred to as a concrete class. That is the vast majority of classes you will write and use in your programming career. Abstract classes and interfaces cannot be instantiated, but they, too, define fields and methods—although they may not implement methods. They are best for forming a contract between a class, its subclasses, and users of its subclasses.

Abstract Class

An abstract class is much like a regular class in that it can have fields and methods that may or may not have bodies.

```
public abstract class Fighter{
    public String name;
    public double health;

    public Fighter(String name, double health){
        this.name = name;
        this.health = health;
    }
    public abstract void attack(Fighter target);
    public abstract void defend();
    ...
}

interface BowArrow{
    public void shotArrow(Fighter target);
}

interface Sword{
    public void swingSword(Fighter target);
}

interface Magic{
    public void castSpell(Fighter target);
}
```

```
class Archer extends Fighter implements BowArrow{
    public Archer(String name, double health){
        super(name, health);
    }
    public void shotArrow(Fighter target){
        target.health -= 10;
    }
    public attack(Fighter target){
        this.target.health -= 5;
    }
    ...
}

class Swordsman extends Fighter implements Sword{
    ...
    public void swingSword(Fighter target){
        target.health -= 50;
    }
}

class Wizard extends Fighter implements Magic{
    ...
    public void castSpell(Fighter target){
        this.target.health = 0;
    }
}
```

INTERFACE

An interface is a bit different than anything we've seen thus far. A Java interface is more like an abstract class than a regular class. An interface can only contain method signatures and static final fields. An interface is merely a contract between the interface and classes that implement it. Like with abstract classes, classes that implement an interface must implement its methods' bodies to provide functionality. Interfaces are best for creating a contract that will ensure that all classes that implement it behave similarly by abiding by the contract.

In that code, you can see that we have our Fighter class and its **attack()** and **defend()** instance methods are abstract. This is because they are all functions that should be unique to each class that implements them; Archer won't attack the same way as swordsman and wizard, so they should implement the **attack()** method differently.