Interfaces, Abstract Classes, Exception Handling

Abstract Class & Abstract Methods

1. An abstract class is a class that is declared with an abstract keyword.
2. An abstract method is a method that is declared without implementation.
3. An abstract class may or may not have all abstract methods. Some of them can be concrete methods
4. A method-defined abstract must always be redefined in the subclass, thus making overriding compulsory or making the subclass itself abstract.

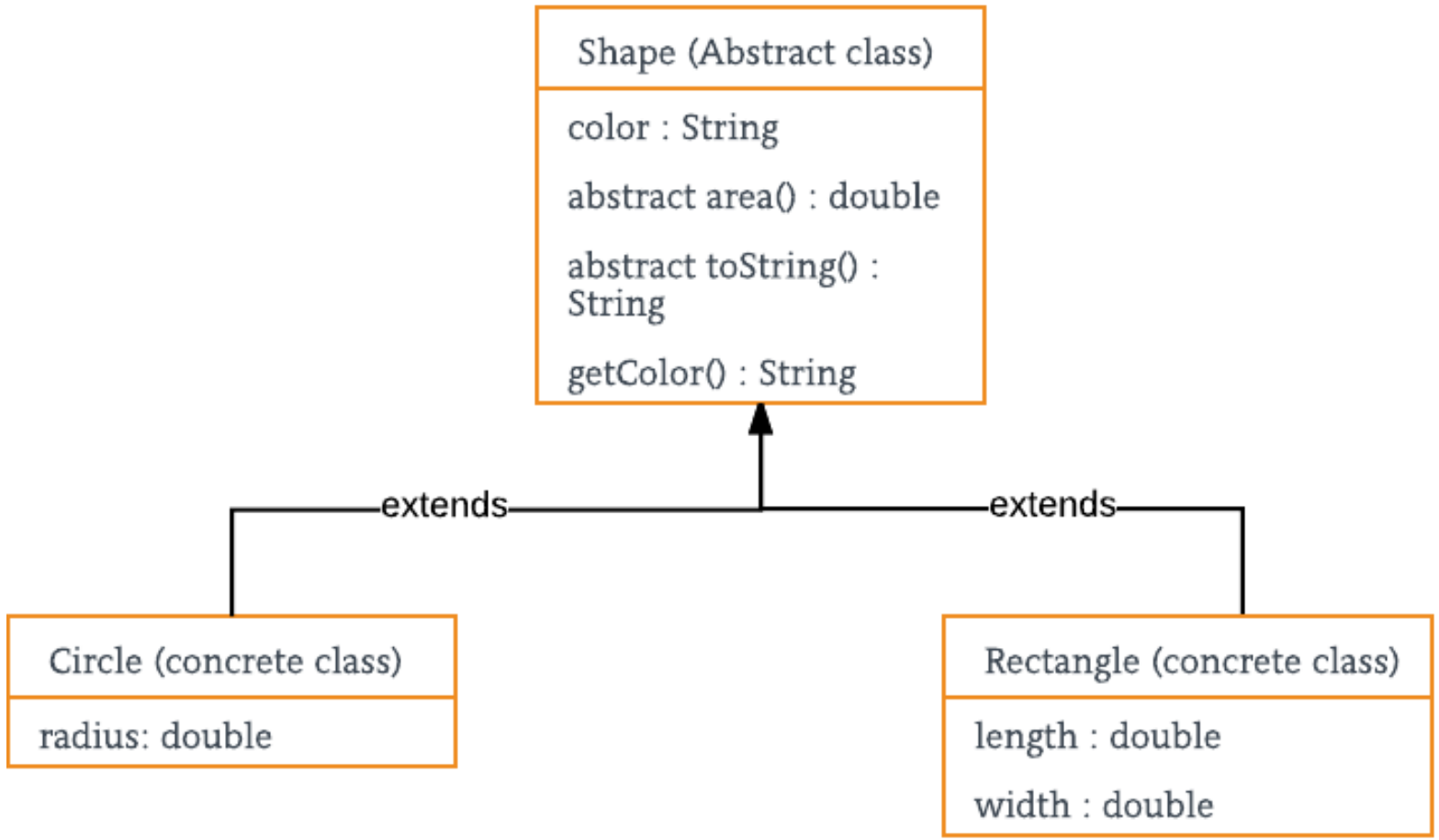
Abstract Class & Abstract Methods

1. Any class that contains one or more abstract methods must also be declared with an abstract keyword.
2. There can be no object of an abstract class. That is, an abstract class can not be directly instantiated with the *new operator*.
3. An abstract class can have parameterized constructors and the default constructor is always present in an abstract class.

When to use

- There are situations in which we will want to define a superclass that declares the structure of a given abstraction without providing a complete implementation of every method.
- Sometimes we will want to create a superclass that only defines a generalization form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.

When to use


Shape (Abstract class)
color : String
abstract area() : double
abstract toString() : String
getColor() : String
extends
extends
Circle (concrete class)
radius: double
Rectangle (concrete class)
length : double
width : double

Abstract Classes in Java

- An instance of an abstract class can not be created.
- Constructors are allowed.
- We can have an abstract class without any abstract method.
- There can be a **final method** in abstract class
  - But, any abstract method in class(abstract class) can not be declared as final.

Abstract Classes in Java

- We can define static methods in an abstract class
- We can use the **abstract keyword** for declaring ***top-level classes (Outer class) as well as inner classes*** as abstract
- If a **class** contains at least **one abstract method** then compulsory should declare a class as abstract
- If the **Child class** is unable to provide implementation to all abstract methods of the **Parent class** then we should declare that **Child class as abstract** so that the next level Child class should provide implementation to the remaining abstract method

Inheritance - Interfaces

- Java allowed only single inheritance
- What if you want to inherit behavior from more than one

Interfaces

- An **Interface in Java** programming language is defined as an abstract type used to specify the behavior of a class.
- An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods.
- Java Interface also **represents the IS-A relationship**.

Interfaces

- Interfaces specify what a class must do and not how. It is the blueprint of the class.
- An Interface is about capabilities that a class implementing must be able to do.
  - It specifies a set of methods that the class has to implement.

```
interface {

        // declare constant fields
        // declare methods that are
abstract
}
```

Abstract Class & Interfaces

- **Inheritance vs Abstraction:** A Java interface can be implemented using the keyword “implements” and an abstract class can be extended using the keyword “extends”.
- **Multiple implementations:** An interface can extend one or more Java interfaces; an abstract class can extend another Java class and implement multiple Java interfaces.
- **Multiple Inheritance:** Interface supports multiple inheritance; an abstraction does not support multiple inheritance.
- **Accessibility of Data Members:** Members of a Java interface are public by default. A Java abstract class can have class members like private, protected, etc.

Abstract Class & Interfaces

- **Type of methods:** Interface can have only abstract methods. An abstract class can have abstract and non-abstract methods. From Java 8, it can have default and static methods also. From Java 9, it can have private concrete methods as well.
- **Final Variables:** Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.
- **Type of variables:** Abstract class can have final, non-final, static and non-static variables. The interface has only static and final variables.
- **Implementation:** Abstract class can provide the implementation of the interface. Interface can’t provide the implementation of an abstract class.

Programming

- Create Examples in Java about Interfaces and Abstract Class to show
  - Abstract class relationships
    - Child class implementing methods or not
      - Can a child class only override abstract methods?
      - What about final methods?
      - Can they access variables of abstract class?
  - Classes implementing interfaces
    - How many interfaces can a class implement?
    - Does a class need to override all methods of an interface for implementing it?
  - Abstract class implementing interfaces
  - Enabling Multiple Inheritance using Abstract class and interfaces

Reference

- Interfaces in Java - GeeksforGeeks