Object Oriented Analysis and Design with Java

UE19CS353

Prof. Sindhu R Pai
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

UE19CS353: Object Oriented Analysis and Design with Java

Single Rooted Hierarchy, Abstract class and interface

Prof. Sindhu R Pai

Department of Computer Science and Engineering

Agenda

- Single rooted Hierarchy
- Abstract class
- Interface
- Why use interface?
- Abstract class vs interface
- Points to think!
- References

Single rooted Hierarchy

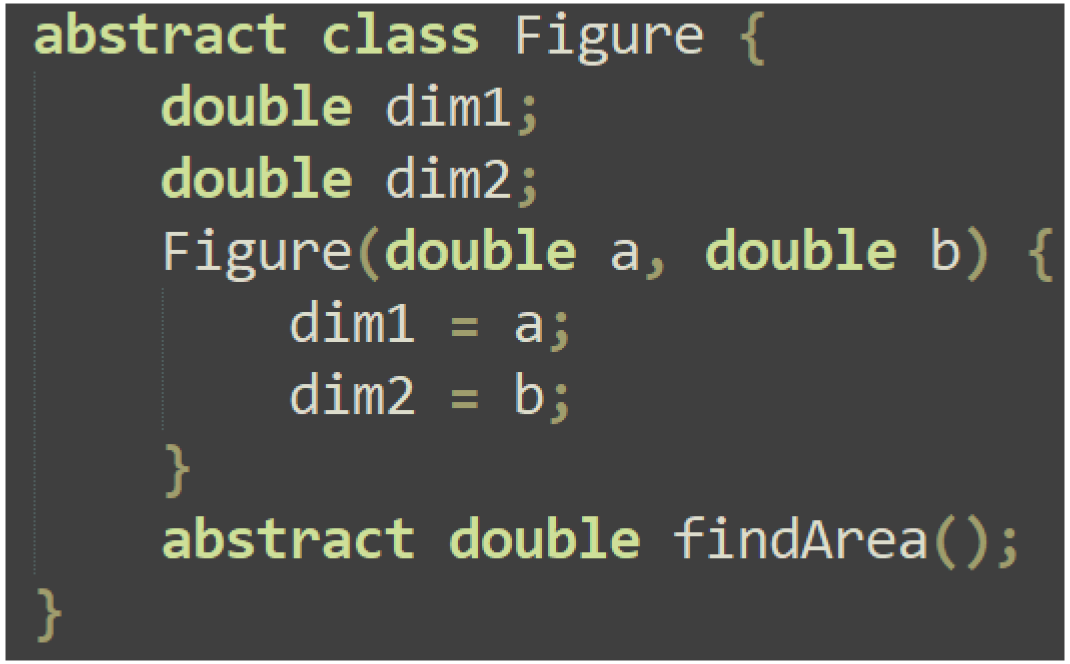
- Characteristic of most OOP languages
- About a common interface all objects must implement
- All classes inherit directly or indirectly from a single root
- Name of this ultimate base class/root is simply **Object**
- All objects in a singly rooted hierarchy can be guaranteed to have certain functionality.
- Greatly simplifies argument passing
- Easier to implement a garbage collector - Required implementation is provided in the base class enabling to send messages to every object.
- Enables platform developer to have some minimum knowledge about all objects which simplifies development of other libraries which can be used on all other objects.

Abstract class

- Refer to L10 slides for explanation
-

Abstract class

Coding Example: If the abstract class contains the below data, how to implement Rectangle and Triangle classes?



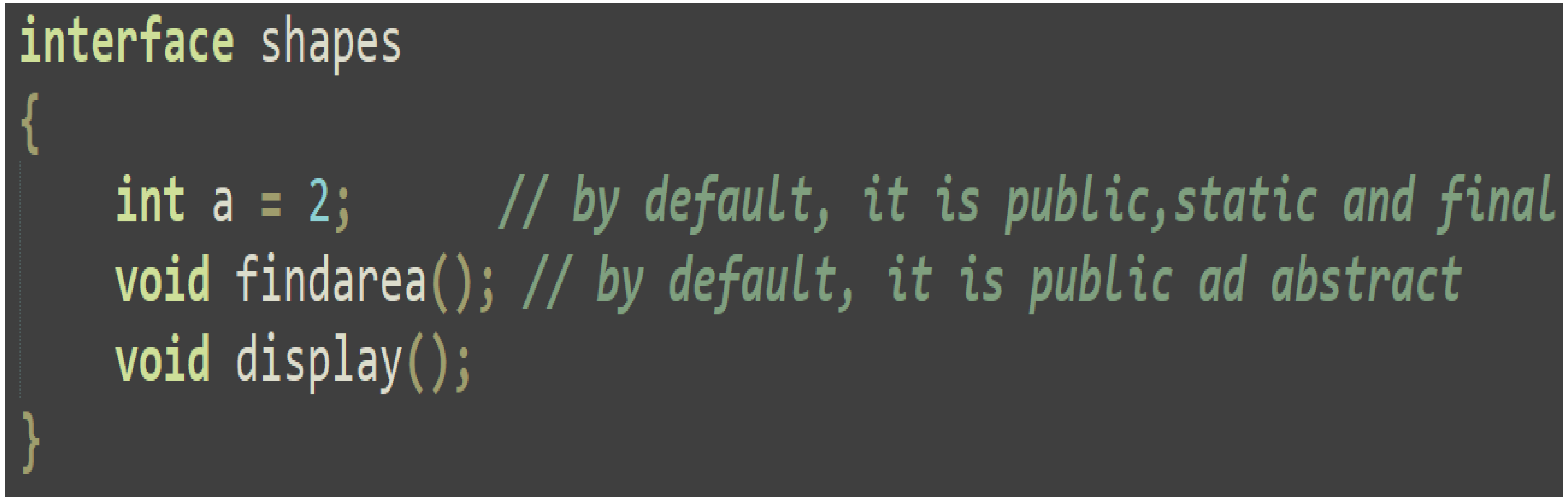
```
abstract class Figure {
    double dim1;
    double dim2;
    Figure(double a, double b) {
        dim1 = a;
        dim2 = b;
    }
    abstract double findArea();
}
```

```
class Rectangle extends Figure {
                ??
        }

class Triangle  extends Figure {
                ? ?
        }
```

Interface

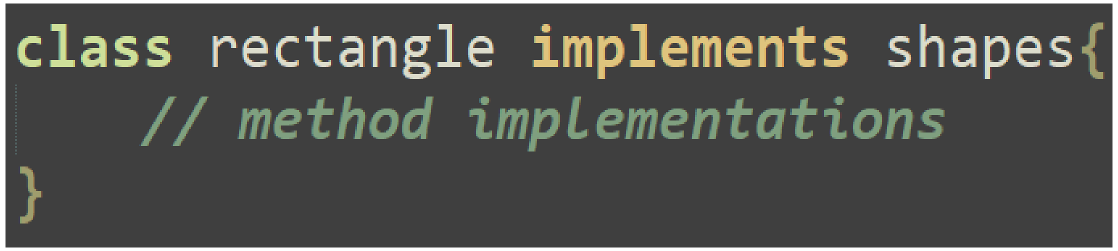
- A description of **what** actions that an object can do and **not how to** do
- An abstract type that is used to specify a behavior that classes must implement
- They are declared using the **interface keyword**, and **may only contain method signatures(public and abstract) and constant declarations** (public, **static and final**).



```
interface shapes
{
    int a = 2;      // by default, it is public,static and final
    void findarea(); // by default, it is public ad abstract
    void display();
}
```

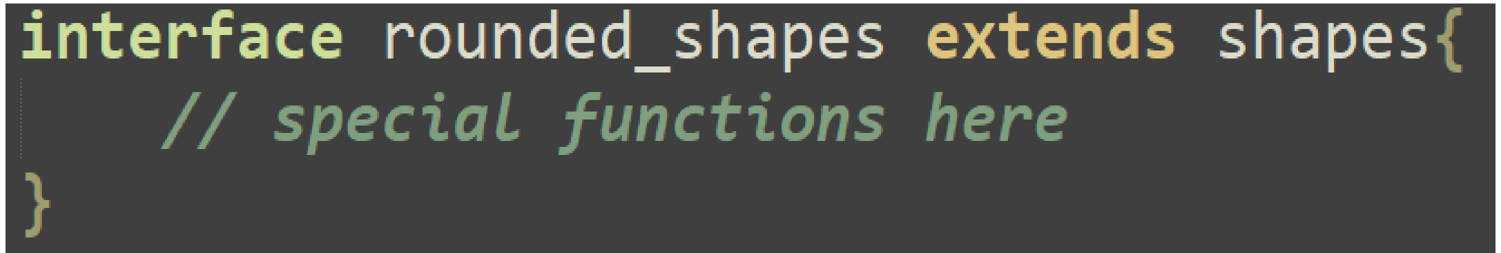
Interface contd..

- Interface specifies the method signatures which has no default implementation
- Interface can contain any number of methods.
- Name of the interface must match the name of the file and the byte code of an interface appears in a .class file.
- An interface is not extended by the class. It is **implemented by a class**. Class **can implement more than one interface**



```
class rectangle implements shapes{
    // method implementations
}
```

- An interface can extend multiple interfaces

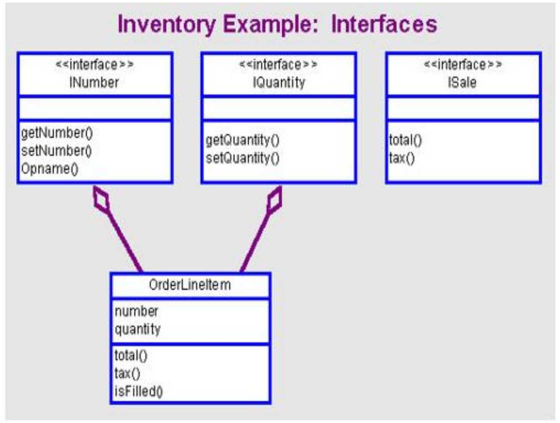


```
interface rounded_shapes extends shapes{
    // special functions here
}
```

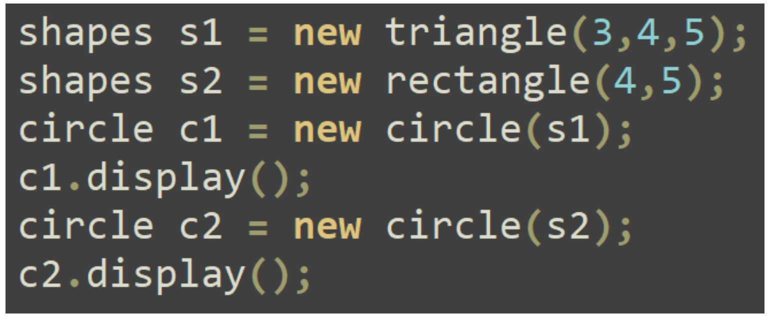
Why use Interface?

- Used to achieve total abstraction

- Using interface java can achieve multiple inheritance



- Used to achieve loose coupling



```
shapes s1 = new triangle(3,4,5);
shapes s2 = new rectangle(4,5);
circle c1 = new circle(s1);
c1.display();
circle c2 = new circle(s2);
c2.display();
```

Abstract class Vs Interface

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.
Abstract class **doesn't support multiple inheritance.**	Interface **supports multiple inheritance**.
Abstract class can have **final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
Abstract class **can provide the implementation of interface.**	Interface **can't provide the implementation of abstract class.**
The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
An abstract class **can extend another Java class and implement multiple Java interfaces.**	An interface **can extend another Java interface only**.
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.**

Note: 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.

Points to think!!

- Can an abstract class have method implementations?   Can an abstract class have fields?
- Can an abstract class have constructors?
- How does instanceof work if we have a number of classes in linear inheritance
- With inheritance, we will be able to override the methods of the base class so that meaningful implementation of the base class method can be designed in the derived class.
- Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes
- Can we implement a class called MyString which always stores the strings in case insensitive manner?
- Can I extend the String class?
- How do we make classes which are not instantiable? Not inheritable?

References

- Singly rooted hierarchy – Wikipedia
- Thinking in Java 1: Introduction to Objects - The singly rooted hierarchy (linuxtopia.org)
- Java sqrt() method with Examples – GeeksforGeeks
- Difference between Abstract class and Interface - Javatpoint

THANK YOU

Prof. Sindhu R Pai

Department of Computer Science and Engineering

sindhurpai@pes.edu

+91 8277606459