SNS COLLEGE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
COIMBATORE – 35
DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

UNIT III – INHERITANCE AND POLYMORPHISM

Inheritance
Super class, Sub class
Types of inheritance
Method Overloading
Method Overriding
Abstract class
this keyword, Final keyword
Packages
Interfaces

Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java.

It can have abstract and non-abstract methods (method with the body).

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you 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.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Abstract class in Java

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Rules for Java Abstract class

1 An abstract class must be declared with an abstract keyword.

2 It can have abstract and non-abstract methods.

3 It cannot be instantiated.

4 It can have final methods

5 It can have constructors and static methods also.
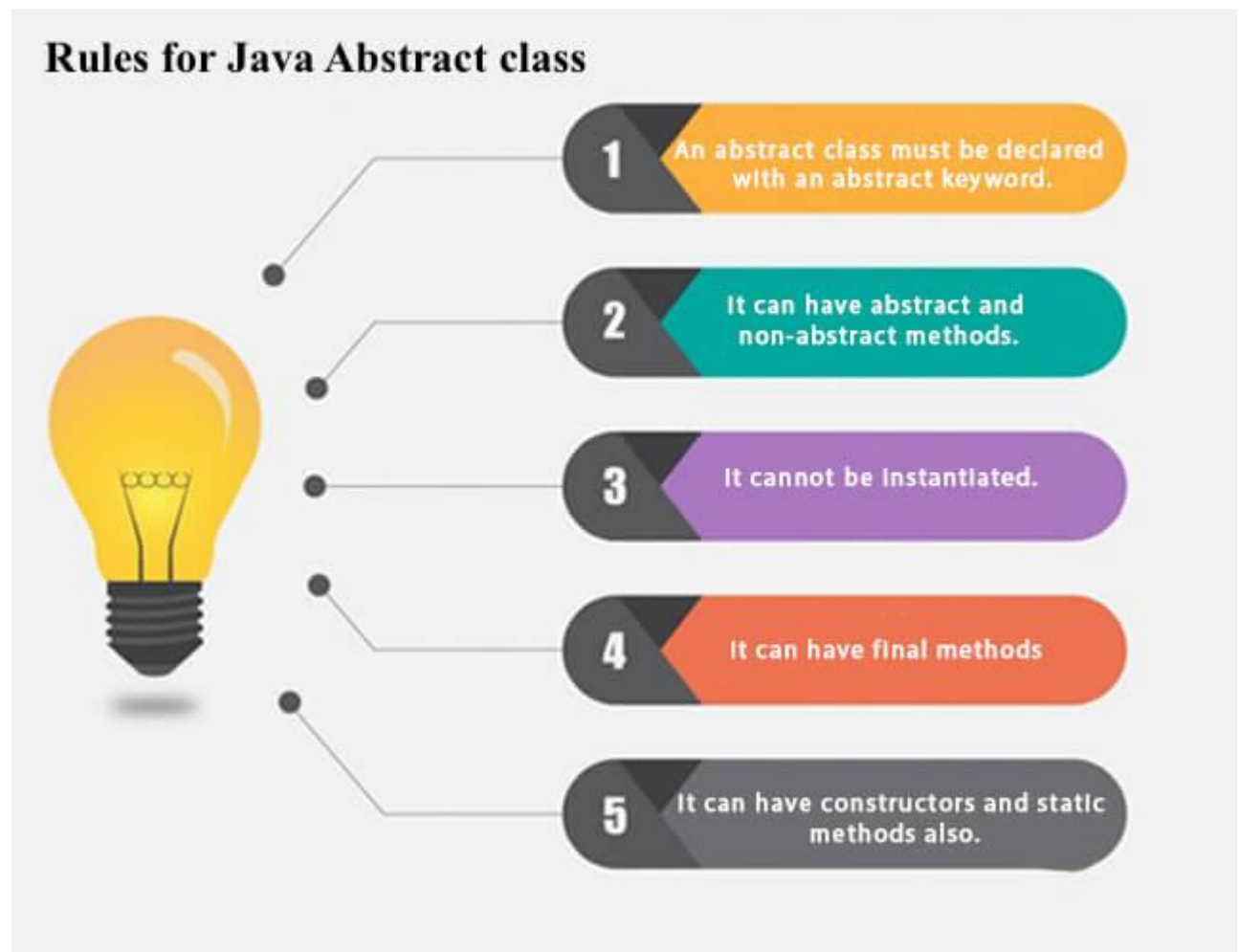
Example of abstract class

1. abstract class A{}

Abstract Method in Java

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method

1. abstract void printStatus();//no method body and abstract

Example of Abstract class that has an abstract method

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
1.  abstract class Bike
2.  {
3.   abstract void run();
4.  }
5.  class Honda4 extends Bike
6.  {
7.  void run()
8.  {System.out.println("running safely");}
9.
10. public static void main(String args[])
11. {
12.  Bike obj = new Honda4();
13.  obj.run();
14. }
15. }
```

```
running safely
```

Understanding the real scenario of Abstract class

In this example, Shape is the abstract class, and its implementation is provided by the Rectangle and Circle classes.

Mostly, we don't know about the implementation class (which is hidden to the end user), and an object of the implementation class is provided by the **factory method**.

A **factory method** is a method that returns the instance of the class. We will learn about the factory method later.

In this example, if you create the instance of Rectangle class, draw() method of Rectangle class will be invoked.

File: TestAbstraction1.java

```
1.   abstract class Shape
2.   {
3.   abstract void draw();
4.   }
5.
6.   //In real scenario, implementation is provided by others i.e. unknown by end user
7.   class Rectangle extends Shape
8.   {
9.   void draw()
10. {System.out.println("drawing rectangle");}
11. }
12.
13. class Circle1 extends Shape
14. {
15. void draw()
16. {System.out.println("drawing circle");}
17. }
18. //In real scenario, method is called by programmer or user
19. class TestAbstraction1
20. {
21. public static void main(String args[])
22. {
23. Shape s=new Circle1();//In a real scenario, object is provided through method, e.g., getShape() method
24. s.draw();
25. }
26. }
```

```
drawing circle
```

Another example of Abstract class in java

File: TestBank.java

```
1.   abstract class Bank
2.   {
3.   abstract int getRateOfInterest();
4.   }
5.   class SBI extends Bank
6.   {
7.   int getRateOfInterest(){return 7;}
8.   }
9.   class PNB extends Bank
10.  {
11.  int getRateOfInterest(){return 8;}
12.  }
13.
14.  class TestBank
15.  {
16.  public static void main(String args[])
17.  {
18.  Bank b;
19.  b=new SBI();
20.  System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
21.  b=new PNB();
22.  System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
23.  }
24.  }
```

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

Abstract class having constructor, data member and methods

An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

File: TestAbstraction2.java

```
1.  //Example of an abstract class that has abstract and non-abstract methods
2.  abstract class Bike
3.  {
4.    Bike()
5.  {System.out.println("bike is created");}
6.
7.    abstract void run();
8.    void changeGear()
9.  {System.out.println("gear changed");}
10. }
11.
12. //Creating a Child class which inherits Abstract class
13.  class Honda extends Bike
14. {
15.  void run()
16. {System.out.println("running safely..");}
17. }
18. //Creating a Test class which calls abstract and non-abstract methods
19.  class TestAbstraction2
20. {
21.  public static void main(String args[])
22. {
23.   Bike obj = new Honda();
24.   obj.run();
25.   obj.changeGear();
26. }
27. }
```

```
bike is created
running safely..
gear changed
```

Rule: If there is an abstract method in a class, that class must be abstract.

1. class Bike12{
2. abstract void run();
3. }

```
compile time error
```

Rule: If you are extending an abstract class that has an abstract method, you must either provide the implementation of the method or make this class abstract.

Another real scenario of abstract class

The abstract class can also be used to provide some implementation of the interface. In such case, the end user may not be forced to override all the methods of the interface.

Note: If you are beginner to java, learn interface first and skip this example.

1. interface A
2. {
3. void a();
4. void b();
5. void c();
6. void d();
7. }
8.
9. abstract class B implements A
10. {
11. public void c()
12. {System.out.println("I am c");}
13. }
14.
15. class M extends B
16. {
17. public void a(){System.out.println("I am a");}
18. public void b(){System.out.println("I am b");}
19. public void d(){System.out.println("I am d");}
20. }
21.
22. class Test5
23. {
24. public static void main(String args[])
25. {
26. A a=new M();

```
27. a.a();
28. a.b();
29. a.c();
30. a.d();
31. }}
```

```
Output:I am a
       I am b
       I am c
       I am d
```