Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

Sairam INSTITUTIONS

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to EEE, CSE, EIE, ICE, IT)

UNIT NO 2

INHERITANCE AND INTERFACES

2.3 The Object class – abstract classes and methods

COMPUTER SCIENCE & ENGINEERING

4 QUALITY EDUCATION

NBA NATIONAL BOARD OF ACCREDITATION
NAAC A+
nirf NATIONAL INSTITUTIONAL RANKING FRAMEWORK
BUREAU VERITAS Certification

OBJECT CLASS

A special class, Object defined by Java. All other classes are subclasses of Object. That is, Object is a super class of all other classes.

This means that a reference variable of type Object can refer to an object of any other class.

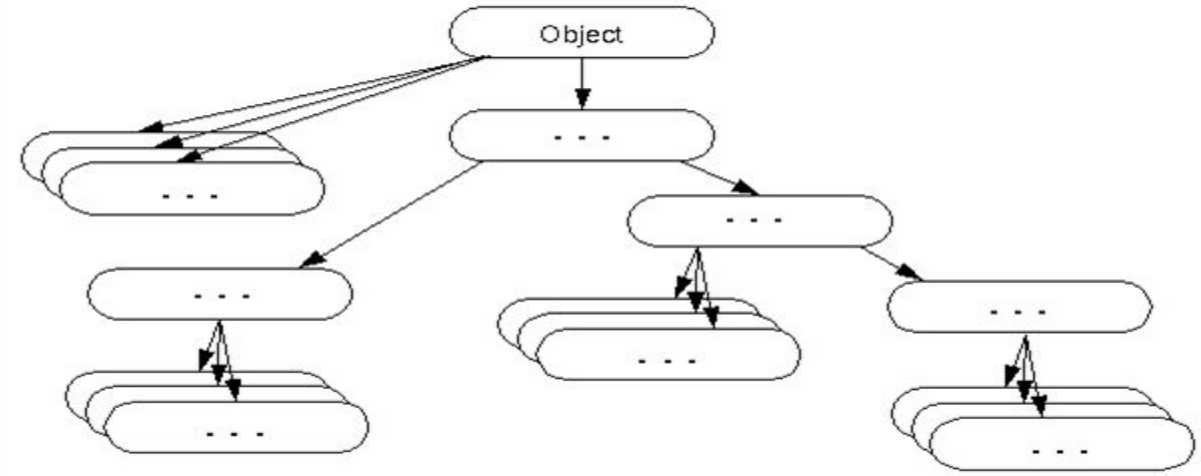
Since arrays are implemented as classes, a variable of type Object can also refer to any array.

Object defines the following methods, which means that they are available in every object.

OBJECT CLASS

•The **Object class** is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

•The Object class is beneficial to refer any object whose type is not known. Notice that parent class reference variable can refer the child class object, know as upcasting.



OBJECT CLASS

The methods getClass( ), notify( ), notifyAll( ), and wait( ) are declared as final.

The equals( ) method compares two objects. It returns true if the objects are equal, and false otherwise. The precise definition of equality can vary, depending on type of objects being compared.

The toString( ) method returns a string that contains a description of the object on which it is called.This method is automatically called when an object is output using println( ).

Many classes override this method. Doing so allows them to tailor a description specifically for the types of objects that they create.

OBJECT CLASS

Method	Description
public final Class getClass()	returns the Class class object of this object. The Class class can further be used to get the metadata of this class.
public int hashCode()	returns the hashcode number for this object.
public boolean equals(Object obj)	compares the given object to this object.
protected Object clone() throws CloneNotSupportedException	creates and returns the exact copy (clone) of this object.
public final void notify()	wakes up single thread, waiting on this object's monitor.
public final void notifyAll()	wakes up all the threads, waiting on this object's monitor.
public final void wait(long timeout)throws InterruptedException	causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
protected void finalize()throws Throwable	is invoked by the garbage collector before object is being garbage collected.

EXAMPLE PROGRAM

```
//Java program to demonstrate working of getClass()
public class Test
{      public static void main(String[] args)
   {
       Object obj = new String("Sairam");
       Class c = obj.getClass();
       System.out.println("Class of Object obj is : "+ c.getName());
   }
}
```

Output:

Class of Object obj is : java.lang.String

ABSTRACT CLASS AND METHOD

A class which contains the **abstract** keyword in its declaration is known as abstract class.

Abstract classes may or may not contain abstract methods, i.e., methods without body (public void get();)

But, if a class has at least one abstract method, then class **must** be declared abstract. If a class is declared abstract, it cannot be instantiated.

To use an abstract class, we have to inherit it from another class, provide implementations to the abstract methods in it.

If it was inherited we have to provide implementations to all the abstract methods in it.

Syntax:

```
abstract returntype functionname ();    //No definition
```

ABSTRACT CLASS AND METHOD

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

The method body will be defined by its subclass.

Abstract method can never be final and static. Any class that extends an abstract class must implement all the abstract methods declared by the super class.

Note:

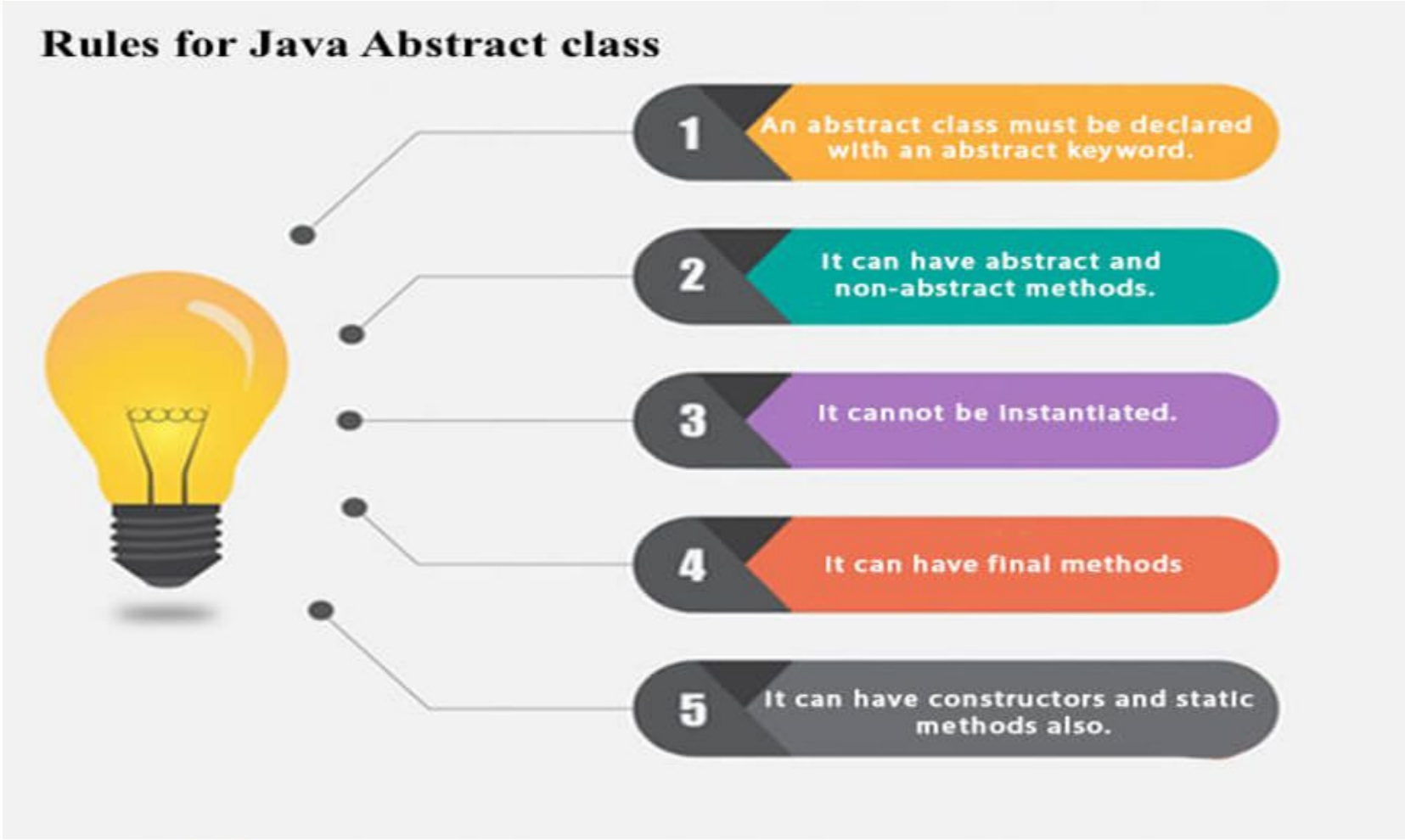
A normal class (non-abstract class) cannot have abstract methods.

ABSTRACT CLASS AND METHOD

Syntax:

```
modifier abstract class classname
{
        //declare fields
       //declare methods
       abstract dataType methodName();
}
modifier class childClass extends className
{
       datatype methodName()
       {
       }
}
```

ABSTRACT CLASS



ABSTRACT CLASS

Rules

1.Abstract classes are not Interfaces.

2.An abstract class may have concrete (complete) methods.

3.An abstract class may or may not have an abstract method. But if any class has one or more abstract methods, it must be compulsorily labeled abstract.

4.Abstract classes can have Constructors, Member variables and Normal methods.

5.Abstract classes are never instantiated.

ABSTRACT CLASS

Rules

6.Reference of an abstract class can point to objects of its sub-classes thereby achieving run-time polymorphism.

7.For design purpose, a class can be declared abstract even if it does not contain any abstract methods.

8.A class derived from the abstract class must implement all those methods that are declared as abstract in the parent class.

9.If a child does not implement all the abstract methods of abstract parent class, then he child class must need to be declared abstract as well.

Sample Program

Filename: TestAbstraction1.java

```
abstract class Shape
{
abstract void draw();
}
// implementation is provided by user
class Rectangle extends Shape
{
void draw()
{
System.out.println("drawirectangle");
}
}
class Circle1 extends Shape
{
void draw()
{
System.out.println("draw circle");
}
}
// method is called by user
class TestAbstraction1
{
public static void main(String ar[])
{
Shape s=new Circle1();
//object provided through method
s.draw();
}
}
```

Output:

```
draw circle
```

Sample program for abstract class with normal method

Abstract classes can also have normal methods with definitions, along with abstract methods.

```
abstract class A
        {
                abstract void callme();
                public void normal()
                {
                System.out.println("this is an normal method.");
                }
        }

        public class B extends A
        {
        void callme()
        {
        System.out.println("this is an abstract method.");
        }
```

Sample program for abstract class with normal method

```
public static void main(String[] args)
        {
                B b = new B();
                b.callme();
                b.normal();
        }
        }
```

Output:

this is an abstract method.

this is an normal method

Program for abstract class without abstract method

```
abstract class Base
        {       void fun()
                {
                        System.out.println("Within Base fun()");
                }
        }
class Derived extends Base
        {
        }
public class sample
        {
        public static void main(String args[])
                {
                        Derived d = new Derived();
                        d.fun();
                }
        }
```

Output:
Within Base fun()

Program for abstract class with final method

```
abstract class Base
        {       final void fun()
                {
                System.out.println("Within Derived fun()");
                }
        }
class Derived extends Base
        {
        }
public class sample
        {
        public static void main(String args[])
                {
                        Base b = new Derived ();
                        b.fun();
                }
        }
```

Output:

Within Derived fun()

Difference between Abstract class and interface

Abstract class	Interface
1) Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. it can have default and static methods also.
2) Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
3) Abstract class can have final, non-final, static and non-static variables.	Interface has only static and final variables.

Difference between Abstract class and interface

Abstract class	Interface
4) An abstract class can extend another Java class and implement multiple Java interfaces.	An interface can extend another interface only.
5) An abstract class can be extended using keyword "extends".	An interface can be implemented using keyword "implements".
6) Abstract class can provide the implementation of interface.	Interface can't provide the implementation of abstract class.
7) The abstract keyword is used to declare abstract class.	The interface keyword is used to declare interface.

Difference between Abstract class and interface

Abstract class	Interface
8) A Java abstract class can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)Example:
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

VIDEO LINKS

- https://www.youtube.com/watch?v=p_4Dyfplqkw&t=424s
- https://www.youtube.com/watch?v=juGJLW1_fXU