Abstract class , abstract method using Array concept :

```
package com.ravi.asbatrct;

abstract class Animal
{
    protected String name;

    public Animal(String name)
    {
        this.name = name;
    }

    public abstract void checkup();
}
class Lion extends Animal
{
    public Lion(String name)
    {
        super(name);
    }

    @Override
    public void checkup()
    {
        System.out.println(name+ " Lion is going for Regular checkup...");
    }
}
class Dog extends Animal
{
    public Dog(String name)
    {
        super(name);
    }

    @Override
    public void checkup()
    {
        System.out.println(name+ " Dog is going for Regular checkup...");
    }

}

public class AbstractBusinessRule {

    public static void main(String[] args)
    {
        //Array in java
        Lion lions[] = new Lion[3];
        lions[0] = new Lion("Simba");
        lions[1] = new Lion("Pasa");
        lions[2] = new Lion("Congo");


        Dog dogs[] = new Dog[2];
        dogs[0] = new Dog("Tommy");
        dogs[1] = new Dog("Tiger");

        animalCheckup(lions);
        System.out.println(".........");
        animalCheckup(dogs);


    }


    public static void animalCheckup(Animal ...animals)
    {
        for(Animal animal : animals)
        {
            animal.checkup();
        }
    }

}
```

WAP to show oevrriding abstract method is compulsory :

```
package com.ravi.asbatrct;

abstract class Alpha
{
    public abstract void demo();
    public abstract void show();
}

abstract class Beta extends Alpha
{
    @Override
    public void demo() // + show();
    {
      System.out.println("Demo method is overridden in Beta class");
    }
}
class Gamma extends Beta
{
    @Override
    public void show()
    {
        System.out.println("Show method is overridden in Gamma class");
    }
}

public class AbstractDemo
{
    public static void main(String[] args)
    {
        Gamma g = new Gamma();
        g.demo();
        g.show();
    }
}
```

Anonymous Inner class in Java :

Outer class Concept :

```
class Outer  //Outer.class
{
    protected class Inner1   //Outer$Inner1.class
    {
    }

    private class Inner2  //Outer$Inner2.class
    {
    }

}

public class IQ  //IQ.class
{
    public static void main(String[] args)
    {

    }
}
```

Defination of Anonymous inner class :

----

* If we decalre a class **without any name inside a method body** then it is called Anonymous inner class.

* Anonymous inner class always ends with terminator (;)

* An Anonymous inner class .class file will be represented by $ symbol, Here we don't have class name so compiler will provide numbers like $1.class, $2.class and so on.

* THE MAIN PURPOSE OF ANONYMOUS INNER CLASS TO EXTEND A CLASS OR IMPLEMENT AN INTERFACE THAT MEANS TO **CREATE A SUB TYPE.**

* Anonymous inner class body declaration and object creation by using new keyword, both are done in the same line i.e At the time of declaration of anonymous inner class.

//Programs :

```
package com.ravi.anonymous_inner_class;

class Super
{
    public void show()
    {
        System.out.println("Super class show method!!!");
    }
}

public class AnonymousDemo
{
    public static void main(String[] args)
    {
        //Anonymous Inner class
        Super sub = new Super()
        {
            @Override
            public void show()
            {
                System.out.println("Sub class show method!!!");
            }
        };

        sub.show();
    }

}
```

```
package com.ravi.anonymous_inner_class;

abstract class Vehicle
{
    public abstract void run();
}

public class AnonymousDemo1
{
    public static void main(String[] args)
    {
        Vehicle car = new Vehicle()
        {
            @Override
            public void run()
            {
                System.out.println("Car is Running");
            }
        };

        car.run();


        Vehicle bike = new Vehicle()
        {
            @Override
            public void run()
            {
                System.out.println("Bike is Running");
            }
        };

        bike.run();
    }

}
```

interface :