IT-22056

Hossain Mohammad Abujar

Answer to the ques: NO-1:-

In zoo different animal do different things. but all animals can make sounds. some animals like doges and cants ar mammals and also have some common behaviors like breathing and sleeping But for others like Reptiles and Amphibians do not breath and sleep like mammals.

When concerned about the ability to make sound for any animal we will use Interface.
But when concerned about animals share common behavior then we will use Abstract class.

For scenario 1: (Using Interface)

```
Interface Animal {
    void makeSound();
}
class Dog implements Animal {
public void makesound()
{
    System.out.println("Dog barks: Woof!");}}
```

```
class Bird implements Animal {
    public void makeSound()
    { System.out.println ("Birds chirps: Tweet! ");
    }
public class AnimalInterface Example{
    public static void main (String [] args)
    { Animal a1 = new Dog();
      Animal a2 = new Bird();
      a1.makeSound();
      a2.makeSound();
    }
```

Scenario -2: (Using Abstract class)

```
abstract class Mammal {
        void sleep(){
            Sout ("sleeping peacefully...");}.
    abstract void makeSound();}.
class cat extends Mammal {
        public void make sound(){
        Sout ("Cat meows : Meow! ");
    }.
```

```
class Dog extends Mammals {
    public void makeSound() {
        Sout("Dog barks: Woof!");
    }
}
public class AnimalAbstractExample {
    public static void main(String[] args) {
        Mamal m1 = new Cat();
        Mamal m2 = new Dog();
        m1.makeSound();
        m1.sleep();
        m2.makeSound();
        m2.sleep();
    }
}
```

Answer to the question No- 2:-

Is it true that invoking methods in interface are slower than Invoking it within the abstract classes?

⇒. Calling a method from an interface can be slightly slower than from an abstract class. because the JVM uses a different lookup mechanism. Interfaces are an interface table while abstract clause are a virtual table. The vtable is usually faster to access. However modern JVMs optimize both types of calls very well. In real world programs the speed difference is so small that it doesn't matter.

Example:

```
Interface MyInterface {
    void show();}
abstract class MyAbstract {
    abstract void show();}
Class InterfaceImpl implements My_Interface {
    public void show() {
    }}
Class AbstractImpl extends MyAbstract {
public void show(){}}
```

```
public class Interface_vs_abstractSpeed {
  public static void main (String [] args)
  { Myinterface obj1 = new InterfaceImpl();
    //          obj2 = new AbstractImpl();
   long start, end;
    start = System.nanoTime ();
    for (int i = 0; i< 1000000 ; i++){
      obj1.show(); }
     end = System.nanoTime();
    System.out.println ("Interface method time: "+
    (end-start) + " ns ");
    start = System.nanoTime();
     for (int i=0; i<1000000 ; i++)
     { obj2.show();
      }
    end = System.nanotime();
     System.out.println (" Abstract class method
    time : "+ (end -start) + "ns ");
}
```

Ans: to the ques: 03.

The difference between abstract class and Interface class :-

Feature	Abstract class	Interface
Method type	Abstract & concrete methods	Abstract (default; static from Java).
Construction support.	Yes	No
object creation	cant create directly	Cant create directly.
Multiple inheritance.	Not supported	supported.
Access Modifiers in methods.	Can use public, protected etc.	All methods are public by default.
Inheritence keyword.	extends	implements