IT-22013

Answer to the question no-1 :-

In zoo different animals do different things, but all animals can make sounds. Some animals like dogs and cats are 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 ~~only~~ 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 AnimalInterfaceExample {
    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 (){
    System.out.println ("sleeping peacefully...");}
    abstract void makeSound(); }

class cat extends Mammal {
    public void makeSound (){
        System.out.println ("Cat meows : Meow!");
    }}

class Dog extends Mammals{
    public void makeSound (){
        System.out.println ("Dog barks: woof!");
    }}

public class AnimalAbstractExample {
    public static void main (String [] args){
        Mammal m1 = new cat();
        Mammal 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 use an interface table, while abstract classes use 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 MyInterface {
  public void show(){
   }}
class AbstractImpl extends MyAbstract{
 public void show() {
  }}
public class Interface_VsAbstractSpeed {
  public static void main (String [] args){
    MyInterface obj1 = new InterfaceImpl();
    MyAbstract obj2 = new AbstractImpl();
```

```
long start, end;
start = System.nanoTime();
for (int i=0; i<1_000_000; i++){
    obj1.show(); }
end = System.nanoTime();
System.out.println("Interface method time:"
                   +(end-start)+"ns");

start = System.nanoTime();
for (int i=0; i<1_000_000; i++){
    obj2.show();
}
end = System.nanoTime();
System.out.println("Abstract class method time:
                   "+(end-start)+"ns");

}
}
```

## Answer to the question no-3:-

The difference between Abstract class and Interface is as follows:-

| Feature | Abstract class | Interface |
|---|---|---|
| Method type | Abstract & Concrete methods | Abstract (default, static from Java |
| Constructor Support | Yes | NO |
| Object Creation | Can't create directly | Can't create directly. |
| Multiple Inheritance | Not Supported | Supported |
| Access Modifiers in Methods | Can use public, protected etc | All methods are publi by default. |
| Inheritence Keyword | extends | implements |