Scala OO concepts

Intellipaat Software Solutions Pvt. Ltd.

Scala Class and Object

A class is a blueprint for objects. Once you define a class, you can create objects from the class blueprint with the keyword **new**. Following is a simple syntax to define a class in Scala:

```
class Point(x:Int,y:Int) {
varx1:Int = x;
vary1:Int = y;
defmove(dx:Int, dy:Int)
  {
x1 = x1+dx
y1 = y1+dy
println(x1+" "+y1)
  }
 }
```

```
objectTest{
def main(args:Array[String]){
valpt=newPoint(10,20);

// Move to a new location
pt.move(10,10);
}
}
```

Command

```
\>scalac Demo.scala

\>scala Demo
```

Extending a Class

You can extend a base Scala class and you can design an inherited class in the same way you do it in Java (use **extends**key word), but there are two restrictions: method overriding requires the **override** keyword, and only the **primary**constructor can pass parameters to the base constructor.

Singleton Objects

Scala is more object-oriented than Java because in Scala, we cannot have static members. Instead, Scala has **singleton objects**. A singleton is a class that can have only one instance, i.e., Object. You create singleton using the keyword **object**instead of class keyword.

Auxiliary Constructors

In **Scala**, We can define **Auxiliary Constructors** like methods by using “def” and “this” keywords. “this” is the **constructor** name. **Auxiliary Constructor** is also know as Secondary **Constructor**. A **Scala** class can contain zero or one or more**Auxiliary Constructors**.

Exception Handling

Scala's exceptions work like exceptions in many other languages like Java. Instead of returning a value in the normal way, a method can terminate by throwing an exception. However, Scala doesn't actually have checked exceptions.
When you want to handle exceptions, you use a try{...}catch{...} block like you would in Java except that the catch block uses matching to identify and handle the exceptions.

Catching Exceptions

Scala allows you to **try/catch** any exception in a single block and then perform pattern matching against it using **case** blocks. Try the following example program to handle exception.

The finally Clause

You can wrap an expression with a **finally** clause if you want to cause some code to execute no matter how the expression terminates. Try the following program.

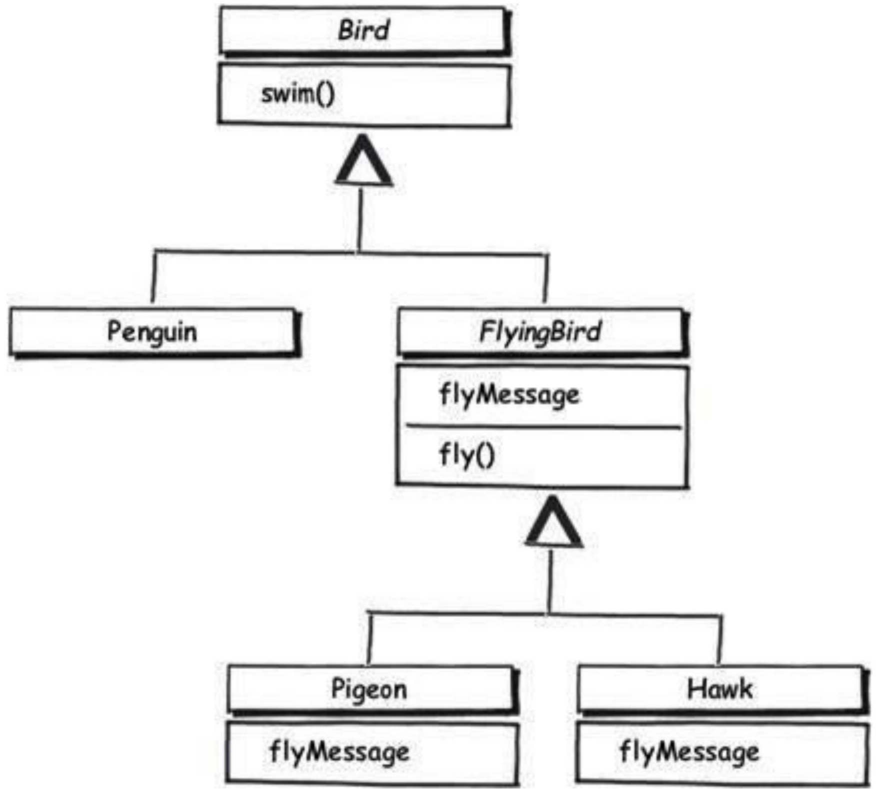
Traits

A trait encapsulates method and field definitions, which can then be reused by mixing them into classes. Unlike class inheritance, in which each class must inherit from just one superclass, a class can mix in any number of traits.

A trait definition looks just like a class definition except that it uses the keyword **trait**. The following is the basic example syntax of trait

```
trait Similarity {
  def isSimilar(x: Any): Boolean
  def isNotSimilar(x: Any): Boolean =
!isSimilar(x)
}
```

Traits


Bird
swim()
Penguin
FlyingBird
flyMessage
fly()
Pigeon
flyMessage
Hawk
flyMessage

Thank You

Email us – support@intellipaat.com

Visit us - https://intellipaat.com