Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Classes and Objects/5b18bde97becc0459de9c064

TUTORIAL

Classes and Objects

Topics

Class

A class is a template or blueprint from which objects are made. Thinking about classes as cookie cutters. Objects are the cookies themselves. When you construct an object from a class, you are said to have created an instance of the class. As you have seen, all code that you write in Java is inside a class. The standard Java library supplies several thousand classes for such diverse purposes as user interface design, dates and calendars, and network programming. Nonetheless, you still have to create your own classes in Java to describe the objects of the problem domains of your applications.

A class can be visualized as a three-compartment box, as illustrated:

i) Class name (or identifier): identifies the class.

ii) Data Members or Variables (or attributes, states, fields): contains the attributes of the class.

iii) Member Functions (or methods, behaviours, operations): contains the dynamic operations of the class.

Classes are created using the keyword class. The declaration of a class is enclosed with curly braces.

For example, below is a class named Circle having two variables and two functions defined.

```
class Circle // class name
{
        double area,radius; // data members ( or variables)
        double getRadius(){ return radius; } // member functions
        double showArea(){ return area;}
}
```

In Python:

```
class ClassName:
   'Optional class documentation string'
   x = 5
  def display():
    print("Hello")
```

A simple rule of thumb in identifying classes is to look for nouns in the problem analysis. Methods, on the other hand, correspond to verbs. For example, in an order-processing system, some of these nouns are

```
• Item           • Order          • Shipping address               •
Payment          • Account
```

These nouns may lead to the classes Item, Order, and so on. Next, look for verbs. Items are added to orders. Orders are shipped or cancelled. Payments are applied to orders. With each verb, such as "add," "ship," "cancel," and "apply," you identify the one object that has the major responsibility for carrying it out. For example, when a new item is added to an order, the order object should be the one in charge because it knows how it stores and sorts items. That is, add should be a method of the Order class that takes an Item object as a parameter. Of course, the "noun and verb" rule is only a rule of thumb and the only experience that can help you decide which nouns and verbs are the important ones when building your classes.

Objects

Defining objects of a class data type is known as class instantiation. When objects are created, only during that moment memory is allocated to them. For example, suppose that we have a class Circle. We can create instances of Circle in the function main () as follows:

```
Circle c1 = new Circle(); // Construct 1 instance c1 of the class
Circle , so memory will be allocated using the new operator.
```

In Python:

```
c1 = Circle()
```

You can create as many objects as you want, each object will be allocated a new memory space, where the data associated with it will be stored. So,

```
Circle c2 = new Circle();
Circle c3 = new Circle();
```

The above code will create two more objects for the class Circle. Each of these 3 objects has a separate space in memory.

Java

```java
class DummyClass
{
  int length, width;
  void setValues(int l, int w)
  {
    length=l;
    width=w;
  }
  int area()
  {
    return (length*width);
  }
}
class Main{
  public static void main(String[] args)
  {
    DummyClass o1 = new DummyClass();   // o1 is an object of class
    DummyClass o2 = new DummyClass();   // o2 is also an object of class
    o1.setValues(5,18);                 // Values
passes to object o1
    o2.setValues(18,10);                // Values
passes to object o2
    System.out.println(o1.area());
    System.out.println(o2.area());
  }
}
```

Python 3

```
class DummyClass:
  length = 0
  width = 0

  def setValues(l, w):
    self.length=l;
    self.width=w;

  def area():
    return (length*width);

o1 = DummyClass()       # o1 is an object of class
o2 = DummyClass()       # o2 is also an object of class
o1.setValues(5,18)      # Values passes to object o1
o2.setValues(18,10)     # Values passes to object o2
print(o1.area())
print(o2.area())

```

The __init__() Function in Python

The examples above are classes and objects in their simplest form and are not really useful in real-life applications.

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties or other operations that are necessary to do when the object is being created:

Python 3

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("Amit", 32)

print(p1.name)
print(p1.age)
```