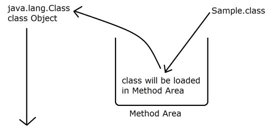
Method Area :
-----------------

* Once we load a class then the class will be loaded inside method area and it returns java.lang.Class class Object.



From this java.lang.Class class Object, We can get complete information regarding the class

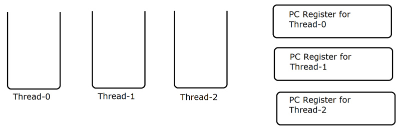
```
Class cls = Class.forName("Sample");

        cls.getName();  -> Get the name of the Class
        cls.getPackageName(); -> Get the name of the package
        cls.getDeclaredMethods(); -> Will give you complete method list
        cls.getDeclaredFields(); -> Will give you complete field list
```

1) public String getName() : Will provide the class name
2) public String getPackageName() :  Will provide the class name
3) public **Method[]** getDeclaredMethods() : Will provide list of methods, return type is Method[]
   Method is a predefined class available in java.lang.reflect sub package.
   * Method class has provided a predefined method called getName() through which we can get the name of the method

4) public Field[] getDeclaredFields() : Will provide list of Fields, return type is Field[]. It is available in java.lang.reflect sub package and provides a predefined method called getName() to get the name of the field

PC Register :
---------------

It stands for Program Counter Register.
In java environment, we can craete multiple threads.
In order to hold the currently executing instruction of each thread we have a separate PC register.



HAS-A Relation in java :
---------------------------

IS-A Relation :
-----------------

```
class Vehicle
{

}
class Car
{

}
```

HAS-A Relation :
------------------

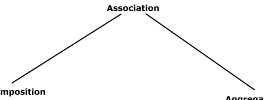
```
class Engine
{

}
class Car
{
    private String carName;
    private int carModel;
    private Engine engine;  //HAS-A Relation
}
```

If we take class name (Engine) as a property to another class (Car class) then it is called HAS-A Relation

* HAS-A Relation we can achieve by using Association concept.

Association :
--------------

It describes the relationship between two classes which is established through **Object Reference.**
It also describes how much a class knows about another class.



**Association**

**Composition**
[ Strong reference, One Object **can't** exist without another Object ]

Example :
------------

Car and Engine
University and Department
Person and Heart
Laptop and Motherboard

```
System.out.println();

class System
{
   PtintStream out; //HAS A relation
}
```

**Aggregation**
[ Weak reference, One Object can exist without another Object]

Example :
------------

Comapny and Employee
College and Student
Customer and Order
Person and Address

```
class Order
{
   orderId
   itemName
   itemPrice
}
class Customer
{
   custId
   name
   address
   mobile
   Private Order order;
}
```