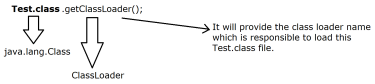
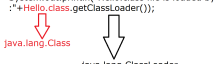
Application class loader is responsible to load the user-defined .class file into JVM Memory :

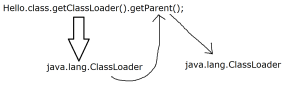
```
class Test
{
}
```

* Whenever Test.class file will be loaded into JVM memory then it will return java.lang.Class class object.




```
class Hello
{
}
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello.class file is loaded by
        :"+Hello.class.getClassLoader());
    }
}
```






```
public ClassLoader getClassLoader();

public ClassLoader getParent();
```

Linking :

* Linking phase contains 3 modules :

a) Verify
b) Prepare
c) Resolve

Verify :

* It ensures the correctness of .class file.
* JVM has a component called **ByteCodeVerifier.**
* It will terminate the program execution for any illegal activity by throwing a Runtime error i.e java.lang.VerifyError
* java.lang.LinkageError is the super class for java.lang.VerifyError

Prepare [Static Data member memory allocation + Initialized by default value]

```
public class Test
{
   static int x = 100;
   int y = 200;

   static Test t1 = new Test();
   Scanner sc = new Scanner(System.in);

   {
     //NSB
   }

   static
   {
     //SB
   }
}
```

With the help of class loader we are going to load Test.class file :

When Test.class file is being loaded into the JVM memory then in the **Prepare phase** **It will scan the class from top to bottom to search all the static data member only.**

We have 2 static data member i.e x and t1

so first of all memory will be allocated for x (4 bytes) and t1 (4 OR 8 bytes depends upon JVM implementation on a particular O.S)

```
x = 0;
t1 = null;
```

default value

So, the conclusion is : all the static data member memory allocation as well as initialized with default value will be done in the prepare phase