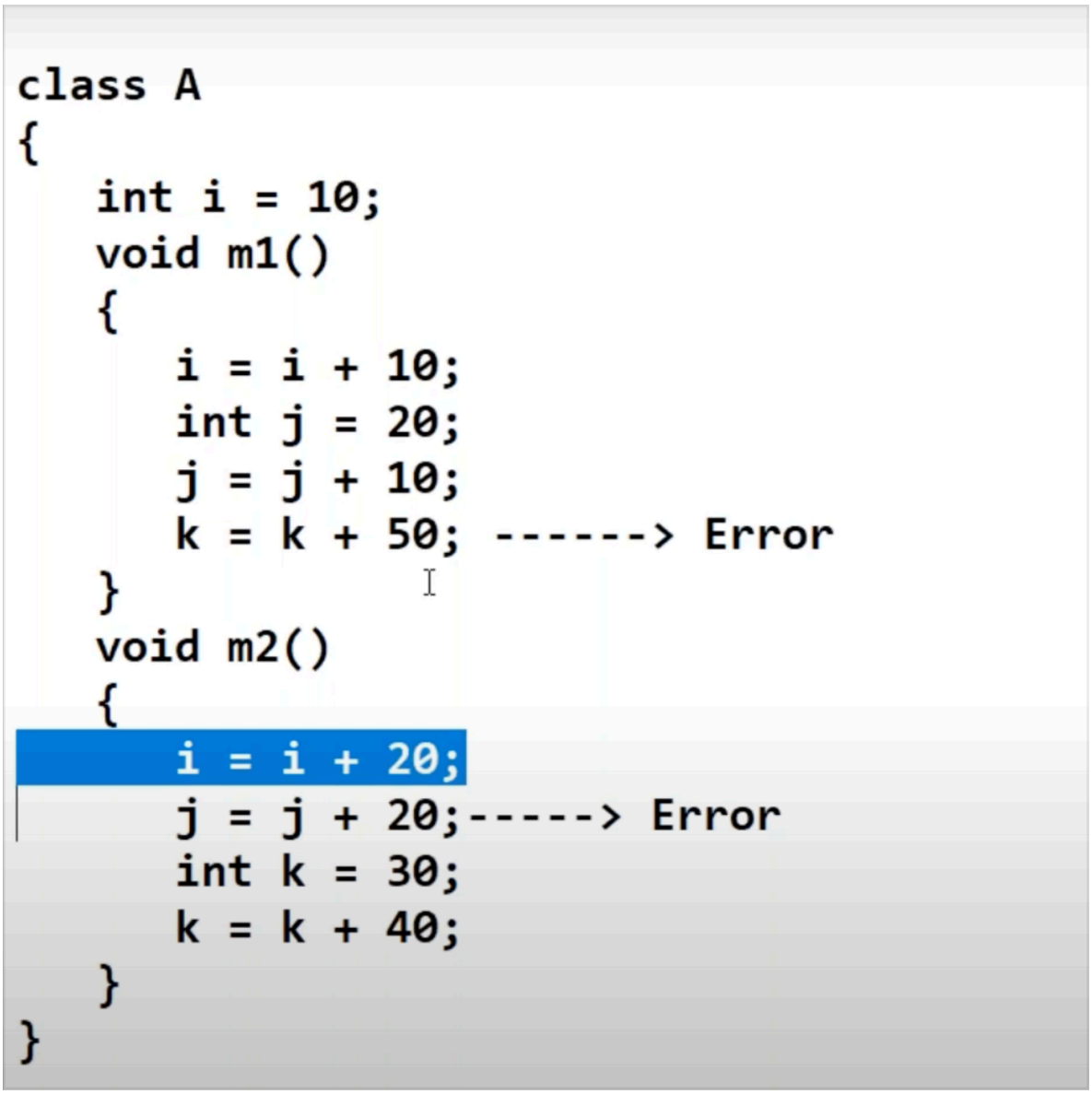
X

```
1. Identifiers:
---------------
Identifier is a name assigned to the programming elements like variables, methods,
clases, interfaces, enums, packages,.......

Rules and Regulations for Identifiers:
--------------------------------------
1. All Java identifiers must not be started with a number, java identifiers must be
started with an alphabet or _ symbol or $ symbol, but, the subsequent symbols may
be a number, an alphabet, _ symbol or $ symbol.
EX:
---
int eno = 111; -------------> Valid
int 9Eno = 999; ------------> Invalid
String _ename = "Durga"; ---> Valid
float $esal = 25000.0f; ----> Valid
String emp9Id = "E-999"; ---> Valid
String emp_Addr = "Hyd"; ---> Valid
```

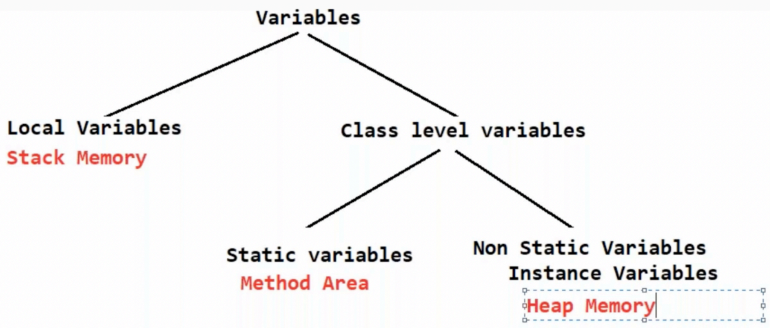
```
3. All Java identifiers are not allowing all operators and all special symbols
except _ symbol and $ symbol.
EX:
---
int emp.No = 111; -------------> Invalid
String emp@Hyd = "Durga"; -----> Invalid
String emp-Name = "Durga"; ----> Invalid
String emp#Addr = "Hyd"; ------> Invalid
int emp_Age = 30; -------------> Valid
float emp+Sal = 10000.0f; -----> Invalid
String emp*Name = "Durga"; ----> Invalid

4. In Java applications, identifiers must not be duplicated with in the same Scope,
but, Identifiers may be duplicated in two different scopes.
```



Class scope and local scope

```
class A
{
    int i = 10;
    long i = 20; ----> Error
    byte f = 50;
    void m1()
    {
        short i = 30;
        float f = 33.33f;
        double f = 234.345; ---> Error
    }
}
```



```
class A
{
    int i = 10;
    void m1()
    {
        short i = 40;
        i = ` + 20;
        Sopln(i);
    }
}

A a = new A();
a.i = a.i + 10;
Sopln(a.i); // 20
```

```
class A
{
    int i = 10;
    void m1()
    {
        short i = 40;
        int k = i + 30;// k = 70
        int j = this.i + 20;// j = 30
    }
}

A a = new A();
a.i = a.i + 10;
Sopln(a.i); // 20
```

```
class A
{
    int i = 10;
    void m1()
    {
        short i = 40;
        i = i + 20;
        Sopln(i);// 60
    }
}

A a = new A();
a.i = a.i + 10;
Sopln(a.i); // 20
```

```java
class Test
{
    public static void main(String[] args)
    {
        int a = 10;
        System.out.println(a);
    }
}
```

```java
class Test
{
    public static void main(String[] args)
    {
        int Exception = 10;
        System.out.println(Exception);
    }
}
```

```java
class Test
{
    public static void main(String[] args)
    {
        String str = "String";
        System.out.println(str);
    }
}
```

```
D:\core_java6>javac Test.java

D:\core_java6>java Test
String

D:\core_java6>
```

```
1  class Test
2  {
3      public static void main(String[] args)
4      {
5          int System = 10;
6          java.lang.System.out.println(System);
7          System = System + 10;
8          java.lang.System.out.println(System);
9          System = System + 10;
10         java.lang.System.out.println(System);
11     }
12 }
13
```

File Edit Format View Help

5. In Java applications, we are able to use all predefined classes names and interfaces as identifiers:
if we declare any predefined class name as an integer variable then in the remaining part of the program we must use that predefined class name as int variables only, we are unable to use as like original class name, in this context, if we want to use the predefined class name as like its original class name there we must use "Fully Qualified Names to the classes".
Note: Specifying classes names along with package names is called as "Fully Qualified Names".