Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

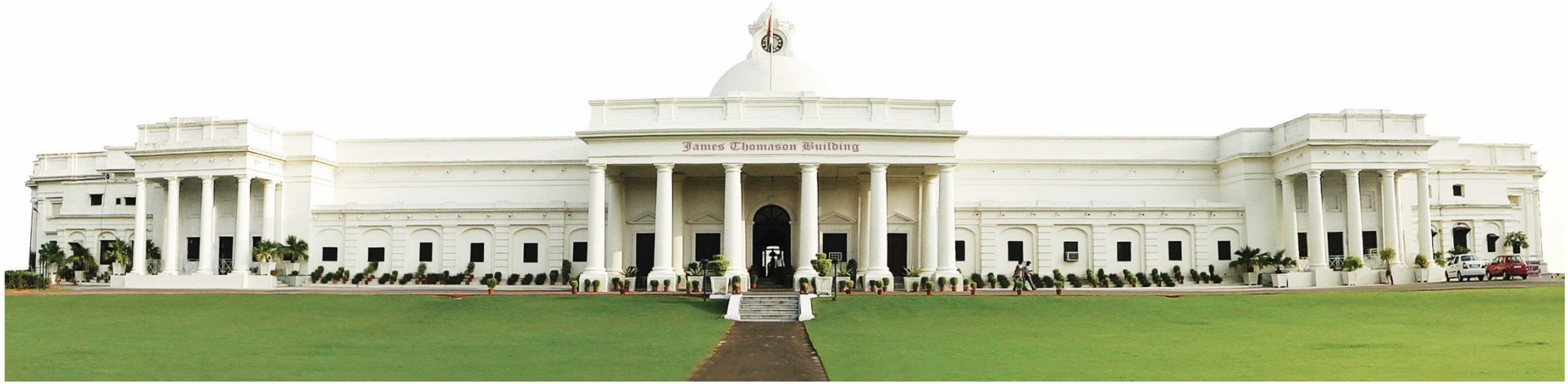
Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/



Why to use this() constructor call

```
1  class Student14{
2      int id;
3      String name;
4      String city;
5
6      Student14(int id,String name){
7      this.id = id;
8      this.name = name;
9      }
10     Student14(int id,String name,String city){
11     this(id,name);//now no need to initialize id and name
12     this.city=city;
13     }
14     void display(){System.out.println(id+" "+name+" "+city);}
15
16     public static void main(String args[]){
17     Student14 e1 = new Student14(18,"Virat");
18     Student14 e2 = new Student14(3,"Suresh","Muradnagar");
19     e1.display();
20     e2.display();
21    }
22 }
```

Terminal

```
sh-4.3$ javac Student14.java
sh-4.3$ java Student14
18 Virat null
3 Suresh Muradnagar
sh-4.3$
```

```java
class Student14{
    int id;
    String name;
    String city;

    Student14(int id,String name){
    this.id = id;
    this.name = name;
    }
    Student14(int id,String name,String city){
    // this(id,name);//now no need to initialize id and name
    this.city=city;
    }
    void display(){System.out.println(id+" "+name+" "+city);}

    public static void main(String args[]){
    Student14 e1 = new Student14(18,"Virat");
    Student14 e2 = new Student14(3,"Suresh","Muradnagar");
    e1.display();
    e2.display();
   }
}
```

Terminal

```
sh-4.3$ javac Student14.java
sh-4.3$ java Student14
18 Virat null
0 null Muradnagar
sh-4.3$
```

```
1  class Student13{
2      int id;
3      String name;
4      Student13(){System.out.println("default constructor is invoked");}
5
6      Student13(int id,String name){
7      //this ();//it is used to invoke current class constructor.
8      this.id = id;
9      this.name = name;
10     this ();
11     }
12     void display(){System.out.println(id+" "+name);}
13
14     public static void main(String args[]){
15     Student13 e1 = new Student13(18,"Virat");
16     Student13 e2 = new Student13(3,"Suresh");
17     e1.display();
18     e2.display();
19    }
20 }
```

Terminal

```
sh-4.3$ javac Student13.java
Student13.java:10: error: call to this must be first statement in constructor
    this ();
         ^
1 error
```

Terminal

```
sh-4.3$ javac Student13.java
Student13.java:10: error: call to this must be first statement in constructor
    this ();
         ^
1 error
```

The this keyword can be used to invoke current class method (implicitly).

```
1  class S{
2    void m(){
3    System.out.println("Use of this keyword in JAVA");
4    }
5    void n(){
6    this.m();//no need because compiler does it for you.
7    }
8    void p(){
9    n();//complier will add this to invoke n() method as this.n()
10   }
11   public static void main(String args[]){
12   S s1 = new S();
13   s1.p();
14   }
15 }
```

Terminal

```
sh-4.3$ javac S.java
sh-4.3$ java S
Use of this keyword in JAVA
sh-4.3$
```

this keyword can be passed as an argument in the method

```
1  class S2{
2    void m(S2 obj){
3    System.out.println("OOP-CSN-103");
4    }
5    void p(){
6    m(this);
7    }
8
9    public static void main(String args[]){
10   S2 s1 = new S2();
11   s1.p();
12   }
13 }
```

Terminal

```
sh-4.3$ javac S2.java
sh-4.3$ java S2
OOP-CSN-103
sh-4.3$
```

Reference object and this, output of both are same

```
1  class A5{
2  void m(){
3  System.out.println(this);//prints same reference ID
4  }
5
6  public static void main(String args[]){
7  A5 obj=new A5();
8  System.out.println(obj);//prints the reference ID
9
10 obj.m();
11 }
12 }
```

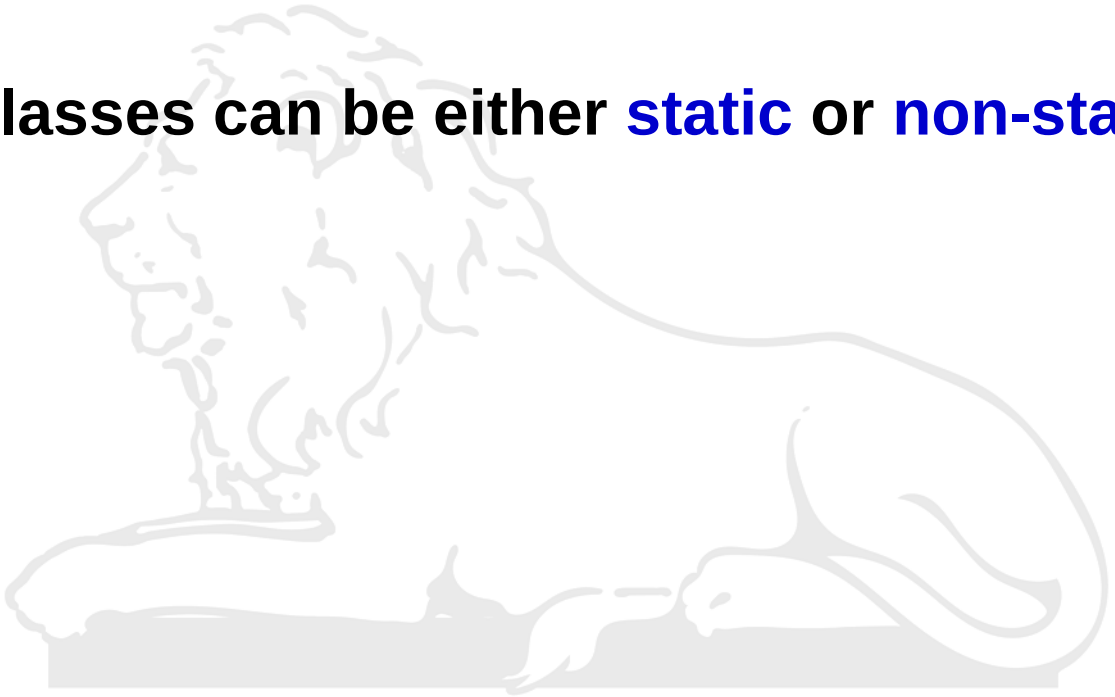
Terminal

```
sh-4.3$ javac A5.java
sh-4.3$ java A5
A5@659e0bfd
A5@659e0bfd
sh-4.3$
```

Inner and Nested Classes in JAVA

- **What is the difference between an inner and nested class in Java? What about the difference between an inner class and a static inner class?**

- **Nested classes can be either static or non-static**



inner class (non static nested class)

```
class OuterClass {
 /* some code here...*/
class InnerClass { }
 /* some code here...*/
}
```

Inner classes are subsets of nested classes

Java Member inner class example

```
1  class TestMemberOuter1{
2   private int data=30;
3   class Inner{
4    void msg(){System.out.println("data is "+data);}
5   }
6   public static void main(String args[]){
7    TestMemberOuter1 obj=new TestMemberOuter1();
8    TestMemberOuter1.Inner in=obj.new Inner();
9    in.msg();
10  }
11 }
```

Terminal

```
sh-4.3$ javac TestMemberOuter1.java
sh-4.3$ java TestMemberOuter1
data is 30
sh-4.3$
```

Example of a static nested class

```
class Outer {
static class NestedStatic { }
}
```

Instantiating a static nested class from a non-enclosing class

```
1  class EnclosingClass {
2    static class Nested {
3    void someMethod() { System.out.println("Welcome to LHC 005"); }
4    }
5  }
6
7  class NonEnclosingClass {
8
9  public static void main(String[] args) {
10 /*instantiate the Nested class that is a static
11   member of the EnclosingClass class:
12 */
13
14 EnclosingClass.Nested n = new EnclosingClass.Nested();
15
16 n.someMethod();  //prints out "hello"
17 }
18 }
```

Terminal

```
sh-4.3$ javac NonEnclosingClass.java
sh-4.3$ java NonEnclosingClass
Welcome to LHC 005
sh-4.3$
```

Instantiating a static nested class from an enclosing class

```
1  class EnclosingClass {
2
3  static class Nested {
4  void anotherMethod() { System.out.println("hi again"); }
5  }
6
7  public static void main(String[] args) {
8  //access enclosed class:
9
10 Nested n = new Nested();
11
12 n.anotherMethod();  //prints out "hi again"
13 }
14
15 }
```

Terminal

```
sh-4.3$ javac EnclosingClass.java
sh-4.3$ java EnclosingClass
hi again
sh-4.3$
```

Dynamic Memory

- C++
  - Designed by Bjarne Stroustrup
  - First appeared 1983; 35 years ago
  - Procedural and Object Oriented!

Simple cpp program

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7     cout << "Hello World" << endl;
8
9     return 0;
10 }
11
12
```

Std : : cout <<

```
Terminal
sh-4.3$ g++ main.cpp
sh-4.3$ a.out
Hello World
sh-4.3$
```

using namespace std

- A symbol may be for instance a function, class or a variable. E.g. if you add **using namespace std;** you can write just cout instead of **std**::cout when calling the operator cout **defined** in the **namespace std**

Simple Program in C++

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int sum, a, b;
8      cout<<"enter a"<<endl;
9      cin>>a;
10     cout<<"enter b"<<endl;
11     cin>>b;
12
13     sum=a+b;
14     cout<<"sum of two numbers="<<sum<<endl;
15
16     return 0;
17 }
```

Terminal

```
sh-4.3$ g++ add2num.cpp
sh-4.3$ a.out
enter a
20
enter b
30
sum of two numbers=50
sh-4.3$
```

Arrays

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7     int a[20];
8     a[0]=1;
9     a[1]=7;
10    a[2]=2;
11    a[3]=9;
12
13    for (int i=0; i<4; i++)
14     {
15      cout<<" "<<a[i];
16     }
17
18    for (int i=4; i<20; i++)
19     {
20      a[i]=a[i-4]+a[i-3];
21      cout<<" "<<a[i];
22        if (i==19)
23      cout<<endl;
24     }
25
26    return 0;
27 }
```