Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

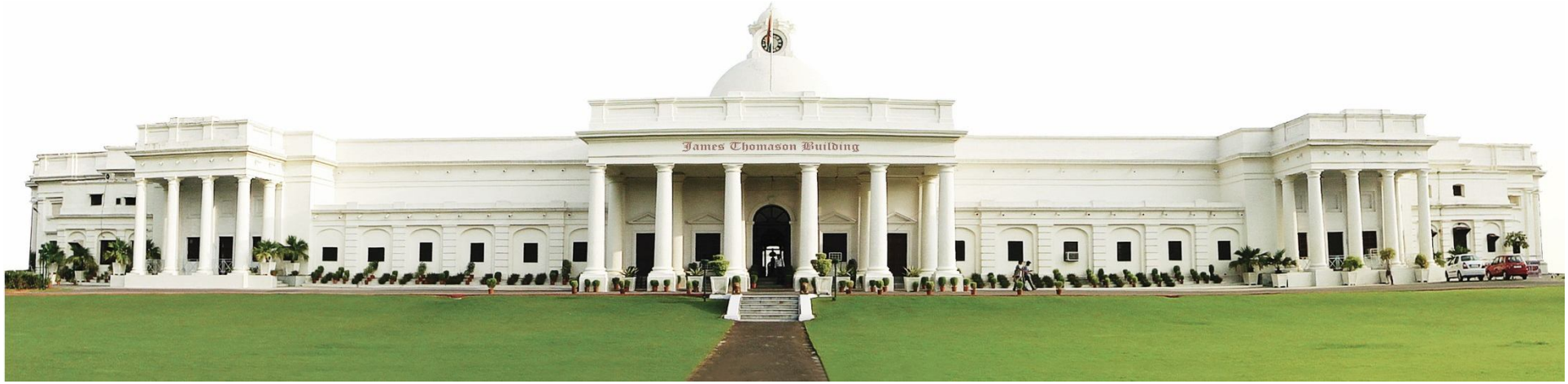
Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

balarfcs@iitr.ac.in

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



Access modifiers in JAVA

- The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.

- There are 4 types of java access modifiers:
  - private
  - default
  - protected
  - public

- There are many non-access modifiers such as static, abstract, synchronized, native, volatile, transient etc. Here, we will learn access modifiers.

private access modifier

- The private access modifier is accessible only within class.

```java
class A{
private int data=40;
private void msg(){
    System.out.println(data);
    System.out.println("Hello java");
    }
}

class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);
   obj.msg();
   }
}
```

compilation info

```
Main.java:12: error: data has private access in A
   System.out.println(obj.data);
                           ^
Main.java:13: error: msg() has private access in A
   obj.msg();
      ^
2 errors
```

stdout

```
Standard output is empty
```

- https://ideone.com/4N9aPL

Public Access Modifier

```java
class A{
private int data=40;
public void msg(){
    System.out.println(data);
    System.out.println("Hello java");
    }
}

class Simple{
 public static void main(String args[]){
   A obj=new A();
 //  System.out.println(obj.data);
   obj.msg();
   }
}
```

stdout

```
40
Hello java
```

- https://ideone.com/0kMsOI

Public Access Modifier

```java
class A{
public int data=40;
public void msg(){
    System.out.println(data);
    System.out.println("Hello java");
    }
}

class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);
   obj.msg();
   }
}
```

stdout

```
40
40
Hello java
```

- https://ideone.com/3S8mKK

Method overriding

```
class Vehicle{
void run(){System.out.println("Vehicle is running");}
}
class Bike2 extends Vehicle{
void run(){System.out.println("Bike is running safely");}

public static void main(String args[]){
Bike2 obj = new Bike2();
obj.run();
}
}
```

stdout

```
Bike is running safely
```

- https://ideone.com/hOByGb

Run Time Polymorphism

Example 1

```java
class X
{
    public void methodA() //Base class method
    {
        System.out.println ("hello, I'm methodA of class X");
    }
}

class Y extends X
{
    public void methodA() //Derived Class method
    {
        System.out.println ("hello, I'm methodA of class Y");
    }
}
class Z
{
   public static void main (String args []) {
       X obj1 = new X(); // Reference and object X
       X obj2 = new Y(); // X reference but Y object
       obj1.methodA();
       obj2.methodA();
   }
}
```

stdout

```
hello, I'm methodA of class X
hello, I'm methodA of class Y
```

https://ideone.com/0cHTNK

Example 2

```java
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){
      System.out.println("Dogs can walk and run");
   }
}

class TestDog{

   public static void main(String args[]){
      Animal a = new Animal(); // Animal reference and object
      Animal b = new Dog(); // Animal reference but Dog object

      a.move();

      b.move();
   }
}
```

stdout

```
Animals can move
Dogs can walk and run
```

- https://ideone.com/l1QEKZ

Example 2, line no. 19

- Here, even though b is a type of Animal it runs the move method in the Dog class.
- The reason for this is: In compile time, the check is made on the reference type. However, in the runtime, JVM figures out the object type and would run the method that belongs to that particular object.
- Hence, in this example, the program will compile properly since Animal class has the method move. Then, at the runtime, it runs the method specific for that object.

Example 3

```java
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){
      System.out.println("Dogs can walk and run");
   }
   public void bark(){
      System.out.println("Dogs can bark");
   }
}

class TestDog{

   public static void main(String args[]){
      Animal a = new Animal(); // Animal reference and object
      Animal b = new Dog(); // Animal reference but Dog object

      a.move();
      b.move();
      b.bark();
   }
}
```

compilation info

```
Main.java:26: error: cannot find symbol
      b.bark();
       ^
  symbol:   method bark()
  location: variable b of type Animal
1 error
```

stdout

```
Standard output is empty
```

https://ideone.com/6TyDVE

Using the super keyword:

```java
class Animal{

   public void move(){
      System.out.println("Animals can move");
   }
}

class Dog extends Animal{

   public void move(){
      super.move(); // invokes the super class method
      System.out.println("Dogs can walk and run");
   }
}

class TestDog{

   public static void main(String args[]){

      Animal b = new Dog(); // Animal reference but Dog object
      b.move(); //Runs the method in Dog class

   }
}
```

stdout

```
Animals can move
Dogs can walk and run
```

https://ideone.com/0wBdks

Runtime polymorphism can't be achieved by data members

```java
class Bike{
 int speedlimit=80;
}
class Honda3 extends Bike{
 int speedlimit=160;

 public static void main(String args[]){
  Bike obj=new Honda3();
  System.out.println(obj);
  System.out.println(obj.speedlimit);
}
}
```

Terminal

```
sh-4.3$ javac Honda3.java
sh-4.3$ java Honda3
Honda3@49e61582
80
sh-4.3$
```

- https://ideone.com/O4gb1v

Runtime Polymorphism

```java
class Bike{
 int speedlimit=80;
 void run(){System.out.println("In Super Class");}
}
class Honda3 extends Bike{
 int speedlimit=160;
 void run(){System.out.println("In Sub Class");}

 public static void main(String args[]){
  Bike obj=new Honda3();
  System.out.println(obj);
  System.out.println(obj.speedlimit);
  obj.run();
}
}
```

Terminal

```
sh-4.3$ javac Honda3.java
sh-4.3$ java Honda3
Honda3@56de24c5
80
In Sub Class
sh-4.3$
```

https://ideone.com/KqT9tL

Run time polymorphism

- Find the output of

```java
class Animal{
void eat(){System.out.println("animal is eating...");}
}

class Dog extends Animal{
void eat(){System.out.println("dog is eating...");}
}

class BabyDog extends Dog{
public static void main(String args[]){
Animal a=new BabyDog();
System.out.println(a);
a.eat();
}}
```

Terminal

```
sh-4.3$ javac BabyDog.java
sh-4.3$ java BabyDog
BabyDog@12402e11
dog is eating...
sh-4.3$
```

- https://ideone.com/YKjQOU

Abstract Class

- A class that is declared with abstract keyword, is known as abstract class in java.
- It can have abstract and non-abstract methods (method with body).
- Before learning java abstract class, let's understand the abstraction in java first.

Abstraction

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
  - SMS example
  - Laundry example by Steve Jobs

Interview with Steve Jobs

- Here, in an excerpt from a 1994 Rolling Stone interview, Steve Jobs explains what object-oriented programming is-

  Jeff Goodell: Would you explain, in simple terms, exactly what object-oriented software is?

  Steve Jobs: Objects are like people. They're living, breathing things that have knowledge inside them about how to do things and have memory inside them so they can remember things. And rather than interacting with them at a very low level, you interact with them at a very high level of abstraction, like we're doing right here.

- Here’s an example: If I’m your laundry object, you can give me your dirty clothes and send me a message that says, “Can you get my clothes laundered, please.” I happen to know where the best laundry place in San Francisco is. And I speak English, and I have dollars in my pockets. So I go out and hail a taxicab and tell the driver to take me to this place in San Francisco. I go get your clothes laundered, I jump back in the cab, I get back here. I give you your clean clothes and say, “Here are your clean clothes.”

- You have no idea how I did that. You have no knowledge of the laundry place. Maybe you speak French, and you can't even hail a taxi. You can't pay for one, you don't have dollars in your pocket. Yet I knew how to do all of that. And you didn't have to know any of it. All that complexity was hidden inside of me, and we were able to interact at a very high level of abstraction. That's what objects are. They encapsulate complexity, and the interfaces to that complexity are high level.

- Source: https://www.quora.com/What-is-object-oriented-programming/answer/Amogh-Talpallikar

Ways to achieve Abstraction

- There are two ways to achieve abstraction in java
  - Abstract class (0 to 100%)
  - Interface (100%)