Inner Classes

Inner Classes

- Inner class (inner) is a class that was declared within the scope of another class (outer).
- Every Instance of the inner class holds the reference for the outer class object, that was in presence when the inner class object was instantiated.

Inner Classes

- Within the scope of methods that are called on the inner object it is possible to use\call variables\methods that were declared in the outer class.

Inner Classes

```
public class HumanBody
{
        private Leg legs[];
        private Hand hands[];
        private Brain mind;

        public HumanBody()
        {
                legs = new Leg[2];
                legs[0] = new Leg("left");
                legs[1] = new Leg("right");
                hands = new Hand[2];
                hands[0] = new Hand("left");
                hands[1] = new Hand("right");
                mind = new Brain();
        }
```

Inner Classes

```
Brain getBrain()
{
        return mind;
}

class Leg
{
        String name;
        Leg(String str)
        {
                name = str;
        }
        void vibe(int num)
        {
                for(int i=0; i<num; i++)
                {
                        System.out.println(name+" leggg...");
                }
        }
}
```

Inner Classes

```
class Hand
{
        String name;
        Hand(String str)
        {
                name = str;
        }
        void vibe(int num)
        {
                for(int i=0; i<num; i++)
                {
                        System.out.println(name+" handddd...");
                }
        }
}
```

Inner Classes

```
        class Brain
        {
                void vibe(int val)
                {
                        legs[0].vibe(val);
                        legs[1].vibe(val);
                        hands[0].vibe(val);
                        hands[1].vibe(val);
                }
        }
}
```

Private Members

- If the enclosing class has a private member, the inner class can see it.
- If the inner class has a private member, the enclosing class can see it.

Private Members

```
public class A
{
        public static void main(String args[])
        {
                A ob = new A();
        }
        private int number;
        B b;
        public A() {b = new B(); b.num = 7;}
        public class B
        {
                private int num;
                public B() { number = 17; }
        }
}
```

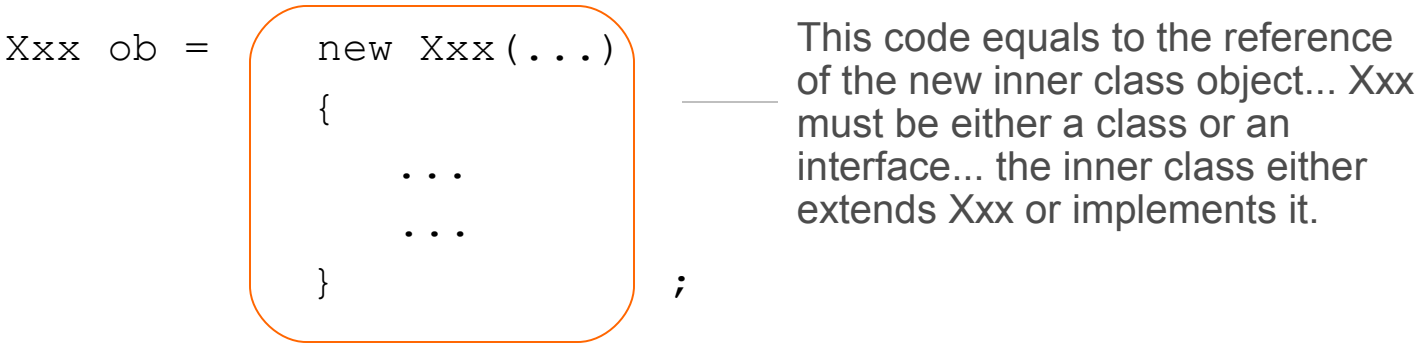
private members defined in the enclosing classare accessible from the inner class and vice verso.

Inner Classes within Methods

- The inner class can be defined within an instance method of the outer class.
- When doing so, the inner class object can use the method's local variables (e.g. its parameters) if they are marked as final only.

Anonymous Inner Classes

- Anonymous inner class is a special inner class that doesn't have a name and was declared just for one instantiation.



Anonymous Inner Classes

The following code creates a new anonymous class that extends JButton. The “OK” is passed over to the JButton constructor that is executed before the anonymous inner class constructor is executed.

```
...
JButton bt = new JButton(“OK”)
{
     ...
     ...
};
...
```

The text in dark red is valued as a reference for a new object instantiated from an anonymous inner class that extends JButton.

Static Inner Class

- The inner class can be declared as a static one. Static inner classes behave differently.
- When instantiating a static inner class, the new object doesn’t have a specific outer object associated with.

Inner Classes

Inner Classes

- Inner class (inner) is a class that was declared within the scope of another class (outer).
- Every Instance of the inner class holds the reference for the outer class object, that was in presence when the inner class object was instantiated.

During the execution of our code, each method or a variable name that are not recognized as ones that belong to the inner object are checked whether they belong to the outer object.

The inner object can call\access methods\variables that belong to the outer object even if there access modifier is private.

The JVM doesn't know the difference between inner class to a none inner class.

The outcome of compiling a class declaration that includes the declaration of an inner class is two java byte code files.

Unlike normal classes, the inner classes can be declared with the 'protected' and with the 'private' access modifiers.

Inner Classes

- Within the scope of methods that are called on the inner object it is possible to use\call variables\methods that were declared in the outer class.

Inner Classes

```
public class HumanBody
{
        private Leg legs[];
        private Hand hands[];
        private Brain mind;

        public HumanBody()
        {
                legs = new Leg[2];
                legs[0] = new Leg("left");
                legs[1] = new Leg("right");
                hands = new Hand[2];
                hands[0] = new Hand("left");
                hands[1] = new Hand("right");
                mind = new Brain();
        }
```

Inner Classes

```
Brain getBrain()
{
        return mind;
}

class Leg
{
        String name;
        Leg(String str)
        {
                name = str;
        }
        void vibe(int num)
        {
                for(int i=0; i<num; i++)
                {
                        System.out.println(name+" leggg...");
                }
        }
}
```

Inner Classes

```
class Hand
{
        String name;
        Hand(String str)
        {
                name = str;
        }
        void vibe(int num)
        {
                for(int i=0; i<num; i++)
                {
                        System.out.println(name+" handddd...");
                }
        }
}
```


Inner Classes

```
        class Brain
        {
                void vibe(int val)
                {
                        legs[0].vibe(val);
                        legs[1].vibe(val);
                        hands[0].vibe(val);
                        hands[1].vibe(val);
                }
        }
}
```

Private Members

- If the enclosing class has a private member, the inner class can see it.
- If the inner class has a private member, the enclosing class can see it.

Private Members

```
public class A
{
        public static void main(String args[])
        {
                A ob = new A();
        }
        private int number;
        B b;
        public A() {b = new B(); b.num = 7;}
        public class B
        {
                private int num;
                public B() { number = 17; }
        }
}
```

private members defined in the enclosing classare accessible from the inner class and vice verso.

Inner Classes within Methods

- The inner class can be defined within an instance method of the outer class.
- When doing so, the inner class object can use the method's local variables (e.g. its parameters) if they are marked as final only.

Anonymous Inner Classes

- Anonymous inner class is a special inner class that doesn't have a name and was declared just for one instantiation.

```
Xxx ob = new Xxx(...)
         {
             ...
             ...
         } ;
```

This code equals to the reference of the new inner class object... Xxx must be either a class or an interface... the inner class either extends Xxx or implements it.

The place, where the inner class is declared, is also the place where it is instantiated.

The Anonymous inner class must extend another class or implements an interface. The Xxx is the name of the class that the anonymous inner class extends or the name of the interface that the anonymous inner class implements.

If the anonymous inner class extends (not implements) another class then it is possible sending, through the brackets, arguments to the super constructor.

The use of anonymous inner class is very common in handling window events. An example for that can be seen in the handling events chapter.

Anonymous Inner Classes

The following code creates a new anonymous class that extends JButton. The “OK” is passed over to the JButton constructor that is executed before the anonymous inner class constructor is executed.

```
...
JButton bt = new JButton(“OK”)
{
     ...
     ...
};
...
```

The text in dark red is valued as a reference for a new object instantiated from an anonymous inner class that extends JButton.

The place, where the inner class is declared, is also the place where it is instantiated.

The Anonymous inner class must extend another class or implements an interface. The Xxx is the name of the class that the anonymous inner class extends or the name of the interface that the anonymous inner class implements.

If the anonymous inner class extends (not implements) another class then it is possible sending, through the brackets, arguments to the super constructor.

The use of anonymous inner class is very common in handling window events. An example for that can be seen in the handling events chapter.

Static Inner Class

- The inner class can be declared as a static one. Static inner classes behave differently.
- When instantiating a static inner class, the new object doesn't have a specific outer object associated with.