**1) How many types of nested classes are there in java?**

Java supports 2 types of nested classes. They are,

a) [Static Nested Classes](https://javaconceptoftheday.com/nested-classes-in-java/)

b) Non-static Nested Classes OR Inner Classes

Non-static nested classes can be of 3 type,

a) [Member Inner Classes](https://javaconceptoftheday.com/inner-classes-in-java/)

b) [Local Inner Classes](https://javaconceptoftheday.com/local-inner-class-in-java/)

c) [Anonymous Inner Classes](https://javaconceptoftheday.com/anonymous-inner-class-in-java/)

**2) Can we access non-static members of outer class inside a static nested class?**

No, we can’t access non-static members of outer class inside a static nested class. We can access only static members of outer class inside a static nested class.

**3) What are member inner classes in java?**

Member inner classes are the classes which are declared as non-static members of another class. Member inner classes can be accessed only by instantiating the outer class.

**4) Can member inner classes have static members in them?**

No, member inner classes can’t have static members in them. They can have only non-static members. But, exception being the static and final field. i.e member inner class can have static and final field, but it must be initialized at the time of declaration only.

**5) Can we access all the members of outer class inside a member inner class?**

Yes, we can access all the members, both static and non-static, of outer class inside a member inner class.

**6) Can we declare local inner classes as static?**

No. Local inner classes can’t be static.

**7) Can we use local inner classes outside the method or block in which they are defined?**

No. Local inner classes are local to method or block in which they are defined. We can’t use them outside the method or block in which they are defined.

**8) Can we declare local inner classes as private or protected or public?**

No. Local inner classes can’t be declared with access modifiers.They can’t be private or protected or public.

**9) What is the condition to use local variables inside a local inner class?**

The condition is that local variables must be final. We can’t use non-final local variables inside a local inner class.

**10) What are anonymous inner classes in java?**

Anonymous inner classes are the inner classes without a name. You can instantiate an anonymous inner class only once. Click [here](https://javaconceptoftheday.com/anonymous-inner-class-in-java/) for more info on anonymous inner classes.

**11) What is the main difference between static and non-static nested classes?**

The main difference between static and non-static nested classes is that you need not to instantiate the outer class to access static nested classes. But, to access non-static nested classes, you have to instantiate the outer class.

1. **Question 1. What Is An Inner Class?**

**Answer :**

Inner class is a class defined inside another class and act like a member of the enclosing class.

1. **Question 2. What Are The Advantages Of Inner Classes?**

**Answer :**

The embedding of inner class into the outer class in the case when the inner class is to be used only by one class i.e. the outer class makes the package more streamlined. Nesting the inner class code where it is used (inside the outer class) makes the code more readable and maintainable. The inner class shares a special relationship with the outer class i.e.

the inner class has access to all members of the outer class and still have its own type is the main advantages of Inner class. Advantage of inner class is that they can be hidden from the other classes in the same package and still have the access to all the members (private also) of the enclosing class. So the outer class members which are going to be used by the inner class can be made private and the inner class members can be hidden from the classes in the same package.

This increases the level of encapsulation. If a class A is written requires another class B for its own use, there are two ways to do this. One way is to write a separate class B or to write an inner class B inside class A. Advantage of writing the inner class B in the class A is you can avoid having a separate class. Inner classes are best used in the event handling mechanism and to implement the helper classes. The advantage of using inner class for event handling mechanism is that the use of if/else to select the component to be handled can be avoided. If inner classes are used each component gets its own event handler and each event handler implicitly knows the component it is working for.

[Adv Java Interview Questions](https://www.wisdomjobs.com/e-university/adv-java-interview-questions.html)

1. **Question 3. What Is Static Member Class?**

**Answer :**

A static member class behaves much like an ordinary top-level class, except that it can access the static members of the class that contains it. The static nested class can be accessed as the other static members of the enclosing class without having an instance of the outer class. The static class can contain non-static and static members and methods.

public class InnerClass

{

static class StaticInner

{

static int i = 9;

int no = 6;

private void method() {}

public void method1() {}

static void method2() {}

final void method3() {}

}

}

**The static inner class can be accessed from Outer Class in the following manner:**

* + InnerClass.StaticInner staticObj= new InnerClass. StaticInner ();
  + No outer class instance is required to instantiate the nested static class because the static class is a static member of the enclosing class.

1. **Question 4. What Are The Different Types Of Inner Classes?**

**Answer :**

**The below mentioned are the types of inner classes:–**

* + Static member class
  + Inner class
  + Member class
  + Anonymous class
  + Local class

[Adv Java Tutorial](https://www.wisdomjobs.com/e-university/adv-java-tutorial-227.html)

1. **Question 5. What Are Non Static Inner Classes?**

**Answer :**

**The different type of static inner classes is:** Non - static inner classes – classes associated with the object of the enclosing class.

**Member class** - Classes declared outside a function (hence a "member") and not declared "static". The member class can be declared as public, private, protected, final and abstract.

**Example:**

public class InnerClass

{

class MemberClass

{

public void method1() { }

}

}

**Method local class** – The inner class declared inside the method is called method local inner class. Method local inner class can only be declared as final or abstract. Method local class can only access global variables or method local variables if declared as final

public class InnerClass

{

int i = 9;

public void method1()

{

final int k = 6;

class MethodLocal

{

MethodLocal()

{

System.out.println(k + i);

}

}

}

}

**Anonymous inner class** - These are local classes which are automatically declared and instantiated in the middle of an expression. Also, like local classes, anonymous classes cannot be public, private, protected, or static. They can specify arguments to the constructor of the superclass, but cannot otherwise have a constructor. They can implement only one interface or extend a class.

public class MyFrame extends JFrame

{

JButton btn = new JButton();

MyFrame()

{

btn.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {}

});

}

}

//Anonymous class used with comparator

List l = new ArrayList();

l.add(new Parent(2));

l.add(new Parent(3));

Collections.sort(l, new Comparator()

{

public int compare(Object o1, Object o2)

{

Parent prt1 = (Parent) o1;

Parent prt2 = (Parent) o2;

if (prt1.getAge() > prt2.getAge())

{

return -1;

}

else if(prt1.getAge() {

return 1;

}

else

{

return 0;

}

}

});

[J2EE Interview Questions](https://www.wisdomjobs.com/e-university/j2ee-interview-questions.html)

1. **Question 6. Can A Static Nested Class Have Access To The Enclosing Class's Non-static Methods Or Instance Variables?**

**Answer :**

No.

1. **Question 7. What Are Disadvantages Of Using Inner Classes?**

**Answer :**

* + Using inner class increases the total number of classes being used by the application. For all the classes created by JVM and loaded in the memory, jvm has to perform some tasks like creating the object of type class. Jvm may have to perform some routine tasks for these extra classes created which may result slower performance if the application is using more number of inner classes.
  + Inner classes get limited support of ide/tools as compared to the top level classes, so working with the inner classes is sometimes annoying for the developer.

[J2EE Tutorial](https://www.wisdomjobs.com/e-university/j2ee-tutorial-230.html) [Core Java Interview Questions](https://www.wisdomjobs.com/e-university/core-java-interview-questions.html)

1. **Question 8. What Are Different Types Of Anonymous Classes?**

**Answer :**

**Plain old anonymous class type one:–**

class superclass{

void doSomething() {

System.out.println(“Doing something in the Super class”);

}

}

class hasAnonymous{

superClass anon = new superClass(){

void doSomething() {

System.out.println(“Doing something in the Anonymous class”);

}

};

Here anon is the reference which is of type superclass which is the class extended by the anonymous class i.e. superclass of the anonymous class. The method doSomething () is the super class method overridden by the anonymous class.

**Plain old anonymous class type two:–**

interface Eatable {

public void prepare Sweets();

}

class serveMeal {

Eatable food = new Eatable(){

public void

prepareSweets(){ //come implementation code goes here

}

};

}

food is reference variable of type Eatable interface which refers to the anonymous class which is the implementer of the interface Eatable. The anonymous implementer class of the interface Eatable implements its method prepare Sweets () inside it.

**Argument defined anonymous class:–**

interface Vehicle {

void getNoOfWheels();

}

class Car {

void getType(Vehical v) { }

}

class BeautifulCars

{

void getTheBeautifilCar()

{

Car c = new Car ();

c.getType (new Vehicle ()

{

public void getNoOfWheels ()

{

System.out.println("It has four wheels");

}

});

}

}

Anonymous class is defined as the argument of the method getTheBeautifilCar (), this anonymous class is the implementer of the interface Vehicle. The method of class Car getTheBeautifilCar () expects the argument as an object of type Vehicle. So first we create an object of Car referenced by the variable ‘c’. On this object of Car we call the method getTheBeautifilCar () and in the argument we create an anonymous class in place which is the implementer of interface Vehicle hence of type Vehicle.

1. **Question 9. If You Compile A File Containing Inner Class How Many .class Files Are Created And Are All Of Them Accessible In Usual Way?**

**Answer :**

If a inner class enclosed with an outer class is compiled then one .class file for each inner class and a .class file for the outer class is created.

**Example:**

class EnclosingOuter

{

class Inner{ }

}

**If you compile the above code with command:**

% javac EnclosingOuter.java Two files will be created. Though a separate inner class file is generated, the inner class file is not accessible in the usual way.

EnclosingOuter.class

EnclosingOuter$Inner.class

[JSP Interview Questions](https://www.wisdomjobs.com/e-university/jsp-interview-questions.html)

1. **Question 10. How To Access The Inner Class From Code Within The Outer Class?**

**Answer :**

The inner class is instantiated only through the outer class instance.

class EnclosingOuter

{

private int noInnerClass = 1;

public void getNoOfInnerClasses()

{

Inner in = new Inner();

System.out.println("No Of Inner classes is :"+ in.getNoOfClassesFromOuter());

}

class Inner

{

public int getNoOfClassesFromOuter()

{ return noInnerClass;

}

}

Here the method getNoOfInnerClasses () is called on the outer class’s instance and through this outer class instance the inner class instance is created.

[Core Java Tutorial](https://www.wisdomjobs.com/e-university/core-java-tutorial-231.html)

1. **Question 11. How To Create An Inner Class Instance From Outside The Outer Class Instance Code?**

**Answer :**

To create an instance of the inner class you must have the instance of its enclosing class.

class Enclosing Outer

{

class Inner{ }

}

To create the instance of inner class from class other than the enclosing class.

1) class OtherThanOuter

{

Enclosing Outer out = new Enclosing Outer ();

EnclosingOuter.Inner in = out.new Inner ();

}

2) class OtherThanOuter

{

EnclosingOuter.Inner out = new EnclosingOuter.Inner ();

}

[Java-Springs Interview Questions](https://www.wisdomjobs.com/e-university/java-springs-interview-questions.html)

1. **Question 12. Which Modifiers Can Be Applied To The Inner Class?**

**Answer :**

**Following modifiers can be applied to the inner class:**

* + Public
  + Private
  + Abstract
  + Final
  + protected
  + Strict
  + Static – turns the inner class into
  + Static nested class

[Adv Java Interview Questions](https://www.wisdomjobs.com/e-university/adv-java-practice-tests-227-327247)

1. **Question 13. Can The Method Local Inner Class Object Access Method Local Variables?**

**Answer :**

No, a method local inner class object cannot access the method local variable.

**Reason:** The local variables are not guaranteed to live as long as the local inner class objects. The method local variable live on stack and exist only till the method lives, their scope is limited upto the code inside the method they are declared in. But the local inner class object created within the method lives on heap and it may exist even after the method ends if in case the reference of this local inner class is passed into some other code and is stored in an instance variable. So we cannot be sure that the local variables will live till the method local inner class object lives, therefore the method local inner class object cannot access the method local variable. To access the method local variables, the variable has to be declared as final.

[JSP Tutorial](https://www.wisdomjobs.com/e-university/jsp-tutorial-279.html)

1. **Question 14. Can A Method Local Inner Class Access The Local Final Variables? Why?**

**Answer :**

Yes. Because the final variables are stored on heap and live as long as the method local inner class objects may live.

1. **Question 15. Which Modifiers Can Be Applied To The Method Local Inner Class?**

**Answer :**

Only abstract or final keyword is allowed.

[JMS(Java Message Service) Interview Questions](https://www.wisdomjobs.com/e-university/jms-java-message-service-interview-questions.html)

1. **Question 16. Can A Local Class Declared Inside A Static Method Have Access To The Instance Members Of The Outer Class?**

**Answer :**

No. There is no reference available in the static method about the instance members of the outer class. The static method class cannot have access to any members of the outer class other than static members.

[Java-Springs Tutorial](https://www.wisdomjobs.com/e-university/java-springs-tutorial-287.html)

1. **Question 17. Can A Method Which Is Not In The Definition Of The Superclass Of An Anonymous Class Be Invoked On That Anonymous Class Reference?**

**Answer :**

No. As the reference variable type of the anonymous class will be of superclass which will not know about any method defined inside the anonymous class and hence the compilation will fail.

class Superclass

{

void doSomething()

{

System.out.println ("In the Super class");

}

}

class has Anonymous

{

Superclass anon = new Superclass ()

{

void doSomething()

{

System.out.println("In the Anonymous class");

}

void doStuff()

{

System.out.println ("An Anonymous class method not present in

superclass");

}

};

public void doIt()

{

anon. DoSomething(); // legal superclass has this method

anon. DoStuff(); // Not legal

}

}

The above code will not compile as the superclass does not know about the anonymous class method doStuff ().

[Java applet Interview Questions](https://www.wisdomjobs.com/e-university/java-applet-interview-questions.html)

1. **Question 18. Can An Anonymous Class Define Method Of Its Own?**

**Answer :**

Yes. But there will be no way by which the methods defined in the anonymous class which are not present in its superclass be invoked. As only those methods which are defined in the superclass which the anonymous class extends be invoked. Hence defining the methods in the anonymous class will be of no use.

[J2EE Interview Questions](https://www.wisdomjobs.com/e-university/j2ee-interview-questions.html)

1. **Question 19. Can An Anonymous Class Implement An Interface And Also Extend A Class At The Same Time?**

**Answer :**

No. An anonymous class can either extend a class or implement a single interface. If the anonymous class is extending a class then it automatically becomes the implementer of all the interfaces implemented by its superclass.

**Q: What is Inner Class in Java?  
A:**In Java similar methods and variables of a class, we can have a class as member of another class . Declaring a class within another is allowed in Java. The class written within is called the nested class, and the class that holds the inner class is called the outer class. We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.

class Java\_Outer\_Demo {

class Java\_Inner\_Demo {

}

}

**Q: What are the advantages of using Inner Class in Java?  
A:**The Advantages of using Polymer are as follows-

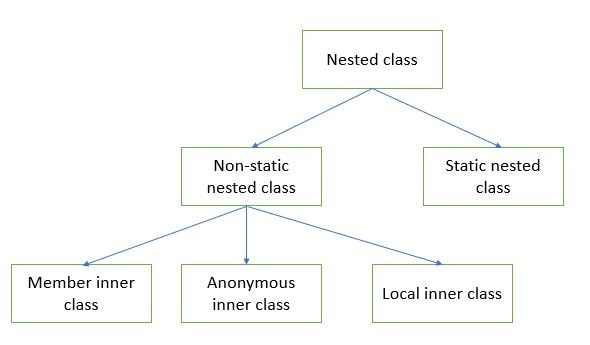
* Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of outer class including private.
* Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.
* Code Optimization: It requires less code to write.

**Q: What are the types of nested classes?  
A:**Nested classes can be divided into two types -

* Non-static nested classes - These are the non-static members of a class.
* Static nested classes - These are the static members of a class.

Non-static class can be further classified into 3 types

* Member inner class
* Anonymous inner class
* Local inner class

  
**Q: What is anonymous inner class in Java?  
A:**It is an inner class without a name and for which only a single object is created. An anonymous inner class can be useful when making an instance of an object with certain "extras"" such as overloading methods of a class or interface, without having to actually subclass a class.  
Anonymous inner classes are useful in writing implementation classes for listener interfaces in graphics programming. Anonymous inner class can be created in 2 ways-

* Class (may be abstract or concrete)
* Interface

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

int returnVal = fileChooser.showOpenDialog(null);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

System.out.println(fileChooser.getSelectedFile().getName());

}

}

});

**Q: Why can outer Java classes access inner class private members?  
A:** The inner class is just a way to cleanly separate some functionality that really belongs to the original outer class. They are intended to be used when you have 2 requirements: Some piece of functionality in your outer class would be most clear if it was implemented in a separate class. Even though it's in a separate class, the functionality is very closely tied to way that the outer class works. Given these requirements, inner classes have full access to their outer class. Since they're basically a member of the outer class, it makes sense that they have access to methods and attributes of the outer class -- including privates.

**Q1) What is an inner class?**

Ans) Inner class is a class defined inside other class and act like a member of the enclosing class.

**Q2) What are the different types of inner classes?**

Ans) There are two main types of inner classes –

* Static member class
* Inner class
  + Member class
  + Anonymous class
  + Local class

**Q3) What is static member class?**

Ans) A static member class behaves much like an ordinary top-level class, except that it can access the static members of the class that contains it. The static nested class can be accessed as the other static members of the enclosing class without having an instance of the outer class. The static class can contain non-static and static members and methods.

public class InnerClass {

static class StaticInner {

static int *i* = 9;

int no = 6;

private void method() {}

public void method1() {}

static void method2() {}

final void method3() {}

}

}

The static inner class can be accessed from Outer Class in the following manner:

InnerClass.StaticInner staticObj= new InnerClass. StaticInner ();

No outer class instance is required to instantiate the nested static class because the static class is a static member of the enclosing class.

**Q4) What are non static inner classes?**

Ans) The different type of static inner classes are: Non - static inner classes – classes associated with the object of the enclosing class.

**Member class** - Classes declared outside a function (hence a "member") and not declared "static".  
The member class can be declared as public, private, protected, final and abstract. E.g.

public class InnerClass {

class MemberClass {

public void method1() { }

}

}

**Method local class** – The inner class declared inside the method is called method local inner class. Method local inner class can only be declared as final or abstract. Method local class can only access global variables or method local variables if declared as final

public class InnerClass {

int i = 9;

public void method1() {

final int k = 6;

class MethodLocal {

MethodLocal() {

System.out.println(k + i);

}

}

}

}

**Anonymous inner class** - These are local classes which are automatically declared and instantiated in the middle of an expression. Also, like local classes, anonymous classes cannot be public, private, protected, or static. They can specify arguments to the constructor of the superclass, but cannot otherwise have a constructor. They can implement only one interface or extend a class.  
Anonymous class cannot define any static fields, methods, or classes, except for static final constants.  
Also, like local classes, anonymous classes cannot be public, private, protected, or static

Some examples:

public class MyFrame extends JFrame {

JButton btn = new JButton();

MyFrame() {

btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {}

});

}

}

//Anonymous class used with comparator

List<Parent> l = new ArrayList<Parent>();

l.add(new Parent(2));

l.add(new Parent(3));

Collections.sort(l, new Comparator() {

public int compare(Object o1, Object o2) {

Parent prt1 = (Parent) o1;

Parent prt2 = (Parent) o2;

if (prt1.getAge() > prt2.getAge()) {

return -1;

}else if(prt1.getAge()<prt2.getAge()) {

return 1;

} else {

return 0;

}

}

});

**Q5)Can a static nested class have access to the enclosing class non-static methods or instance variables?**

Ans) No .

**Q6)What are the advantages of Inner classes?**

Ans) The embedding of inner class into the outer class in the case when the inner class is to be used only by one class i.e. the outer class makes the package more streamlined. Nesting the inner class code where it is used (inside the outer class) makes the code more readable and maintainable.

The inner class shares a special relationship with the outer class i.e. the inner class has access to all members of the outer class and still have its own type is the main advantages of Inner class. Advantage of inner class is that they can be hidden from the other classes in the same package and still have the access to all the members (private also) of the enclosing class. So the outer class members which are going to be used by the inner class can be made private and the inner class members can be hidden from the classes in the same package. This increases the level of encapsulation.

If a class A is written requires another class B for its own use, there are two ways to do this. One way is to write a separate class B or to write an inner class B inside class A. Advantage of writing the inner class B in the class A is you can avoid having a separate class. Inner classes are best used in the event handling mechanism and to implement the helper classes. The advantage of using inner class for event handling mechanism is that the use of if/else to select the component to be handled can be avoided. If inner classes are used each component gets its own event handler and each event handler implicitly knows the component it is working for.

**Q7)What are disadvantages of using inner classes?**

Ans)

* Using inner class increases the total number of classes being used by the application. For all the classes created by JVM and loaded in the memory, jvm has to perform some tasks like creating the object of type class. Jvm may have to perform some routine tasks for these extra classes created which may result slower performance if the application is using more number of inner classes.
* Inner classes get limited support of ide/tools as compared to the top level classes, so working with the inner classes is sometimes annoying for the developer.

**Q8) What are different types of anonymous classes?**

Ans 1) **Plain old anonymous class type one**–

class superClass{

void doSomething() {

System.out.println(“Doing something in the Super class”);

}

}

class hasAnonymous{

superClass anon = new superClass(){

void doSomething() {

System.out.println(“Doing something in the Anonymous class”);

}

};

Here anon is the reference which is of type superClass which is the class extended by the anonymous class i.e. superclass of the anonymous class. The method doSomething() is the super class method overridden by the anonymous class.

**Plain old anonymous class type two** –

interface Eatable {

public void prepareSweets();

}

class serveMeal {

Eatable food = new Eatable(){

public void

prepareSweets(){ //come implementation code goes here }

};

}

food is reference variable of type Eatable interface which refers to the anonymous class which is the implementer of the interface Eatable. The anonymous implementer class of the interface Eatable implements its method prepareSweets() inside it.

**Argument defined anonymous class** –

interface Vehicle {

void getNoOfWheels();

}

class Car {

void getType(Vehical v) { }

}

class BeautifulCars {

void getTheBeautifilCar() {

Car c = new Car ();

c.getType (new Vehicle () {

public void getNoOfWheels () {

System.out.println("It has four wheels");

}

});

}

}

Anonymous class is defined as the argument of the method getTheBeautifilCar(), this anonymous class is the implementer of the interface Vehicle. The method of class Car getTheBeautifilCar() expects the argument as an object of type Vehicle. So first we create an object of Car referenced by the variable ‘c’. On this object of Car we call the method getTheBeautifilCar() and in the argument we create an anonymous class in place which is the implementer of interface Vehicle hence of type Vehicle.

**Q9) If you compile a file containing inner class how many .class files are created and what are all of them accessible in usual way?**

Ans) If a inner class enclosed with an outer class is compiled then one .class file for each inner class an a .class file for the outer class is created. e.g.

class EnclosingOuter {

class Inner{ }

}

If you compile the above code with command

**% javac EnclosingOuter.java**

Two files are created will be created. Though a separate inner class file is generated, the inner class file is not accessible in the usual way.

***EnclosingOuter.class  
EnclosingOuter$Inner.class***

**Q10) How to access the inner class from code within the outer class?**

Ans) The inner class is instantiated only through the outer class instance.

class EnclosingOuter {

private int noInnerClass = 1;

public void getNoOfInnerClasses(){

Inner in = new Inner();

System.out.println("No Of Inner classes is :"+ in.getNoOfClassesFromOuter());

}

class Inner{

public int getNoOfClassesFromOuter(){

return noInnerClass;

}

}

Here the method getNoOfInnerClasses() is called on the outer class’s instance through this outer class instance the inner class instance in is created.

**Q1.  What are different types of inner classes ?**

Ans. Simple Inner Class, Local Inner Class, Anonymous Inner Class , Static Nested Inner Class.

**Q2.  Which access specifier can be used with Class ?**

Ans. For top level class we can only use "public" and "default". We can use private with inner class.

**Q3.  Difference between nested and inner classes ?**

Ans. Inner classes are non static nested classes.

**Q4.  What is a nested interface ?**

Ans. Any interface declared inside a class or an interface. It is static by default.

**Q5.  What is the benefit of inner / nested classes ?**

Ans. You can put related classes together as a single logical group.

Nested classes can access all class members of the enclosing class, which might be useful in certain cases.

Nested classes are sometimes useful for specific purposes. For example, anonymous inner classes are useful for writing simpler event-handling code with AWT/Swing.

**Q6.  Explain Static nested Classes ?**

Ans. The accessibility (public, protected, etc.) of the static nested class is defined by the outer class.

A static nested class is not an inner class, it's a top-level nested class.

The name of the static nested class is expressed with OuterClassName.NestedClassName syntax.

When you define an inner nested class (or interface) inside an interface, the nested class is declared implicitly public and static.

Static nested classes can be declared abstract or final.

Static nested classes can extend another class or it can be used as a base class.

Static nested classes can have static members.

Static nested classes can access the members of the outer class (only static members, obviously).

The outer class can also access the members (even private members) of the nested class through an object of nested class. If you don’t declare an instance of the nested class, the outer class cannot access nested class elements directly.

**Q7.  Explain Inner Classes ?**

Ans. The accessibility (public, protected, etc.) of the inner class is defined by the outer class.

Just like top-level classes, an inner class can extend a class or can implement interfaces. Similarly, an inner class can be extended by other classes, and an inner interface can be implemented or extended by other classes or interfaces.

An inner class can be declared final or abstract.

Inner classes can have inner classes, but you’ll have a hard time reading or understanding such complex nesting of classes.

**Q8.  Explain Method Local Inner Classes ?**

Ans. You can create a non-static local class inside a body of code. Interfaces cannot have local  classes, and you cannot create local interfaces.

Local classes are accessible only from the body of the code in which the class is defined. The local classes are completely inaccessible outside the body of the code in which the class is defined.

You can extend a class or implement interfaces while defining a local class.

A local class can access all the variables available in the body of the code in which it is defined. You can pass only final variables to a local inner class.

**Q9.  Explain about anonymous inner classes ?**

Ans. Anonymous classes are defined in the new expression itself, so you cannot create multiple objects of an anonymous class.

You cannot explicitly extend a class or explicitly implement interfaces when defining an anonymous class.

An anonymous inner class is always created as part of a statement; don't forget to close the statement after the class definition with a curly brace. This is a rare case in Java, a curly brace followed by a semicolon.

Anonymous inner classes have no name, and their type must be either a subclass of the named type or an implementer of the named interface