**Types of Inner Class**

* Member Inner Class / Regular Inner Class
* Local Inner Class
* Static Inner Class
* Annonyous Inner Class

**Member Inner class**

* A non-static class that is created inside a class but outside a method is called member inner class. It is also known as a regular inner class.

**Example**

**class** Outer{

**class** Inner{

**void** show() {

System.***out***.println("Member Inner Class : show()...");

}

}

}

**public** **class** MemberInnerClassDemo {

**public** **static** **void** main(String[] args) {

Outer obj = **new** Outer();

Outer.Inner inner = obj.**new** Inner();

inner.show();

}

}

**Local inner class**

* A class created inside a method is called local inner class.

**Example**

**class** Outer{

**void** show() {

**class** Inner{

**void** show2() {

System.***out***.println("Local Inner Class : show2()...");

}

}

Inner ob = **new** Inner();

ob.show2();

}

}

**public** **class** LocalInnerClassDemo {

**public** **static** **void** main(String[] args) {

Outer ob = **new** Outer();

ob.show();

}

}

**Static Nested Class**

* A static class is a class that is created inside a class, is called a static nested class.

**Example**

**class** Outer{

**static** **class** Inner{

**void** show() {

System.***out***.println("Static Inner Class : show()..");

}

}

}

**public** **class** StaticInnerClassDemo {

**public** **static** **void** main(String[] args) {

Outer.Inner ob = **new** Outer.Inner();

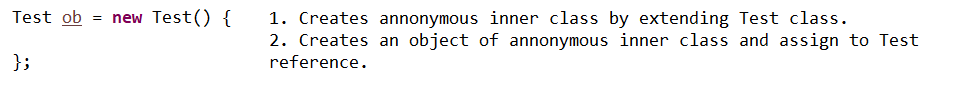
ob.show();

}

}

**Anonymous inner class**

* Anonymous inner class is an inner class without a name.



**Method Overring : Traditional Approach**

**class** Test{

**void** show() {

System.***out***.println("old logic");

}

}

**class** Best **extends** Test{

**void** show() {

System.***out***.println("new logic");

}

}

**public** **class** AnnonymousInnerClassDemo {

**public** **static** **void** main(String[] args) {

Test ob = **new** Best();

ob.show();

}

}

**Method Overring : Annonymous Inner Class Extending Class**

**class** Test{

**void** show() {

System.***out***.println("old logic");

}

}

**public** **class** AnnonymousInnerClassDemo {

**public** **static** **void** main(String[] args) {

Test ob = **new** Test() {

**void** show() {

System.***out***.println("new logic...");

}

};

ob.show();

}

}

**Providing Implementation for abstract method**

**interface** Test{

**void** show();

}

**class** Best **implements** Test{

**public** **void** show() {

System.***out***.println("show()....");

}

}

**public** **class** AnnonymousInnerClassDemo {

**public** **static** **void** main(String[] args) {

Test ob = **new** Best();

ob.show();

}

}

**Annonymous Inner Class Implementing Interface**

**interface** Test{

**void** show();

}

**public** **class** AnnonymousInnerClassDemo {

**public** **static** **void** main(String[] args) {

Test ob = **new** Test() {

**public** **void** show() {

System.***out***.println("show()....");

}

};

ob.show();

}

}

**Multithreading using Annonymous Inner Class**

**Approach1 : without annonymous class**

**class** MyThread **extends** Thread{

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread : "+i);

}

}

}

**public** **class** MultithreadingDemo1 {

**public** **static** **void** main(String[] args) {

MyThread mt = **new** MyThread();

mt.start();

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Main Thread : "+i);

}

}

}

**Approach2: using annonumous inner class**

**public** **class** MultithreadingDemo1 {

**public** **static** **void** main(String[] args) {

Thread t = **new** Thread() {

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread : "+i);

}

}

};

t.start();

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Main Thread : "+i);

}

}

}

**Approach3**

**public** **class** MultithreadingDemo1 {

**public** **static** **void** main(String[] args) {

**new** Thread() {

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread : "+i);

}

}

}.start();

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Main Thread : "+i);

}

}

}

**Creating Multiple Thread using Annonymous Inner Class**

**public** **class** MultithreadingDemo1 {

**public** **static** **void** main(String[] args) {

**new** Thread() {

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread1 : "+i);

}

}

}.start();

**new** Thread() {

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread2 : "+i);

}

}

}.start();

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Main Thread : "+i);

}

}

}

**Lambda Expression**

* Interface having only one abstract method is called functional interface.
* Lambda expression is used with functional interface.

**Example1:**

**interface** Test{

**void** show();

}

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Test t = () -> {

System.***out***.println("code...");

System.***out***.println("code...");

};

t.show();

}

}

**Example2:**

**interface** Test{

**void** add(**int** x, **int** y);

}

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Test t = (a,b) -> {

System.***out***.println(a+b);

};

t.add(10, 30);

}

}

**Example3:**

**interface** Test{

**int** add(**int** x, **int** y);

}

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Test t = (a,b) -> {

**return** a+b;

};

**int** res = t.add(10, 30);

System.***out***.println(res);

}

}

**Example4:**

**interface** Test{

**int** add(**int** x, **int** y);

}

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Test t = (a,b) -> a+b;

**int** res = t.add(20, 30);

System.***out***.println(res);

}

}

**Multithreading Using Lambda Expresssion**

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Runnable ob = () ->{

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Child Thread : "+i);

}

};

Thread t = **new** Thread(ob);

t.start();

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("Main Thread : "+i);

}

}

}

**Create Multiple Thread using Lambda Expression**

**public** **class** LambdaDemo1 {

**public** **static** **void** main(String[] args) {

Runnable ob = () -> {

**for** (**int** i = 1; i <= 10; i++) {

System.***out***.println("Child Thread1 : " + i);

}

};

Thread t = **new** Thread(ob);

t.start();

Runnable ob1 = () -> {

**for** (**int** i = 1; i <= 10; i++) {

System.***out***.println("Child Thread2 : " + i);

}

};

Thread t1 = **new** Thread(ob1);

t1.start();

**for** (**int** i = 1; i <= 10; i++) {

System.***out***.println("Main Thread : " + i);

}

}

}