**Multithreading Enhancements – Part-05**

* **ThreadLocal:**

ThreadLocal class provides thread local variables.

ThreadLocal class maintains values per thread basis.

Each ThreadLocal object maintains as separate value like userId, transactionId, etc… for each thread that accesses that object.

Thread can access its local value, can manipulates its value and even can remove its value.

In every part of the code which is executed by the thread, we can access its local variable.

Example:

Consider a servlet which invokes some business methods, we have a requirement to generate a unique transactionId, for each and every request and we have to pass this transaction id to the business methods for this requirement we can use ThreadLocal to maintain a separate transaction id for every request that is for every thread.

Note:

ThreadLocal class introduced in 1.2 version and enhanced in 1.5 version.

ThreadLocal can be associated with thread scope.

Total code which is executed by the thread has access to the corresponding thread local variables.

A thread can access its own local variables and can’t access other threads local variables.

Once thread entered into dead state all its local variables are by default eligible for garbage collection.

* **Constructor:**

ThreadLocal tl = new ThreadLocal();

Creates a thread local variable

Methods:

Object get()

Returns the value of ThreadLocal variable associated with current thread.

Object initialValue()

Returns initial value of ThreadLocal variable associated with current thread.

The default implementation of this method returns null.

To customize our own initial value we have to override this method.

void set(Object newValue);

To set a new value.

void remove()

To remove the value of ThreadLocal variable associated with current thread.

It is newly added method in 1.5 version.

After removal if we are trying to access, it will be reinitialized once again by invoking its initial value method.

* **ThreadLocal Example:**

class ThreadLocalDemo1{

public static void main(String[] args){

ThreadLocal tl = new ThreadLocal();

System.out.println(tl.get()); // null

tl.set(“durga”);

System.out.println(tl.get()); // durga

tl.remove();

System.out.println(tl.get()); // null

}

}

**Overriding of initialValue method:**

class ThreadLocalDemo1{

public static void main(String[] args){

ThreadLocal tl = new ThreadLocal(){

public Object initialValue(){

return “abc”;

}

};

System.out.println(tl.get()); // null

tl.set(“durga”);

System.out.println(tl.get()); // durga

tl.remove();

System.out.println(tl.get()); // null

}

}

class CustomerThread extends Thread{

static Integer custId = 0;

private static ThreadLocal tl = new ThreadLocal(){

protected Integer initialValue(){

return ++custId;

}

};

CustomerThread(String name){

super(name);

}

public void run(){

System.out.println(Thread.currentThread().getName()+”executing with Customer Id :”+tl.get());

}

}

class ThreadLocalDemo2{

public static void main(String[] args){

CustomerThread c1 = new CustomerThread(“Customer-Thread-1”);

CustomerThread c2 = new CustomerThread(“Customer-Thread-2”);

CustomerThread c3 = new CustomerThread(“Customer-Thread-3”);

CustomerThread c4 = new CustomerThread(“Customer-Thread-4”);

c1.start();

c2.start();

c3.start();

c4.start();

}

}

Note:

In the above program, for every customer thread a separate customerId will be maintained by ThreadLocal object.

* **ThreadLocal VS Inheritance:**

Parent threads ThrealLocal variable by default not available to the child thread.

If we want to make parent threads ThreadLocal variable values available to the child thread, then we should go for InheritableThreadLocal class.

By default child thread value is exactly same as parent threads value, but we can provide customized value for child threads by overriding childValue method.

**Constructor:**

InheritableThreadLocal tl = new InheritableThreadLocal();

**Methods:**

InheritableThreadLocal is the child class of ThreadLocal and hence all methods present in ThreadLocal by default available to inheritable thread local. In addition to these methods, it contains only one method.

public Object childValue(Object childValue)

class ParentThread extends Thread{

public static InheritableThreadLocal tl = new InheritableThreadLocal(){

public Object childValue(Object p){

return “CC”;

}

};

public void run(){

tl.set(“PP”);

System.out.println(“Parent Thread Value---”+tl.get());

ChildThread ct = new ChildThread();

ct.start();

}

}

class ChildThread extends Thread{

public void run(){

System.out.println(“Child Thread value:”+ParentThread.tl.get());

}

}

class ThreadLocalDemo5{

public static void main(String[] args){

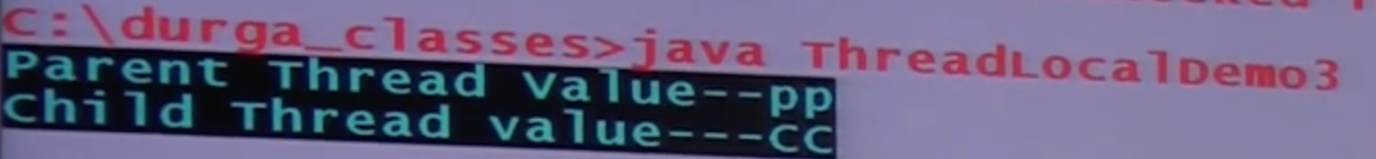
ParentThread pt = new ParentThread();

pt.start();

}

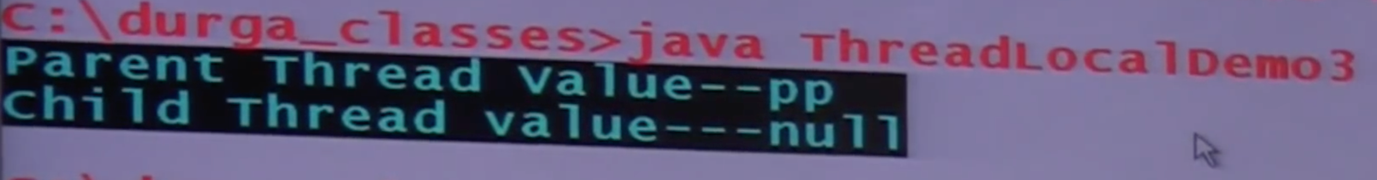
}

Output:



Note:

In the above program, if we replace InheritableThreadLocal with ThreadLocal and if we are not overriding childValue() method, then the output is:



In the above program, if we are maintaining InheritableThreadLocal and if we are not overriding childValue() method, then the output is

