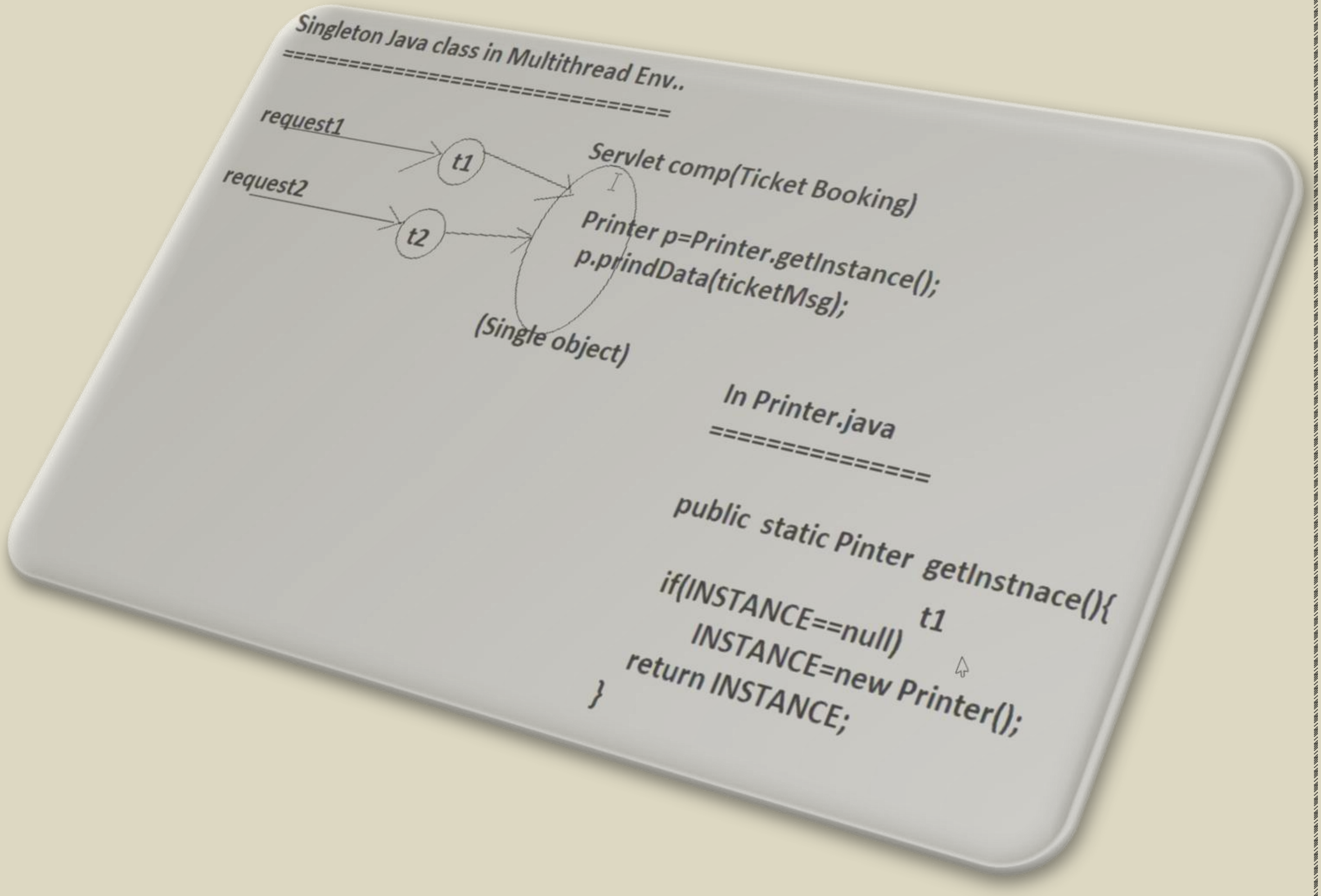
SINGLETON WITH MULTITHREADING



Singleton Java class in Multithread Env..
===================================

```
Servlet comp(Ticket Booking)

Printer p=Printer.getInstance();
p.prindData(ticketMsg);

(Single object)

In Printer.java
===============

public  static Pinter  getInstnace(){
   if(INSTANCE==null)       t1
      INSTANCE=new Printer();
   return INSTANCE;
}
```

"By calling factory method in multithreading env possibility of breaking singleton pattern .. and possibility of creating more then one object for singleton java class"

In above case given two request ...and for two request two thread will be started and both the request given to same servlet component .and servlet component will do ticket booking ...servlet component is calling printer class getInstance method() and print the data for ticket message.. Both thread are acting on same object.

SINGLETON WITH MULTITHREADING

Ex:-

```
package com.nit.dp;
public class printer {
//egar instantiation--> only if object creation at the time of declaration

//private static printer instence=new printer();
private static printer instence;
	private printer() {
		System.out.println("0 param constructor");
	}
	/*public static printer getinstence() {
		return instence;
	}*/

	//statif factory method
	public static printer getInstance() {
		//static factory method
		if(instence==null) {
			instence =new printer(); //lazy instantiation good practice
		}
		return instence;
	}
	public void printDate(String data) {
		System.out.println(data);
	}
}

=========================================================================
```

```
package com.nit.dp;
public class ticketBooking implements Runnable  {
	printer p=null;
	@Override
	 public void run() {
	p=printer.getInstance();
System.out.println("hascode of printer class is="+p.hashCode()+" and thread is="+Thread.currentThread().getName());
	}
}
=========================================================================
```

Test:-

```
	package com.nit.test;
	import com.nit.dp.ticketBooking;

	public class ticketBookingtest {

	public static void main(String[] args) {
		ticketBooking t=new ticketBooking();
		Thread t1=new Thread(t);
		Thread t2=new Thread(t);
		Thread t3=new Thread(t);
		t1.start();
		t2.start();
		t3.start();
	}
	}
```

Output:-

```
	0 param constructor
	0 param constructor
	0 param constructor
	hascode of printer class is=220585323 and thread is=Thread-0
```

SINGLETON WITH MULTITHREADING

```
hascode of printer class is=1201007699 and thread is=Thread-1
hascode of printer class is=1618742454 and thread is=Thread-2
```

“here above output minimum standred of singleton is breaking by multithreading when more then one thread executing at a time.”

To resolve this issue we have to make thread as synchronized thread

Solution is :- using synchronized and without synchronized