

به نام خدا

آموزش پنجاه و سوم

اهداف آموزشی این قسمت عبارتند از:

۱. نحوه ایجاد یک Thread
۲. آشنایی با Interface یی تحت عنوان Runnable
۳. آشنایی با متدهای run و start

پس از آشنایی با مفاهیم Thread و Interface در دو آموزش گذشته، در این آموزش قصد داریم تا به صورت عملی اقدام به ساخت یک Thread نماییم. به طور کلی می توان گفت که به دو روش مختلف می توانیم اقدام به ساخت یک Thread نماییم:

۱. روش اول به این صورت است که کلاسی ایجاد کرده سپس آن کلاس ویژگی های خود را از کلاس Thread جاوا به ارث ببرد.

۲. روش دوم به این صورت است که کلاسی ایجاد کرده و آن کلاس به اجرای یک Interface تحت عنوان Runnable پردازد (لازم به ذکر است کلاسی که اقدام به اجرای Runnable نماید بدون آنکه خصوصیتی را از کلاس Thread به ارث ببرد خواهد توانست اقدام به ساخت یک Thread نماید).

در ادامه آموزش هر دو روش را مورد بررسی دقیق قرار خواهیم داد.

برای شروع یک پروژه جدید تحت عنوان Session 53rd ایجاد کرده و کلاسی در آن تحت عنوان MyThread ایجاد می کنیم:

```
public class ThreadA extends Thread {  
  
}
```

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی

همانطور که در کد فوق مشاهده می شود کلاسی که تحت عنوان ThreadA ایجاد کردیم کلیه خصوصیات، ویژگی ها و متدهای کلاس Thread جاوا را به ارث خواهد برد. حال کد فوق را به صورت زیر تکمیل می کنیم:

```
public class ThreadA extends Thread {
    @Override
    public void run() {
        System.out.println("Thread A");
        for (int i = 1; i <= 5; i++) {
            System.out.println("From thread A loop no= " + i);
        }
    }
}
```

همانطور که در کد فوق ملاحظه می شود ابتدا متدی تحت عنوان run را وارد برنامه خود کرده ایم. لازم به ذکر است زمانیکه یک کلاس از کلاس Thread ارث بری می کند می بایست این متد را Override کرد که در غیر این صورت اکلیپس دوره واژه run را نقطه چین قرار خواهد داد و چنانچه نشانگر موس خود را روی نقطه چین قرار دهیم، اکلیپس به ما پیشنهاد می دهد که این متد را Override کنیم.

حال از آنجا که قصد داریم علاوه بر کلاسی تحت عنوان ThreadA کلاس دیگری تحت عنوان ThreadB نیز ایجاد کنیم، در متد run دستوری را می نویسیم مبنی بر اینکه عبارت Thread A را در پنجره Console به نمایش در آورد تا بدانیم که خروجی برنامه متعلق به کدام Thread است. حال یک Loop از جنس for تعریف می کنیم (برای آشنایی بیشتر با for به آموزش بیست و دوم مراجعه نمایید). در واقع هر کاری که بخواهیم Thread ما انجام دهد می بایست داخل متد run نوشته شود.

متغیری از جنس int تحت عنوان i ایجاد می کنیم به این شکل که نقطه شروع Loop عدد یک خواهد بود، نقطه پایان Loop کوچکتر یا برابر با 5 و در هر بار Loop شدن یک واحد به متغیر i اضافه خواهد شد. سپس داخل for دستور System.out.println را نوشته سپس داخل پرانتز متد println عبارت: From Thread A loop no: را می نویسیم که حاکی از آن است که "از Thread A به همراه Loop شماره:" را در پنجره Console به نمایش در آورد. سپس یک علامت به علاوه قرار داده و نام متغیری که داخل for ایجاد کردیم را می نویسیم.

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی

حال کلاس دیگری تحت عنوان ThreadB ایجاد کرده و آن را به شکل زیر تکمیل می کنیم:

```
public class ThreadB extends Thread {
    @Override
    public void run() {
        System.out.println("Thread B");
        for (int i = 1; i <= 5; i++) {
            System.out.println("From thread B: loop no = " + i);
        }
    }
}
```

همانطور که ملاحظه می شود این کلاس نیز از کلاس Thread ارث بری می کند و تنها تفاوتی که با کلاس ThreadA دارد این است که هر کجا که حرف A داشتیم به حرف B تغییر یافته است.

حال مجدد اقدام به کلاس دیگری تحت عنوان ActionClass می نماییم و به خاطر داشته باشیم که در حین ساخت این کلاس حتماً گزینه public static void main را تیک بزنیم چرا که این کلاس به منزله نقطه شروع برنامه ما خواهد بود:

```
public class ActionClass {
    public static void main(String args[]) {
    }
}
```

حال نیاز است تا از روی هر کدام از کلاس های ThreadA و ThreadB یک شیء ایجاد کنیم. برای این منظور کد فوق را به صورت زیر تکمیل می کنیم:

```
public class ActionClass {
    public static void main(String args[]) {
        ThreadA a = new ThreadA();

        ThreadB b = new ThreadB();
    }
}
```

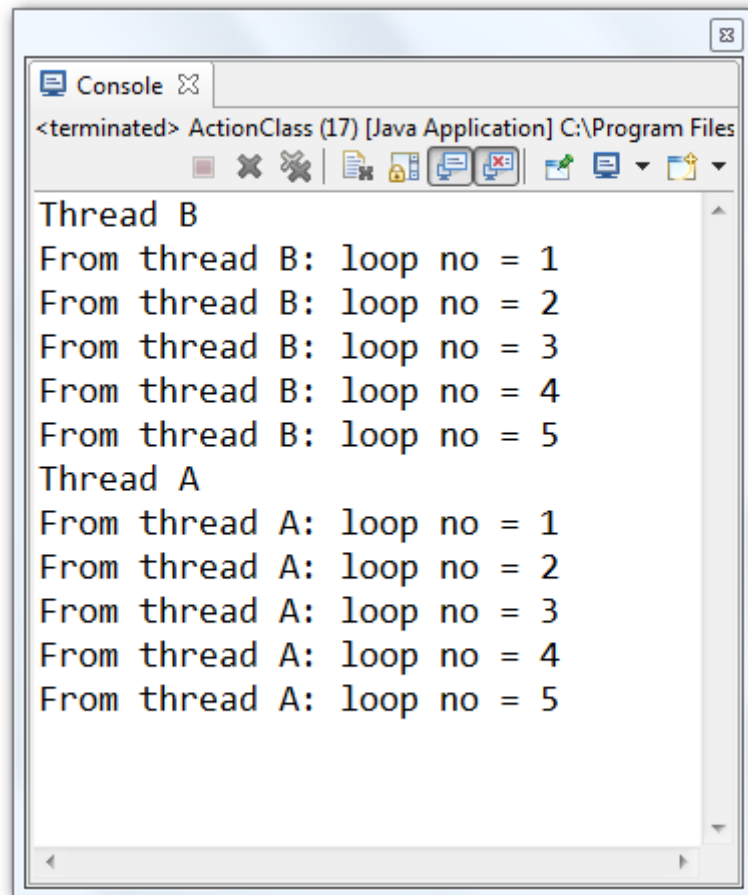
همانگونه که در آموزش های پیشین توضیح داده شد، ابتدا نام کلاس مد نظر را نوشته سپس نامی برای شیئی که می خواهیم از روی آن کلاس ایجاد کنیم در نظر می گیریم. سپس یک علامت مساوی قرار داده و از آنجا که می خواهیم یک شیء جدید ایجاد کنیم می بایست از کلید واژه

new استفاده نماییم. سپس مجدد نام کلاس مد نظر را نوشته و یک علامت (;) پس از آن قرار می دهیم.

اکنون برای آنکه بتوانیم شیئی های ساخته شده از روی کلاس Thread را اجرا نماییم نیاز به متدی تحت عنوان start داریم. برای این منظور همانطور که در کد زیر مشخص است متد start را به شیئی های ساخته شده از روی کلاس های ThreadA و ThreadB ضمیمه می کنیم:

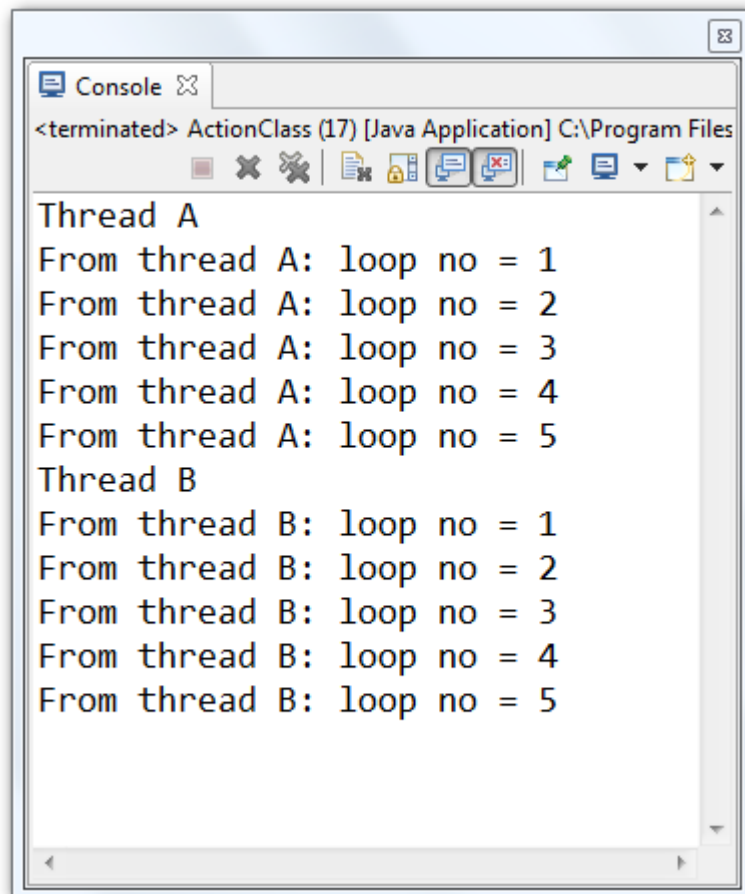
```
public class ActionClass {  
    public static void main(String args[]) {  
        ThreadA a = new ThreadA();  
        a.start();  
        ThreadB b = new ThreadB();  
        b.start();  
    }  
}
```

به طور خلاصه شیئی ساخته شده از روی کلاس ThreadA نامش a است و شیئی ساخته شده از روی کلاس ThreadB نامش b است. حال متد start را به هر دوی این Object ها ضمیمه می کنیم. اکنون اگر برنامه خود را اجرا کنیم با خروجی زیر مواجه خواهیم شد:



```
<terminated> ActionClass (17) [Java Application] C:\Program Files
Thread B
From thread B: loop no = 1
From thread B: loop no = 2
From thread B: loop no = 3
From thread B: loop no = 4
From thread B: loop no = 5
Thread A
From thread A: loop no = 1
From thread A: loop no = 2
From thread A: loop no = 3
From thread A: loop no = 4
From thread A: loop no = 5
```

همانطور که در اجرای فوق می بینیم اول Thread مرتبط با کلاس ThreadB اجرا شده است سپس Thread مرتبط با کلاس ThreadA. حال یک بار دیگر برنامه را اجرا می کنیم:



می بینیم که این بار ترتیب اجرای Thread ها عکس اجرای قبل است. در حقیقت علت این مسئله از آنجا ناشی می شود که در یک برنامه جاوا Thread نه تنها به صورت هم زمان اجرای می شوند بلکه این در حالی است که هر دو Thread مستقل از یکدیگر اجرا می شوند و به همین دلیل نیز می باشد که هر کدام ابتدا شانس اجرا شدن را پیدا کند اول اجرا خواهد شد.

حال فرض کنیم که می خواهیم یک Thread به روش دومی که در بالا به آن اشاره شد ایجاد کنیم. برای این منظور پروژه دیگری تحت عنوان Runnable ایجاد کرده و کلاسی در آن به نام MyThread ایجاد می کنیم:

```
public class MyThread {  
  
}
```

به طور خلاصه می توان گفت که در این روش به منظور ایجاد یک Thread از Interface یی تحت عنوان Runnable استفاده خواهیم کرد. در واقع وظیفه ای که این Interface بر عهده

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی

دارد این است که چنانچه کلاسی این Interface را اجرا کند، آن کلاس بدون آنکه چیزی را از کلاس Thread جاوا به ارث ببرد این امکان را خواهد داشت تا یک Thread ایجاد نماید. حال که با وظیفه Runnable بیشتر آشنا شدیم نیاز است تا کد فوق را به صورت زیر تکمیل کنیم:

```
public class MyThread implements Runnable {  
  
    @Override  
    public void run() {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("From My Thread loop no:" + i);  
        }  
    }  
}
```

همانطور که در کد فوق مشاهده می شود ابتدا کلاس MyThread با استفاده از کلید واژه implements اقدام به اجرای Runnable خواهد کرد. سپس متد مرتبط با ایجاد یک Thread که run نام دارد را وارد برنامه خود می کنیم (به خاطر داشته باشیم برای آنکه بتوانیم از این متد استفاده کنیم می بایست آنرا Override کنیم). سپس همانند Thread های قبلی یک Loop از جنس for ایجاد کرده و کدهای قبلی را برای آن در نظر می گیریم. حال نیاز است تا کلاس دیگری تحت عنوان ActionClass ایجاد کنیم که به منزله نقطه شروع برنامه ما خواهد بود (به خاطر داشته باشیم که گزینه public static void main را تیک دار نماییم):

```
public class ActionClass {  
  
    public static void main(String[] args) {  
  
    }  
}
```

حال بایستی یک شیء از روی کلاسی که تحت عنوان MyThread ایجاد کردیم در این کلاس ایجاد نماییم. برای این منظور کد فوق را به صورت زیر تکمیل می کنیم:

```
public class ActionClass {

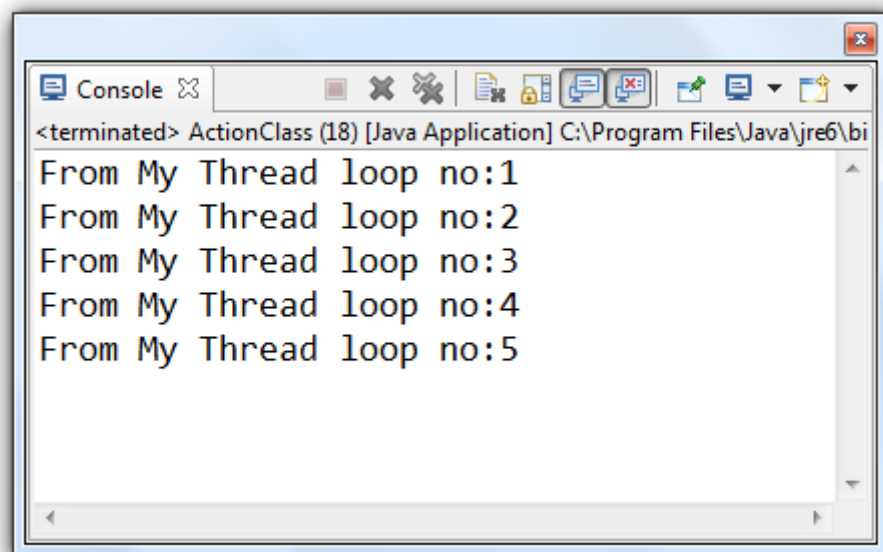
    public static void main(String[] args) {
        MyThread test = new MyThread();
    }
}
```

نام شیئی که از روی کلاس `MyThread` ایجاد کرده ایم `test` است. اگر خاطرمาน باشد در روش اول ساخت یک `Thread` نام شیئی ساخته شده از روی کلاس خود را مستقیماً به متدی `start` ضمیمه می کردیم و `Thread` ما اجرا می شود اما این در حالی است که زمانیکه از `Runnable` برای ایجاد یک `Thread` استفاده می کنیم می بایست شیئی از روی کلاس `Thread` نیز ایجاد کرده، آن شیئی را به شیئی که از روی کلاس خود ساختیم مرتبط ساخته و در نهایت شیئی ساخته شده از روی کلاس `Thread` را به متد `start` ضمیمه کنیم. برای این منظور کد فوق را به صورت زیر تکمیل می کنیم:

```
public class ActionClass {

    public static void main(String[] args) {
        MyThread test = new MyThread();
        Thread thread = new Thread(test);
        thread.start();
    }
}
```

همانطور که ملاحظه می شود یک `Object` از روی کلاس `Thread` جاوا تحت عنوان `thread` ایجاد کرده ایم (اگر توجه کنیم می بینیم که شیئی ساخته شده از روی کلاس `Thread` با حرف کوچک نوشته شده است. به جای این نام می توانیم از هر نام دیگری نیز استفاده کنیم). سپس داخل پرانتز مرتبط با کلاس `Thread` نام شیئی تحت عنوان `test` که از روی کلاس `MyThread` ایجاد کردیم را نوشته و در نهایت شیئی `thread` را به متد `start` که وظیفه دارد یک `Thread` را ایجاد کند ضمیمه می کنیم. حال اگر برنامه خود را اجرا کنیم با خروجی زیر مواجه خواهیم شد:



```
<terminated> ActionClass (18) [Java Application] C:\Program Files\Java\jre6\bin
From My Thread loop no:1
From My Thread loop no:2
From My Thread loop no:3
From My Thread loop no:4
From My Thread loop no:5
```

شاید مطالب این آموزش تا حدودی سخت به و غیر قابل فهم به نظر برسند، اما این در حالی است که در ادامه آموزش ها به خصوص آموزش های توسعه اندروید به طور عملی خواهیم دید که Thread ها چه کاربردهای فراوانی دارند.

پس از مطالعه این آموزش انتظار می رود بتوانیم به سؤالات زیر پاسخ بدهیم:

۱. چند روش برای استفاده از Thread ها وجود دارد؟
۲. وظیفه متد run چیست؟
۳. نام متدی منجر به اجرای یک Thread می شود چیست؟
۴. روش ساخت یک Thread با استفاده از Runnable چگونه است؟