توسعه برنامه های موبایل

جلسه نهم مجازی

بخش دوم

سحر صادقی

آموزش ساخت تایمر معکوس در برنامه نویسی اندروید

دوستان عزیز در این سری از آموزش های برنامه نویس اندروید به آموزش ساخت تایمر معکوس در برنامه نویسی اندروید می پردازیم . تایمر در برنامه نویسی اندروید خیلی کاربردی است به طور مثال می توانید برای فروش یک محصول off **بزنید و از تایمر استفاده کنید قبل تر ما آموزش های مربوط به تایمر قرار داده بودیم کافی است در سایت سرچ کنید در ادامه با ما همراه باشید.**

برای ایجاد تایمر معکوس یا تایمر CountDown **همانند زیر عمل می کنیم.**

```
CountDownTimer cTimer = null;

void startTimer() {
  cTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished)  {

     Toast.makeText(getApplicationContext(),millisUntilFinished                    ,
Toast.LENGTH_LONG);

    }
    public void onFinish() {
        Toast.makeText(getApplicationContext(), "پایان", Toast.LENGTH_LONG);
    }
  };
  cTimer.start();
}


void cancelTimer() {
  if(cTimer!=null)
    cTimer.cancel();
}
```

ورودی این تایمر دو مقدار است اولی زمانی که می خواهید معکوس شود و میلی ثانیه است یعنی برای ۱۰ ثانیه باید ۱۰۰۰۰ را وارد کنید و دومی مقدار زمانی است که می خواهید از عدد اولی کم شود

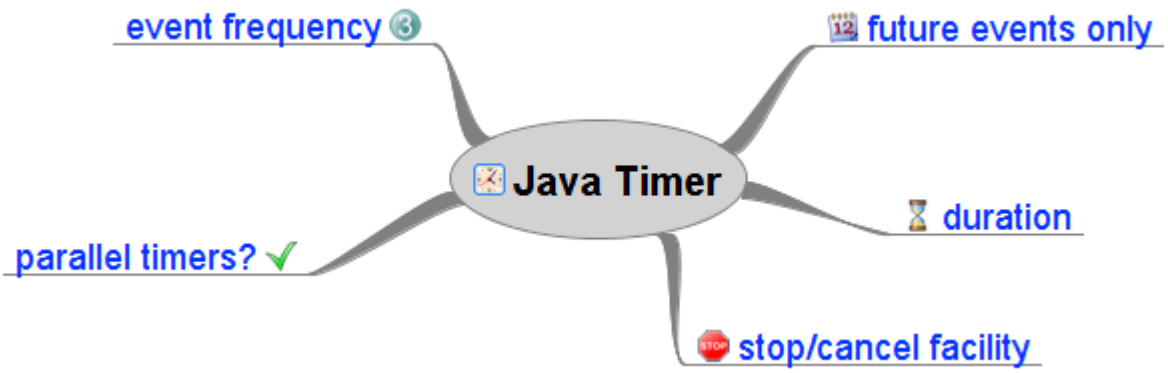
به طور مثال الان بروی یک ثانیه تنظیم شده است (۱۰۰۰ میلی ثانیه) اگر می خواهید هر دو ثانیه از آن یک عدد کم شود آن را به ۲۰۰۰ زیاد کنید و مقدار تایمر که از آن کم می شود در متغییر millisUntilFinished **قرار می گیرد و** void **و زمانی که به پایان می رسد وارد** void **ی به نام** onFinish **می شود و بخش مهم کار** void

دوم یا CancelTimer **هست اگر از این استفاده نکنید باعث** memory leak **می شود و سبب ایجاد خطا می شود و در آخر برای اجرای آن در بعد از** Oncreate **باید** void **را اجرا کنید.**

و در آن void **بررسی می کنیم که آیا تایمر** null **یا خالی شده است اگر خالی بود تایمر را** cancel **یا لغو می کنیم.**

run
1 startTimer();

کلاس های Timer و TimerTask در جاوا



از ویژگی های پکیج java.util **توانایی زمانبندی کارها می باشد. در جاوا پکیج** util **با گنجاندن دو کلاس** Timer **و** TimerTask **، امکان زمانبندی کارها را بسیار ساده کرده است و به شما این اجازه را می دهد که بدون درگیر شدن با مفاهیم** MutliTasking **و** MultiThreading **، به سادگی یک عمل را در زمان تعین شده به صورت خودکار انجام دهید.**

در اصل کار اصلی Timer **و** TimerTask **در جاوا ایجاد یک** Thread **در پس زمینه است که منتظر است تا زمان تعین شده فرا برسد تا کار خاصی را که به او محول شده را انجام دهد. این دو کلاس امکانات متنوعی به شما می دهند تا بتوانید کار را برای اجرا در زمانی مشخص و یا اجرا به صورت پی در پی زمانبندی کنید.**

کلاس های Timer **و** TimerTask **با هم کار می کنند.** Timer **وظیفه زمانبندی کار را بر عهده دارد و** TimerTask **وظیفه انجام کار معین شده در هنگام فرا رسیدن زمان مشخص شده را بر عهده دارد.** Timer **با استفاده از نمونه ای از** TimerTask **کاری را در زمان مشخصی انجام می دهد. پس باید نمونه ای از** TimerTask **ایجاد کنیم و سپس آن را با ایجاد نمونه ای از** Timer **زمانبندی کنیم.**

کلاس TimerTask:

این کلاس یک کلاس abstract **می باشد که متد** run() **را برای پیاده سازی توسط شما به صورت** abstract **تعریف کرده است. برای اینکه بتوانیم از این کلاس نمونه ای بسازیم مجبوریم که این متد را پیاده سازی کنیم. بهترین روش برای این کار ایجاد کلاسی است که از کلاس** TimerTask **مشتق می شود (ارث بری دارد) و متد** run() **را پیاده سازی می کند.**

در زیر کلاسی را می بینید که از کلاس TimerTask **ارث بری کرده و متد** run() **را پیاده سازی می کند.**

این کلاس یک Object **از نوع** String **دریافت کرده و وظیفه دارد به در زمانی که کلاس** Timer **تعیین میکند متد** run() **را اجرا و** String **و تعداد کلماتش را در خروجی چاپ کند.**

```
import java.util.TimerTask;

class MyTimerTask extends TimerTask {

    private String str;

    public MyTimerTask(String str) {
        this.str = str;
    }

    public void run() {
        String words[] = str.split(" ");
        System.out.println("\n" + str);
        System.out.println("number of words: " + words.length);
    }
}
```

کلاس Timer :

پس از TimerTask **که همان کار مورد نظر است که باید در زمان مشخص انجام شود، باید امور مربوط به زمانبندی کار را انجام دهیم. همانطور که گفته شد ، با استفاده از کلاس** Timer **این کار را انجام میدهیم.**

کلاس Timer **چهار نمونه** constructor **مختلف دارد که چون غیر از نوع اول ، سه نوع بعدی کاربرد خیلی کمتری دارند ، ما فقط به بررسی نوع اول می پردازیم.**

```
public Timer()                                          1
public Timer(boolean isDaemon)                          2
public Timer(String name)                               3
public Timer(String name, boolean isDaemon)             4
```

برای زمانبندی کار ایجاد شده، باید یک Object **از نوع** Timer **ایجاد کنیم و با استفاده از متد** schedule **تنظیمات مربوط به زمان بندی را اعمال کنیم.**

متد schedule **چهار نگارش مختلف دارد. که با هر کدام برای کار بخصوصی ایجاد شده اند.**

```
public void schedule(TimerTask task, long delay)        1
```

این نسخه از متد schedule **، با گرفتن شی** TimerTask **و زمان تاخیر(** delay **) ، کار مورد نظر را بعد از انتظار به اندازه** delay **انجام میدهد.(** delay **زمانی بر حست میلی ثانیه است)**

```
public void schedule(TimerTask task, Date time)         1
```

این نسخه از متد schedule **با گرفتن شی** TmerTask **و یک شی از کلاس** Date **کاری را دقیقا در یک تاریخ مشخص که این تاریخ در همین شی** Date **کپسوله شده است انجام می هد.**

```
public void schedule(TimerTask task, long delay, long period)    1
```

این نسخه از متد schedule **با گرفتن شی** TmerTask **علاوه بر انجام کار های نسخه اول، زمان تکرار کار را هم که با متغییر** period **مشخص شده را بر حسب میلی ثانیه می گیرد و بعد از تاخیر به اندازه** delay **، بعد از گذشت زمان** period**، کار مشخص شده در** TimerTaks **را مدام تکرار می کند.**

```
public void schedule(TimerTask task, Date firstTime, long period)
```

آخرین نسخه از متد schedule **کاری مشابه نسخه سوم انجام می دهد با این تفاوت که زمان** delay **برای انتظار اولیه را در شی** Date **دریافت می کند.**

- **نکته: برای خاتمه دادن به زمانبندی باید با از متد cancel() استفاده کنیم. وگرنه Thread مربوط به این شی از بین نخواهد رفت. و برنامه همچنان باز خواهد ماند.**

مثال

در زیر مثالی را می بینید که برای استفاده از کلاس Timer **نوشته شده و با به خدمت گرفتن کلاس** MyTimerTask **(که در بالا پیاده سازی این کلاس را دیدیم) ، کاری را که در این کلاس تعریف کرده ایم را**

زمانبندی می کند. در این مثال با استفاده از نسخه سوم متد schedule**، کار تعیین شده در** MyTimerTask **را بعد از تاخیر اولیه ۵۰۰ میلی ثانیه ، و سپس تکرار کار در هر ۱۰۰۰ میلی ثانیه یکبار، اجرا میکنیم.و پس از گذشت ۵۰۰۰ میلی ثانیه** cancel() **را فراخوانی کرده و به کار** Timer **خاتمه می دهیم.**

```
import java.util.Timer;

class Main {

    public static void main(String args[]) {
        String str = "hi. welcome to neo-one.ir. this lesson is Timer and TimerTask learning in java";
        Timer timer = new Timer();
        timer.schedule(new MyTimerTask(str), 500, 1000);
        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {}
        timer.cancel();
    }
}
```

<u>ایجاد تایمر در برنامه نویسی اندروید</u>

برای اینکه یک سری کد به صورت چند ثانیه یک بار انجام بشود باید از تایمر استفاده کرد.

در برنامه نویسی اندروید با ایکلیپس کدهای تایمر به صورت زیر میباشد:

```
Timer timer= new Timer();

timer.scheduleAtFixedRate(new TimerTask(){

                    @Override
```

```
public void run() {
        // TODO Auto-generated method stub
         runOnUiThread(new Runnable() {
    public void run() {
        //what you want to do
    }
});


}


}, 100, 10000);
```

در کد بالا ۱۰۰۰۰مدت زمان وقفه میباشد که بر حسب میلی ثانیه میباشد.