פרק 18 - אנדרואיד Thread

לכל יישום המבקשת לרוץ, המערכת מקצה תהליכון ראשי שנקרא main thread (נקרא גם ui thread, תהליכון ממשק משתמש).

תהליכון זה אחראי על ממשק המשתמש הגרפי.

מה זה אומר תהליכון?

תהליכון (process) הוא קוד שמכיל הרבה threadים ומבקש לרוץ.

דוגמה לתהליכון ראשי: Activity.

1. התהליכונים הפועלים (threads Worker) תהליכון לביצוע משימה מקבילה לתהליכון הראשי בלי לחסום את ריצתו או לשבש את ריצתו. נעזרים בדרך כלל בפעולות הבאות
   - Activity.runOnUiThread(Runnable)
   - View.post(Runnable)
   - View.postDelayed(Runnable, long)
2. שינוי ממשק המשתמש מתוך Thread ע"י **Handler** בשיטת שליחת ההודעות.
3. שינוי ממשק המשתמש מתוך Thread ע"י משימה אסינכרונית **AsyncTask**: מאפשרת לבצע משימה אסנכרונית על ממשק המשתמש כך שמבצעת את הפעולות החוסמות ב "תהליכון הפועל" ואז מעדכנת את התוצאות על תהליכון ממשק המשתמש " UI thread", בלי לטפל בתהליכונים.
4. responsiveness היא דאגה עיקרית של אנדרואיד
5. אנדרואיד נותן 5 שניות כדי להפעיל יישום או מציג שגיאה.

בהמשך נלמד על Handler ועל AsyncTask.

לסיכום UI Thread

1. לכל אפליקציה קיים main thread שאחראי לביצוע כל האירועים של האפליקציה
2. ה-main thread אחראי לאינטראקציה עם ה-ui ולכן הוא נקרא ui thread
3. במידה וה-main thread ימתין יותר מ-5 שניות האפליקציה תקרוס
4. לכן כאשר יהיו לנו משימות מורכבות לא נעשה אותן ב-thread הראשי אלה נפנה אותו ל-thread נפרד שיבצע זאת במקביל ל-ui thread.

1. ל-thread אסור לבצע שינויים בממשק המשתמש - אנדרואיד לא מאפשר זאת.
2. רק ה-thread הראשי רשאי לבצע שינוי ב-ui.
3. כל יישום מורכב מ-
   a. Main thread
   b. thread'ים נוספים
4. אינטראקציה בין ה-thread'ים מתבצעת ב-2 אפשרויות
   a. אובייקט Handler
   b. AssynTask

AssyncTask

1. המחלקה מורכבת מטיפוסים גנריים - params,Progress,Result

AsyncTask שיפעל יותר מכמה שניות יכול להיות שיהרג על ידי מערכת הפעלה.

ל- AssyncTask יש 4 פונקציות עיקריות

1. onPreExecute - תהליך שמטפל במשימות גדולות.לא ניתן לבצע בו עדכון ui
2. RunInBackGround - תהליך שנעשה ברקע אך לא ניתן לעדכן בו את ממשק המשתמש
3. onProgressUpdate - ניתן לשנות בעזרתו את ממשק המשתמש
4. onPostExecute - מקבל כפרמטר את result שמוחזר על ידי הפונקציה RunInBackGround.ומעדכן את ממשק המשתמש.

ל- AsynTasks יש גם פעולת עזר publishProgress שתפקידה לבצע את העדכונים.

<AsyncTask<Params,Progress,Result

params - זה מערך של נתונים שנקבל ביצירה של האובייקט.לדוגמה מערך של תמונות להורדה.
Progress - סוג הנתונים של יחידות ההתקדמות
Result - סוג הנתונים של התוצאה.סוג הנתונים שתחזיר פונקציית doBackGround.

במידה ונסמן Void - זהו טיפוס ללא שימוש

במידה ואנו רואים סימון של ...(שלוש נקודות)זה מסמל מערך.

בדוגמה הראשונה שלנו נבנה שעון בעזרת AssyncTask.

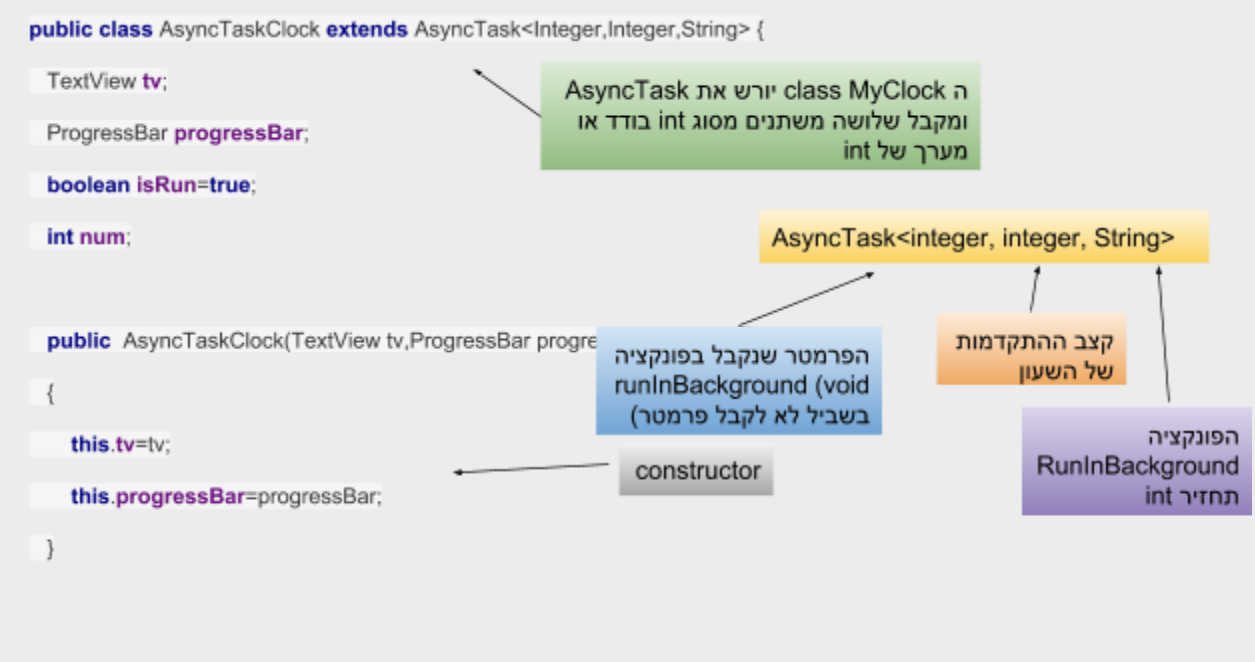
שלב 1 - עיצוב MainActivity:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

  xmlns:android="http://schemas.android.com/apk/res/android"

  xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"

  android:layout_height="match_parent"
```

```
  android:orientation="vertical"
tools:context=".MainActivity">

  <TextView
      android:text="AsyncTask!"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:textSize="30sp"
      />
  <TextView
      android:layout_width="200dp"
      android:layout_height="200dp"
      android:id="@+id/tv"
      android:text="10"
      android:textSize="50sp"
      android:layout_gravity="center"
      android:gravity="center"
      android:textColor="#fa0b19"
      android:layout_marginTop="30sp"
      />
```

```
  <ProgressBar
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      style="?android:attr/progressBarStyleHorizontal"
      android:id="@+id/progress"
      />
  <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/btnStart"
      android:text="play"
      />
  <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/btnStop"
      android:text="stop"
      />
</LinearLayout>
```

שלב 2 - בניית class שיורש מ AssyncTask:



```
@Override
  protected void onPreExecute() {
     super.onPreExecute();
  }

@Override
  protected String doInBackground(Integer... params) {
     num=params[0];
     while (num>0){
        try {
           Thread.sleep(1000);
        } catch (InterruptedException e) {
           e.printStackTrace();
        }
        if(isRun){
           num--;
           publishProgress(num);
        }
        if(num==0)
           isRun=false;
     }
     return "timer finished " + num;
  }
```

הפעלת הפונקציה
OnProgressUpdate

```
    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        tv.setText(String.valueOf(values[0]));
        progressBar.setProgress(values[0]);
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        tv.setText(s);
    }
}
```

נשנה את הטקסט של הכפתור

מופעל בסיום ה Thread

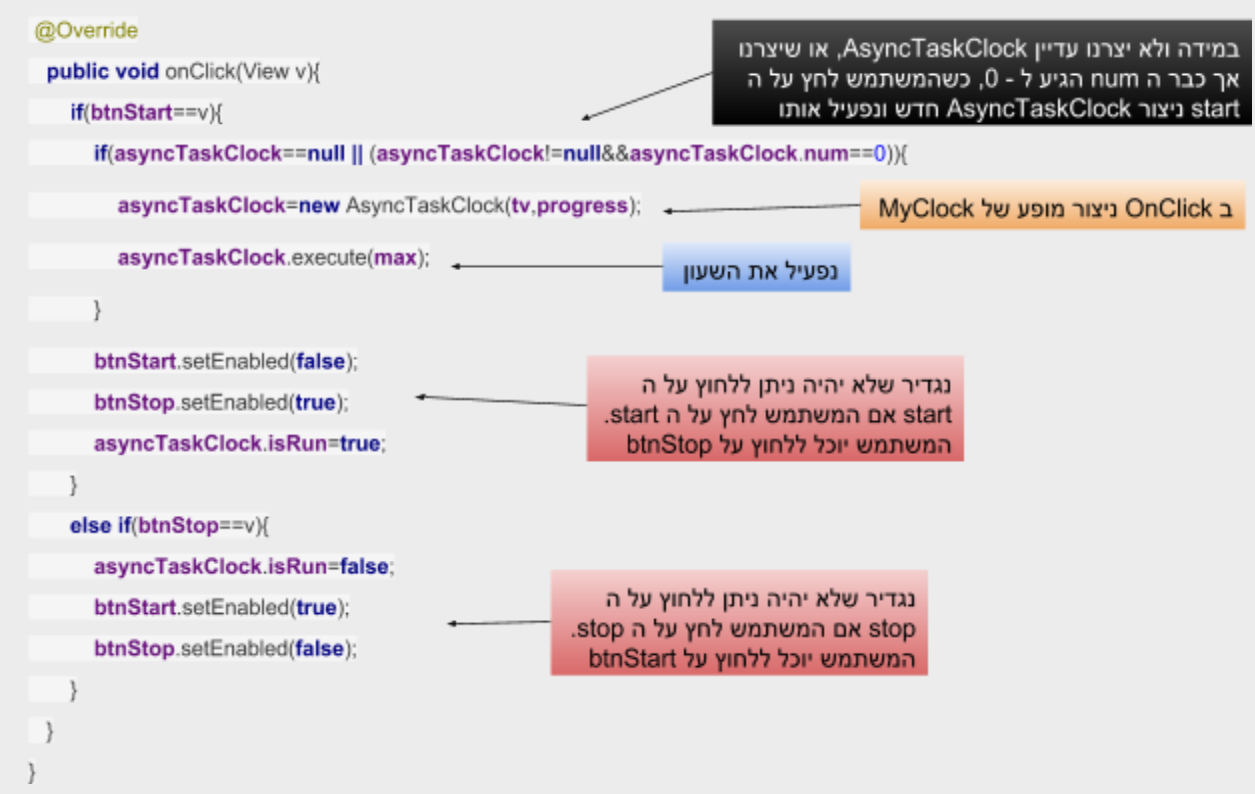
שלב 3 - מימוש הקוד ב MainActivity:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    TextView tv;

    ProgressBar progress;

    Button btnStop,btnStart;

    int max=10;

    AsyncTaskClock asyncTaskClock;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);


        btnStart=(Button)findViewById(R.id.btnStart);

        btnStop=(Button)findViewById(R.id.btnStop);

        btnStop.setOnClickListener(this);

        btnStart.setOnClickListener(this);
```

נגדיר ProgressBar בשביל לראות את כל התהליך בפעולה

הצהרה על משתנים

הפניה לאובייקטים והאזנה לכפתור

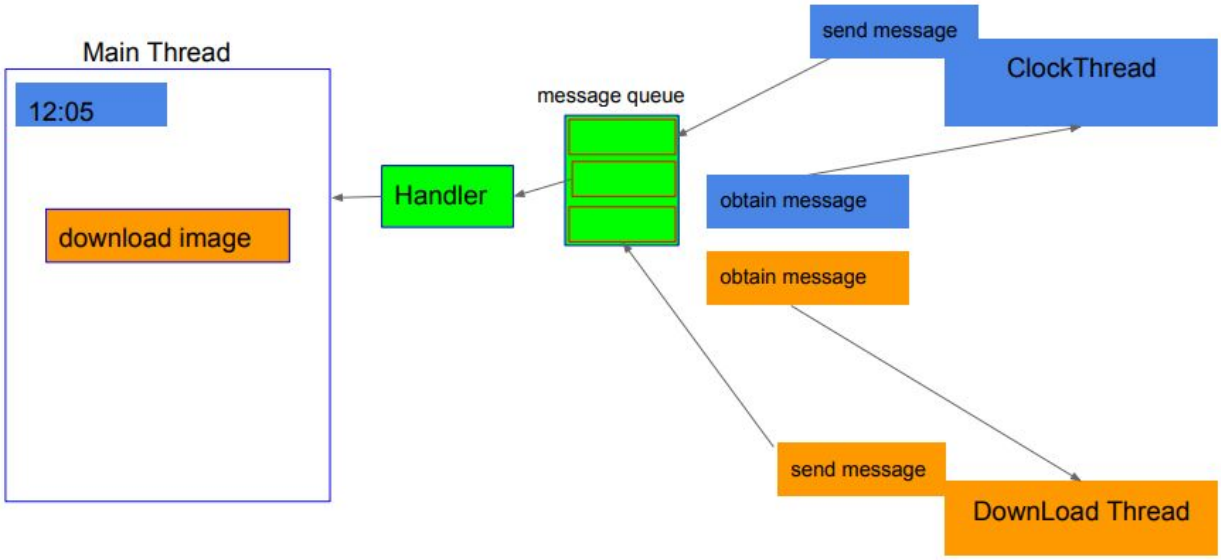


הריצו את האפליקציה! :)

Handler

ה Handler מאפשר לנו לתווך בין ה Thread שאנחנו יוצרים ל main Thread.

1. מתווך בין ה-main thread לשאר ה-thread'ים.
2. ה-handler מכיל תור של מסרים.כמו כן ה-handler מקושר לתור המסרים שיצר אותו.תפקידו של ה-handler לתזמן את המסרים ולהכניס פעולות לתור המסרים על מנת שיתבצעו ב-thread הראשי.
3. כל Thread שולח הודעות לתור ההודעות של ה-handler
4. תפקידו של ה-handler לשלח מסרים ואובייקטי runnable ל-main thread ולבצע את מסר שקיבל מכל thread ו-thread
5. ה-handler נעזר בשתי פונקציות
   a. sendMessage
   b. obtainMessage



דוגמה ל - Handler שעון שרץ אחורנית

שלב 1 - עריכת MainActivity:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
tools:context=".MainActivity">

  <TextView
     android:text="Handler!"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_gravity="center"
     android:textSize="30sp"
     />
  <TextView
     android:layout_width="200dp"
     android:layout_height="200dp"
     android:id="@+id/tv"
     android:text="10"
     android:textSize="50sp"
     android:layout_gravity="center"
     android:gravity="center"
     android:textColor="#fa0b19"
     android:layout_marginTop="30sp"
     />

  <ProgressBar
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     style="?android:attr/progressBarStyleHorizontal"
     android:id="@+id/progress"
     />
  <Button
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
```

```
    android:id="@+id/btnStart"
    android:text="play"
    />
  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnStop"
    android:text="stop"
    />
</LinearLayout>
```

שלב 2 - נבנה את class ThreadClock:

```
public class ThreadClock extends  Thread {
  int num;
  boolean isRun=true;
  Handler handler;

  public ThreadClock(Handler handler,int num){
    this.handler=handler;
    this.num=num;
  }
  @Override
  public void run() {
    super.run();

    while(true){
      if(isRun&&num>=0){
        try {
          Thread.sleep(1000);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        Message msg = new Message();
        msg.arg1 = num;
        num--;
        handler.sendMessage(msg);
      }
    }
  }
}
```

יורשים מ Thread

constructor

נשלח את ה num
בהודעה ב handler

שלב 3 נבנה את ה- MainActivity:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
  TextView tv;

 ProgressBar progress;
 Button btnStop,btnStart;
 Handler handler;
 int max=10;
 ThreadClock threadClock;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btnStart=(Button)findViewById(R.id.btnStart);
    btnStop=(Button)findViewById(R.id.btnStop);
    btnStop.setOnClickListener(this);
    btnStart.setOnClickListener(this);

    progress=(ProgressBar)findViewById(R.id.progress);
    tv=(TextView)findViewById(R.id.tv);
    progress.setMax(max);
    progress.setProgress(max);
    btnStop.setEnabled(false);

    handler=new Handler(new Handler.Callback() {
       @Override
       public boolean handleMessage(Message msg){
          tv.setText(String.valueOf(msg.arg1));
          return true;
       }
    });
}
```

הצהרה על משתנים

הפניה לאובייקטים והאזנה לכפתורים

ניצור Handler חדש שיקבל את ההודעה ששלחנו מהThreadClock. נשנה את הטקסט של ה TextView להיות התוכן של ההודעה ששלחנו

```
@Override
public void onClick(View v) {
   if(btnStart==v){
      btnStart.setEnabled(false);
      btnStop.setEnabled(true);
      if(threadClock==null || (threadClock!=null&&threadClock.num==0)){
         threadClock = new ThreadClock(handler, max);
         threadClock.start();
      }

      btnStart.setEnabled(false);
      btnStop.setEnabled(true);
      threadClock.isRun=true;
   }

   else if(btnStop==v){
      btnStart.setEnabled(true);
      btnStop.setEnabled(false);
      threadClock.isRun=false;
    }
 }
}
```

בלחיצה על btnStart במידה ולא יצרנו עדיין threadClock, או שיצרנו וה num שלו הגיע ל - 0, יווצר לנו מופע של threadClock.
מיד לאחר מכן נפעיל את ה thread

הריצו את האפליקציה :)