Android השתלמות מתחילים
Background-עבודה ברקע

אלון חימוביץ

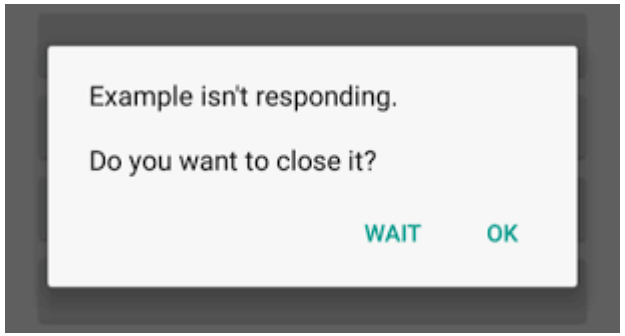
מטרות

- היכרות עם המערכת והבנה של תהליכים שקורים מאחורי הקלעים
- מהו Threadוהצורך בו
- עבודה עם Thread
- דרכי מימוש ב Android
- דרכים לעדכון מסך המשתמש

רקע

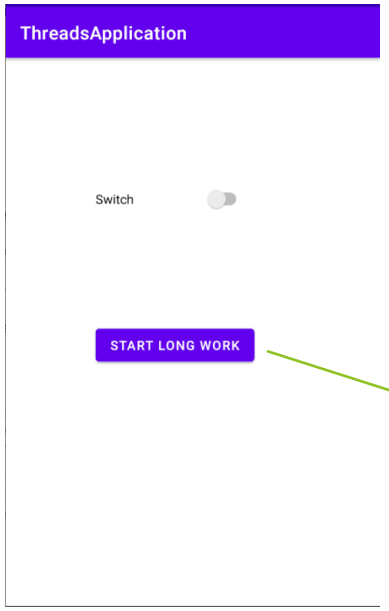
- אנדרואיד מתחיל תהליך Process -לאפליקציה בעת שמתחילה
  - כל אפליקציה היא תהליך נפרד באנדרואיד
- כל רכיבי האפליקציה -רצים באותו תהליך
- יש Threadבודד שמריץ את הרכיבים UI Thread / MainThread -
- כל עדכוני המסך ותגובות לפעילות המשתמש מתבצעים מה UI Thread
- מגבלה על עומס העבודה שתתבצע ב UI Thread
- ANR

ANR



- באנדרואיד קיים מנגנון שבודק שאפליקציה מגיבה
- חלון כאשר אפליקציה אינה מגיבה למשך זמן נתון לארוע
- כאשר מבצעים פעולה ארוכה על ה UIThread
  - הורדת קובץ מהרשת
  - כתיבת נתונים ל DATABASE

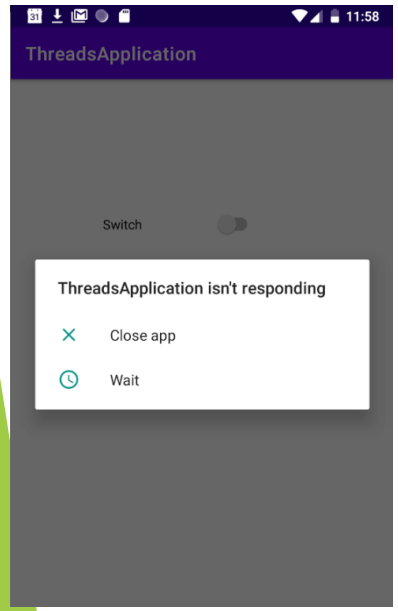
דוגמה



לחיצה על הכפתור תזמן את הפעולה startLongWork

```java
public class LongWorkOnActivity extends AppCompatActivity {

    private static final String TAG = "LongWorkOnActivity ";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_long_work_on);
    }
    public void startLongWork(View view)
    {
        for (int i = 0; i <5 ; i++)
        {
            SystemClock.sleep( ms: 6000);
            Log.d(TAG, msg: "startLongWork: " + i);
        }
    }
}
```

Sleep - מדגים גישה לרשת או פעולה ארוכה אחרת שמתבצעת מה UIThread



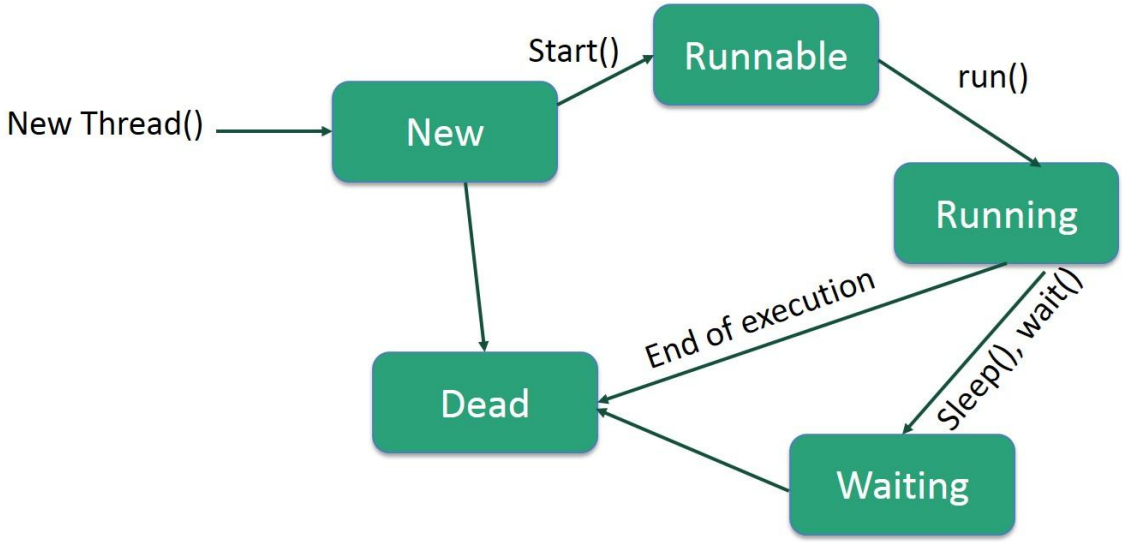
ANR
נסו להזיז את ה switch לפני ולאחר לחיצה על הכפתור

Thread - עבודה במקביל

- יצירת Thread מאפשרת לבצע פעולות במקביל
- נקרא / Worker Thread Background
- פעולות ארוכות נוציא מה UIThread ונעביר ל Worker Thread
- האפליקציה תמשיך להגיב לארועי משתמש
- לא נקבל ANR מאנדרואיד

Thread ב JAVA -

- אנדרואיד מספקת מספר מחלקות שעוטפות את העבודה עם Threads
- עם זאת חשוב להכיר את העבודה עם Threads באופן ישיר לטובת הבנה
- ניתן להגדיר Thread בשתי דרכים
  - לרשת ממחלקה Thread
  - מימוש ממשק Runnable



מקור: https://www.tutorialspoint.com/java/java_multithreading.htm

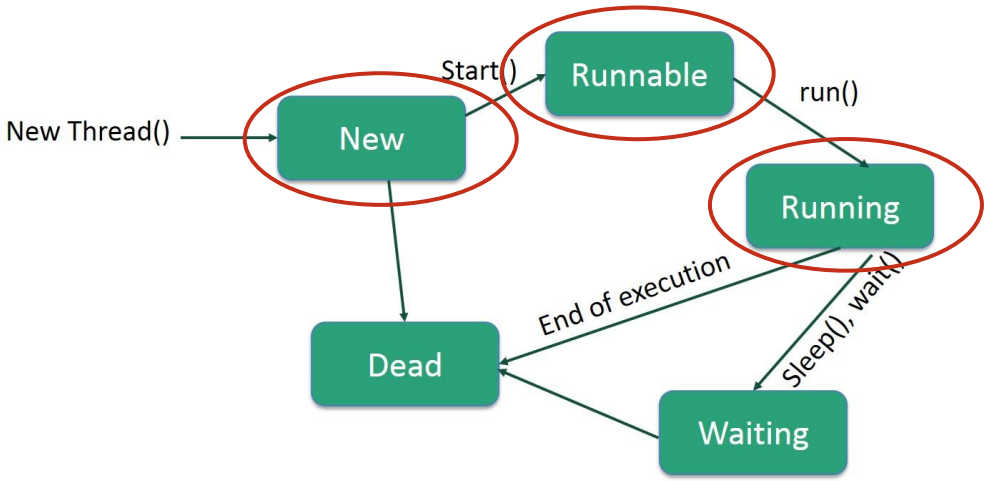
Thread - Extends Thread תהליך יצירת

```java
public class LongWorkOnThreadActivity extends AppCompatActivity {

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

        ThreadExample threadExample = new ThreadExample();
        threadExample.start();
    }
```

יצירת מופע

הפעלת ה Threadהפעולה run תרוץ במקביל ל Activity

```java
public class ThreadExample extends Thread{
    private static final String TAG = "ThreadExample";
    @Override
    public void run() {
      //  super.run();
        for (int i = 0; i <5 ; i++)
        {
            SystemClock.sleep( ms: 6000);
            Log.d(TAG,  msg: "startLongWork: " + i);
        }
    }
}
```



מקור: https://www.tutorialspoint.com/java/java_multithreading.htm

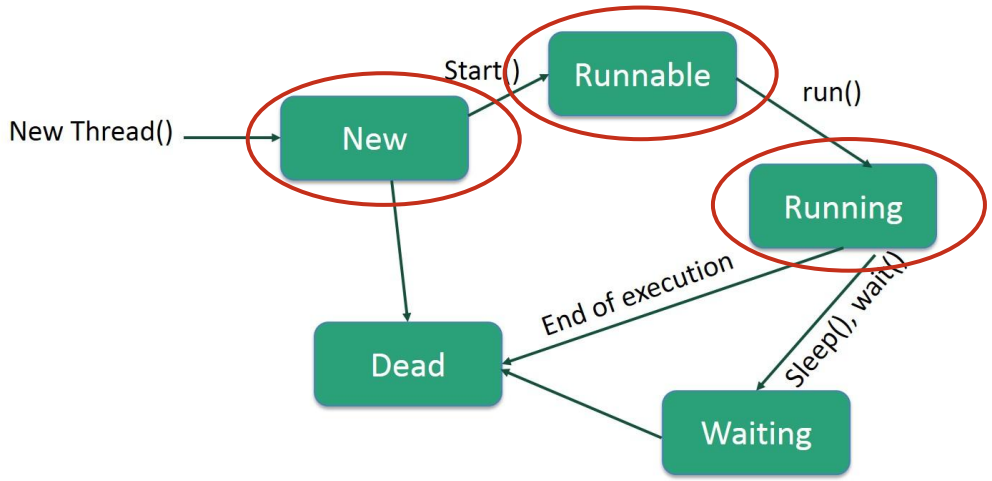
Thread - Runnable Implements תהליך יצירת

```java
public class LongWorkOnThreadActivity extends AppCompatActivity {

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


        // Runnable יצירת מופע של
        ThreadExample threadExample = new ThreadExample();
        //  Thread ל  Runnable העברת המופע של
        Thread t = new Thread(threadExample);
        // Thread הפעלת ה
        // ThreadExample במחלקה run הפעולה
        t.start();
    }
```

יצירת מופע

הפעלת ה Thread הפעולה run תרוץ במקביל ל Activity

```java
public class ThreadExample implements Runnable{
    private static final String TAG = "ThreadExample";
    @Override
    public void run() {
      //  super.run();
        for (int i = 0; i <5 ; i++)
        {
            SystemClock.sleep( ms: 6000);
            Log.d(TAG,  msg: "startLongWork: " + i);
        }
    }
}
```



מקור: https://www.tutorialspoint.com/java/java_multithreading.htm

מימוש באמצעות מחלקה אנונימית

```
public class LongWorkOnThreadActivity extends AppCompatActivity {

    private static final String TAG = "LongWorkUsingThread";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_long_work_on_thread);
        // הרצת Thread באמצעות מחלקה אנונימית
        new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i <5 ; i++)
                {
                    SystemClock.sleep( ms: 6000);
                    Log.d(TAG,  msg: "startLongWork: " + i);
                }
            }
        }).start();
    }
```

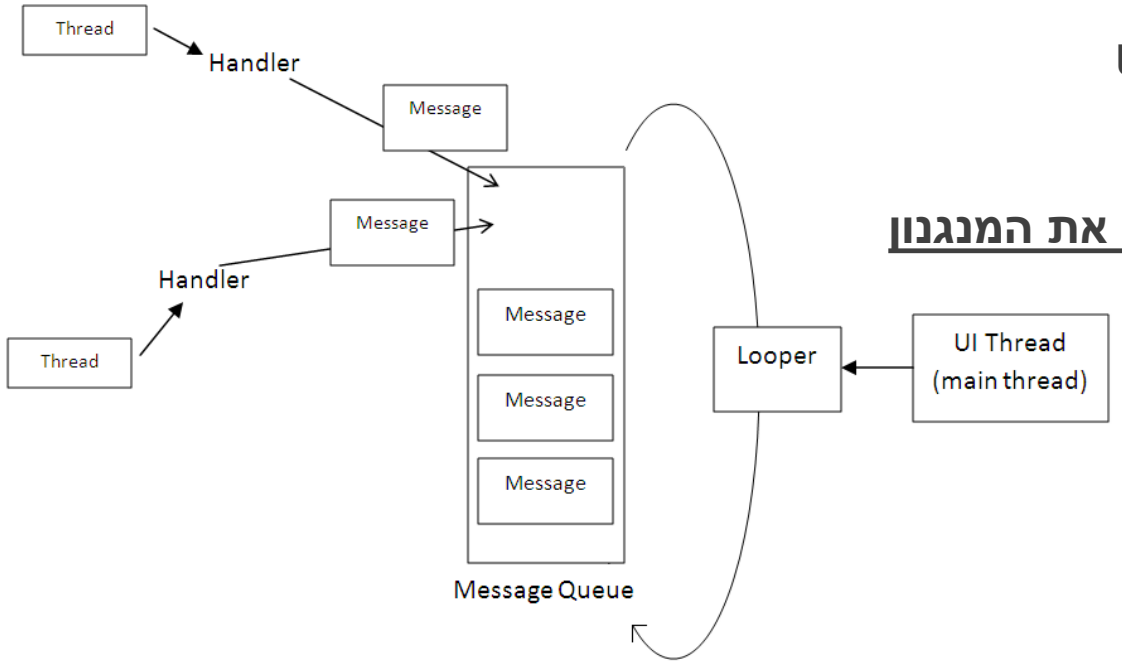
עדכון התצוגה מתוך ה Thread

- לעיתים נרצה לעדכן את התצוגה במידע שהגיע מה thread
  - עדכון רכיב ImageViewבתמונה שירדה מהרשת באמצעות thread
  - עדכון רכיב תצוגה במשך זמן שנותר
  - עדכון progress barבהתאם לכמות הזמן שנותרה לפעולה

- שימו לב -אי אפשר לעדכן רכיבי תצוגה מחוץ ל UIThread
  - עדכון רכיב תצוגה מחוץ ל UIThreadזורק חריגה והאפליקציה תקרוס

רקע ומושגים -כיצד המנגנון עובד

- MessageQ -תור הודעות שקיים לכל MainThread/UIThread
- Looperמחלקה שתפקידה להוציא הודעות מהתור ולהעבירן ל UIThread
  - לולאה אינסופית שדואגת שריצת ה UI Threadלא תסתיים
- העברת מידע מ Threadל - UIThreadחייב להתבצע דרך תור ההודעות
  - מידע נכנס לתור ההודעות מתוך ה Thread
  - מידע יוצא מהתור ורץ מתוך ה UI Thread
- מתבצע באמצעות מנגנון שנקרא Handler
- **<u>יש מחלקות מוכנות באנדרואיד שעוטפות את המנגנון</u>**



מקורות:

https://guides.codepath.com/android/managing-threads-and-custom-services#processing-messages-on-handlerthread

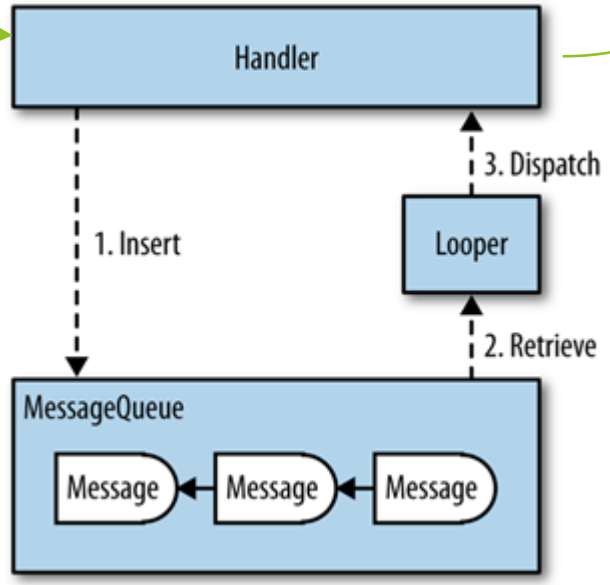
Handler

Threadשמבקש לעדכן את ה UIThreadמזמן פעולה על אובייקט שמכניסה הודעה לתור ההודעות דרך ה Handler

```
handler.sendMessage
)m);
```

```
handler = new Handler (){
    @Override
    public void handleMessage(@NonNull Message
msg}(
{
{;
```

ההודעה יוצאת מתור ההודעות ומתבצעת בפעולה בצד של ה - UI Thread בדרך זו ניתן להעביר מידע ו/או לעדכן רכיבי תצוגה



מקורות:

https://www.oreilly.com/library/view/efficient-android-threading/9781449364120/ch04.html#section_looper

המחלקה Message

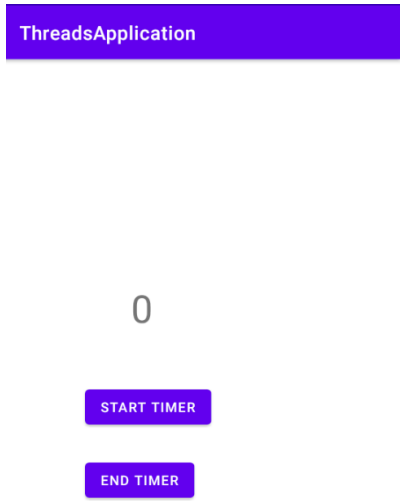
public int	arg1 arg1 and arg2 are lower-cost alternatives to using setData() if you only need to store a few integer values.
public int	arg2 arg1 and arg2 are lower-cost alternatives to using setData() if you only need to store a few integer values.
public Object	obj An arbitrary object to send to the recipient.
public Messenger	replyTo Optional Messenger where replies to this message can be sent.
public int	sendingUid Optional field indicating the uid that sent the message.
public int	what User-defined message code so that the recipient can identify what this message is about.

Handler - כיצד נראה ב JAVA

```java
private Handler handler;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_long_work_on_thread_and_update_ui);
    initViews();
    createHandler();

}

private void createHandler()
{
    handler = new Handler(Looper.getMainLooper()){
        @Override
        public void handleMessage(@NonNull Message msg) {
            // better use as a constant or enum
            if(msg.what== 1)
                // this means update image
                imageView.setImageResource(R.drawable.dog);
        }
    };
}
```

```java
new Thread(new Runnable() {
    @Override
    public void run() {
        for (int i = 0; i <5 ; i++)
        {
            SystemClock.sleep( ms: 1000);
            Log.d(TAG,  msg: "startLongWork: " + i);

        }
        // can be new Message();

        Message message = Message.obtain();
        // type of message sent
        // use as a constant or enum
        message.what=1;
        handler.sendMessage(message);
    }
}).start();
```

תרגיל :שעון עצר



- ניצור אפליקציה שתפעל כשעון עצר
- הגדירו במסך 2כפתורים ו TextViewאחד
- לחיצה על כפתור Startתריץ threadשכל שנייה יעדכן את המסך בערך המעודכן
- לחיצה על כפתור Stopעוצרת את שעון העצר
- לחיצה על Startמחדשת את הריצה
- לחיצה נוספת על Startבמהלך הריצה מבצעת Pauseולאחר מכן Resume
  - אפשר במקום להוסיף כפתור Pause/Resumeשמשהה ומחדש את ריצת השעון

- רשות -הזיזו את מיקום הכפתורים -מעלה או מטה 10 pixelsכל שנייה

עדכון תצוגה על ידי runOnUiThread & post

- שתי דרכים פשוטות לעדכון התצוגה ,שימו לב ש runOnUiThread וpostמעבירות את הקוד להתבצע ב - UIThreadמאפשר עדכון תצוגה)מבלי להבין את המנגנון מהשקף הקודם(...

```
new Thread(new Runnable() {
    @Override
    public void run() {
        for (int i = 0; i <5 ; i++)
        {
            SystemClock.sleep( ms: 1000);
            Log.d(TAG,  msg: "startLongWork: times = " + i + " id = "+ Thread.currentThread().getId());
        }
        Log.d(TAG,  msg: "startLongWork: finished  id = "+ Thread.currentThread().getId());

        //...
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                imageView.setImageResource(R.drawable.dog);
            }
        });
        //...
        imageView.post(new Runnable() {
            @Override
            public void run() {
                imageView.setImageResource(R.drawable.dog);
            }
            });
    }
```

הפעולה run תתבצע ב UIThread

הפעולה run תתבצע ב UIThread

הבנת התהליך לעומק Handler -

- Handler - מנגנון של אנדרואיד
- מחלקה קיימת שתפקידה לאפשר מעבר מידע בין threads
- בדוגמה שלנו - מ thread שרץ ברקע ל UI Thread
- הודעה או קוד - (Runnable)שיתבצע ב UI Thread

- אנדרואיד מאפשר מעטפת להעברת קוד Runnable על ידי:
  - runOnUiThread.
  - post.
  - postDelayed.
- במקרים רבים זה עונה על הצורך