תרגיל בית מס. 4 – Service

באחת ההרצאות הקודמות פגשנו במנשק בשם ExecutorService. מנשק זה מגדיר מספר מתודות המאפשרות יצירת מאגר "חוטים להשכיר": ניתן להפעיל באמצעותם מספר תוכניות במקביל, אבל אפשר גם לעשות בחוטים שימוש חוזר – משתוכנית אחת הסתיימה, החוט שהריץ אותה מתפנה להרצת תכנית אחרת. הסידור הזה מספק שני יתרונות:

1. העלות של הקמת חוט מתפזרת על פני מספר תוכניות: במקום שכל תכנית תתחיל חוט חדש, כמה תוכניות משתמשות בחוט יחיד בזו אחר זו. (הקמה של חוט היא פעולה יחסית כבדה.)

2. מספר החוטים המתבצעים בו זמנית נקבע על פי מספר החוטים במאגר, ולא על פי מספר התוכניות שמבקשות לרוץ. הדבר מגביל את העומס שעלול להיווצר גם בנוכחות בקשות מרובות לביצוע.

בתרגיל זה ננסה לממש מחלקה שמממשת את המנשק Service, הדומה ל-ExecutorService.

```
public interface Service
{
    public void execute( Runnable r );
    public void shutdown();
    public void shutdownNow();
    public boolean isShutdown();
}
```

מתודות אלו פועלות באופן הבא:

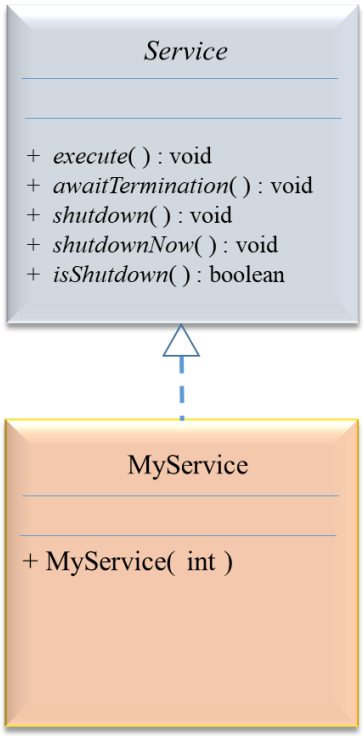
execute: זו המתודה החשובה כאן. היא מקבלת כפרמטר עצם מסוג Runnable המייצג תוכנית ודואגת שהיא תרוץ על אחד החוטים. אם אין חוט פנוי, העצם יחכה עד שאחד החוטים יתפנה.

shutdown: משנקראה מתודה זו, כל הבקשות לביצוע שהתקבלו עד כה יבוצעו, אבל לא יתקבלו בקשות חדשות: קריאה ל-()execute אחרי הקריאה ל-()shutdown תגרור זריקת חריגה מתאימה. (חישבו על מכולת בשעת סגירה: דלת הכניסה ננעלת, אבל הלקוחות שכבר נמצאים בחנות מורשים לסיים את קניותיהם, גם אלה שעדיין בתור.)

shutdownNow: מתודה זו סוגרת את השרות באופן מיידי: כל החוטים המבצעים מטלות יופסקו (באמצעות קריאה ל-()interrupt), ובקשות המחכות בתור לא יטופלו. (חישבו על מכולת בשעת הפסקת חשמל: כל השירותים מופסקים וכולם מתבקשים לעזוב את המקום ללא המצרכים שכבר אספו.)

isShutdown: מתודה המחזירה את מצב השרות: האם המערכת בפעולה (מחזירה **false**) או שהיא במצב shutdown (מחזירה **true**), מצב בו אין טעם לקרוא ל-()execute יותר (דלת החנות סגורה).

המחלקה שנכתוב, MyService, תממש את המנשק Service המתואר לעיל. למחלקה יש בנאי המקבל כפרמטר את מספר החוטים שיש להקים. המופע ישתמש בדיוק במספר הזה של חוטים. (המימושים הסטנדרטיים של מערכות דומות ב-JDK מאפשרים הגדלת מספר החוטים אם יש בקשות רבות, וגם הקטנת מספרם, אם רבים מהם בטלים מעבודה, אבל אנחנו לא נסתבך עם זה...)



המנשק Service, כפי שהוא מוגדר באתר התרגיל, מכיל מתודה אחת נוספת:

```
public void awaitTermination();
```

שפעולתה היא כדלקמן:

awaitTermination: חוט הקורא למתודה זו נבלם (blocked) עד שהשרות מסיים לבצע את כל הבקשות שנתנו לו עד אותו הרגע. מתודה זו עלולה לזרוק InterruptedException כמו כל מתודה בולמת אחרת (כגון sleep() או join()).

מימוש מתודה זו אינו מחייב: אפשר לממש אותה באופן ריק, כך שהפעלתה תסתכם בזריקת חריגה מסוג UnimplementedException. ברם, בונוס מצפה למי שיטרח לנסות לממש את המתודה כך שתפעל לפי המתואר לעיל.

להלן דוגמה לתכנית שעושה שימוש ב-MyService:

```
public static void main( String[] args ) throws InterruptedException
{
   Service s = new MyService( 5 );     // create a pool of 5 threads

   for( int j = 0; j < 30; j++ )       // generate 30 Runnables
   {
      s.execute( new Runnable()
      {
         public void run()
         {
            long id = Thread.currentThread().getId();
            System.out.println( "Thread: " + id + "   task: " + this );

            for( int i = 0; i < 100000000; i++ )  // take some time
               double d = Math.sin( (double) i );

            System.out.println( "Thread: " + id + "   end " );
         }
      } );
   }

   s.awaitTermination();                     // only if implemented
   System.out.println( "job done" );

   s.shutdown();
   System.out.println( "isShutdown() = " + s.isShutdown() );

   s.shutdownNow();
   System.out.println( "Terminated" );

}
```

פלט אפשרי מתכנית זו:

```
Thread: 1   task: MyService$1@1014cf9f
Thread: 4   task: MyService$1@28a66d58
Thread: 5   task: MyService$1@1127ec3d
Thread: 2   task: MyService$1@6fda4c80
Thread: 3   task: MyService$1@22e13983
Thread: 5   end task
Thread: 5   task: MyService$1@1fe00069
Thread: 1   end task
Thread: 1   task: MyService$1@114f7def
Thread: 4   end task
Thread: 4   task: MyService$1@2661963c
Thread: 1   end task
Thread: 1   task: MyService$1@3fc1c699
Thread: 3   end task
Thread: 3   task: MyService$1@22c5c3af
Thread: 2   end task
Thread: 2   task: MyService$1@611dbf0b
Thread: 5   end task
Thread: 5   task: MyService$1@5edd0a52
Thread: 1   end task
Thread: 1   task: MyService$1@55f77887
Thread: 2   end task
Thread: 2   task: MyService$1@652227f0
Thread: 4   end task
Thread: 4   task: MyService$1@225e0628
Thread: 3   end task
Thread: 3   task: MyService$1@281952f3
Thread: 4   end task
Thread: 4   task: MyService$1@675c9c07
Thread: 2   end task
Thread: 2   task: MyService$1@2020b530
Thread: 1   end task
Thread: 1   task: MyService$1@6001ded5
Thread: 5   end task
Thread: 5   task: MyService$1@703543b0
Thread: 4   end task
Thread: 4   task: MyService$1@7c8be119
Thread: 3   end task
Thread: 3   task: MyService$1@6d1bc994
Thread: 4   end task
Thread: 4   task: MyService$1@56f158c2
Thread: 5   end task
Thread: 5   task: MyService$1@6fc9f697
Thread: 1   end task
Thread: 1   task: MyService$1@a24b773f
Thread: 5   end task
Thread: 5   task: MyService$1@8cbe21cc
Thread: 3   end task
Thread: 3   task: MyService$1@461dd895
Thread: 4   end task
Thread: 4   task: MyService$1@4c2c7b72
Thread: 2   end task
Thread: 4   end task
Thread: 5   end task
Thread: 1   end task
Thread: 3   end task
job done
isShutdown() = true
Terminated
```

ההגשה בזוגות עד ה-10 ביוני. זו עבודה לא גדולה: בפתרון שלי יש לא יותר מ-100 שורות שאינן ריקות, הערות או הוראות import, כולל מימוש מלא של awaitTermination(). תפקידה הוא רק לחזק את בטחונכם בהבנה שיש לכם בנושא חוטים והרצתם 😊

בהצלחה!