ריבוי נימים (multithreading) בשפת Java

חלק שני

1. חקירת המחלקה Thread

בשיעור זה נבצע חקירה נוספת של המחלקה Thread .

ההגדרות שניתן לפעולות בשיעור זה תהינה כלליות ופשוטות וחלקן תתעדכנה במהלך השיעורים הבאים כאשר ניכנס לנושא הסנכרון.

1.1. המתודה sleep

1.1.1. הצהרת המתודה, שימוש ראשון

התיאור הרשמי של המתודה:

static void	**sleep**(long millis) Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

הפונקציה גורמת לנים (currentThread) ממנו מתבצעת הקריאה למתודה זו לוותר על המעבד למשך millis שניות. כלומר, במשך זמן זה התכנית לא מבקשת את המעבד, אינה מתחרה עם התכניות האחרות.

דוגמא לשימוש:

```
try {
      Thread.sleep(1000);
}
catch (InterruptedException e) {
}
```

הנים אשר יבצע פקודה זו ילך לשון למשך שניה אחת.

מתי נזרקת InterruptedException אותה אנחנו תופסים בדוגמא? במהלך ביצוע רגיל חריגה זו לא תיזרק. במקרה שנים אחר רוצה להעיר את הנים שיושן, אותו נים יפעיל פקודה אשר "תעיר" את הנים הישן הנים המתעורר יתפוס את החריגה וימשיך משם. (נראה דוגמא לכך בהמשך).

1.1.2. דוגמא שניה

התוכנית הבאה מריצה 2 נימים אשר כל אחד מהם הולך לישון זמן אקראי במשך 10 פעמים.

```
package example1;

/**
 * @author Guy Oren
 */
class SleepingThread extends Thread {
	public SleepingThread(String name) {
		setName(name);
	}
	public void run(){
		for(int i = 0;i<10;i++){
			int timeToSleep = (int)(1000*Math.random());
			System.out.println("Thread" + getName() +
			" is sleep for " + timeToSleep);
			try{
				Thread.sleep(timeToSleep);
			}
			catch(InterruptedException e1){
			}
		}
	}
}
```

הרצה של שני נימים:

```
package example1;
/**
 * @author Guy Oren
 */
public class SleepTest{
	public static void main (String[] args){
		Thread[] trds = new Thread[2];
		for(int i = 0;i<trds.length;i++){
			trds[i] = new SleepingThread(i+"");
			trds[i].start();
		}
	}
}
```

דוגמא לתוצאות ריצה אפשרית:

```
Thread0 is sleep for 972
Thread1 is sleep for 351
Thread1 is sleep for 165
Thread1 is sleep for 87
Thread1 is sleep for 565
Thread0 is sleep for 893
Thread1 is sleep for 115
Thread1 is sleep for 299
Thread1 is sleep for 778
Thread0 is sleep for 866
Thread1 is sleep for 316
Thread1 is sleep for 347
Thread0 is sleep for 177
Thread0 is sleep for 193
Thread1 is sleep for 560
Thread0 is sleep for 990
```

1.1.3. טעות נפוצה – להיזהר!

נשים לב לטעות נפוצה אצל מתחילים.

```
public class SleepTest{
      public static void main (String[] args)
      {
            Thread myThread = new Thread();
            …
            myThread.sleep(1000);
            …
            }
      }
}
```

מי הולך לישון פה? נים ה- main או הנים myThread? התשובה: ה-main .
מערכת ההפעלה בודקת מי הוא ה- currentThread ברגע שמופעלת הפונקציה sleep. היא לוקחת את נים זה ומוציאה אותו מהמעבד לזמן הרצוי. בזמן הריצה של ה- main הוא הנים הראשי ולכן הוא "יושכב לישון".
שימו לב שהטעות נובעת מהעובדה ש-sleep היא סטאטית!
הערה: currentThread היא מתודה אשר קיימת ב-API של Thread ומחזירה רפרנס לנים נוכחי.

שני דוגמאות נוספות:

```
class XThread extends Thread{
	public void run(){
		..
		Thread,sleep(1000);
		..
	}

	public void f(){
		Thread,sleep(1000);
	}
}
```

```
public class Test{
	public void test(){
		XThread x = new …;

		x.start(); //כאשר נים זה יבוצע הוא ילך לישון

		x.f();// הטרד מבצע את המתודה טסט הוא זה שילך לישון

		x.run();//.כמו קודם
	}
}
```

מה היה קורא אילו בתכנית הקודמת במקום Thread.sleep היינו כתובים sleep ?
אין שום בעיה, מאחר והמחלקה SleepingThread נורשת מ-Thread.

אגב, הקומפיילר של eclipse למשל מורה לך להעדיף את Thread.sleep כדי שיהיה ברור יותר ש-sleep היא סטאטית.
אם במקום לרשת מ-Thread היינו ממשים את ממשק Runnable אז כמובן שהייתה מתקבלת שגיאה.
Runnable הוא בסה"כ ממשק המכיל את המתודה run .

1.1.4. תרגיל

כתוב תכנית אשר מפעילה 2 נימים הרצים כל אחד בלולאה אינסופית. אחד הנימים יבקש כל 100 איטרציות של עצמו מהנים השני ללכת לישון. אם הנים השני ישן בזמן הבקשה אז הבקשה תבוטל.
יש להראות ע"י פלט מתאים. כל הודעה על שינוי מצב תודפס פעם אחת בלבד. (לשם כך השתמשתי בקוד במשתנים הבולאניים).

פתרון:

המחלקה של הנים המבקש מהנים האחר לישון.

```
package example2;
/**
 * @author Guy Oren
 */
public class SleepMaker extends Thread {
          //מקבל נים ממנו הוא מבקש לישון
         Sleeper theSleeper;
         public SleepMaker(Sleeper slp) {
           theSleeper = slp;
         }
         public void run(){
            //2 משתנים בוליאנים לצרכי סידור ההדפסה
           boolean msg1Printed = false;
           boolean msg2Printed = false;
           while(!theSleeper.isAlive());//מחכה הנים השני יתחיל
             for(int i = 0;;i++){
                   if(i%100 == 0){
                   //In this line the sleeper is asked to go sleep.
                   //if he is not sleeping he the method returns true.
             //and in the next time sleeper thread will go to sleep
                         if(theSleeper.goToSleep()){
                            if(!msg1Printed){
                               System.out.println("I put him to sleep");
                                     msg1Printed = true;
                                     msg2Printed = false;
                               }
                            }
                         else{
                         if(!msg2Printed){//he is allready sleeping
                               System.out.println
                               ("He is already sleeping");
                               msg2Printed = true;
                               msg1Printed = false;
                   }
                 }
             }
           }
         }
}
```

הקוד של המחלקה של הנים אשר הולך לישון בפקודה.

```
package example2;
class Sleeper extends Thread{
        // משתנה מצב המתאר אם הנים ישן
        private boolean sleeping;
        public void run(){
           //משתנה בוליאני לצרכי הדפסה
          boolean printed = false;
          for(;;){
            if(!printed){
              System.out.println("sleeper is working");
            printed = true;
            }
            if(sleeping){//בזמן הריצה הוא רואה האם הוא קיבל בקשה לשון
            try{
              Thread.sleep(100);
                             System.out.println("Waking up");
            }
            catch(InterruptedException e){
            }
            finally{
              sleeping = false;
              printed = false;
            }
            }//end if
          }//end for
        }//end run
        public boolean goToSleep(){
          if(sleeping)
            return false;
          return sleeping = true;
        }
}
```

הרצה:

```
package example2;

/**
 * @author Guy Oren
 */
public class SleepMakerTest {
      public static void test(){
            Sleeper    sleeper = new Sleeper();
            SleepMaker smaker  = new SleepMaker(sleeper);

            sleeper.start();
            smaker.start();
      }
      public static void main(String[] args) {
            test();
      }
}
```

נקודות לתשומת לב:

1.
כאשר thread רוצה לבקש משהו מנים אחר הוא צריך לדעת מיהו (רפרנס). הבקשות נעשות ע"י פונקציות או משתנים.
צריך להבין שכתיבה בריבוי נימים אינה רק להגיד שאובייקט הוא נים וגמרנו. זהו משהו שבהחלט מתערב לך בDesign ובאופן כתיבת הקוד.

2.
השיטה בה השורה בה הנים המבקש לשון מחכה למצב בו הנים השני יתחיל לרוץ לא יעילה נקראת busy waiting. נלמד דרכים יותר טובות לטפל במצבים כאלה.

3.
נראה זאת גם בהמשך הדרך הנכונה שנים יכול לבקש משהו מנים אחר היא ע"י בקשה ולא כפייה.
כלומר נים אחד מבקש והנים השני מבצע בעצמו את הבקשה.

1.2. פעולות על נים ישן

המתודה isAlive(): המתודה בודקת ומחזיקה האם הנים החל לרוץ, ועדיין לא הסתיים.

איך מפריעים ל thread בשנתו?

```
public void interrupt()
```

הפונקציה מופעלת ע"י thread אחד על thread אחר.

כאשר נים הפעיל את sleep (ובהמשך נראה עוד 2 פונקציות wait,join) הדרך היחידה להעיר אותו לפני ששינתו תמה היא שנים אחר יפעיל עליו את פעולת ה – interrupt הפעלת הinterrupt גורמת לפקודה sleep לזרוק חריגה.

הערה:
אם הנים לא נמצא באחת מ-3 הפונקציות שלעיל עדיין ניתן להשתמש בפונקציה interput בצמוד לעבודה אם הפונקציות:

```
public static boolean interrupted()
public static boolean isInterrupted()
```

(נראה דוגמאות בהמשך הקורס)

1.2.1. תרגיל

1. שנה את התרגיל הקודם כך שאם הטרד SleepMaker מגלה שהנים Sleeping ישן אז בהסתברות של 30% הוא מעיר את הנים השני.

הקודים לאחר השינוי: המחלקה SleepMaker:

```
public void wakeHimUp(){
  double prob = Math.random();
  if(prob<0.3)
    theSleeper.interrupt();
}
public void run(){
  boolean msg1Printed = false;
  boolean msg2Printed = false;
  while(!theSleeper.isAlive());
  for(int i = 0;;i++){
    if(i%100 == 0){
    if(theSleeper.goToSleep()){
      if(!msg1Printed){
       System.out.println("Shh I put him to slepp");
       msg1Printed = true;
       msg2Printed = false;
      }
    }
    else{
      if(!msg2Printed){
        System.out.println("He is allready sleeping");
        msg2Printed = true;
        msg1Printed = false;
        wakeHimUp();
      }
    }
    }
  }
}
```

בנים Sleeper בתוך ה- run:

```
public void run(){
  boolean printed = false;
  for(;;){
    if(!printed){
      System.out.println("sleeper is working");
    printed = true;
    }
    if(sleeping){
    try{
      Thread.sleep(100);
      System.out.println("Waking up");
    }
    catch(InterruptedException e){
      System.out.println("Waking up, before time");
    }
    finally{
      sleeping = false;
      printed = false;
    }
    }//end if
  }//end for
}//end run
```

1.3. כיצד לעצור נים?

במחלקה Thread יש מתודה הנקראית stop. מתודה זו אמורה לעצור את הנים.
אם תקפלו קוד המשתמש במתודה זו תקבלו הודעה שהמתודה היא deprecated - שזה בעברית המתודה לא מומלצת לשימוש. סאן לא לוקחת אחריות...

איך המתודה אמורה לעבוד?

```
Thread someThread  = …


…

someThead.stop(); //(*)

…
```

החל משורה (*) someThread אמור לא לרוץ. ובמקרים רבים בנימים פשוטים זה אכן עובד.
אבל בסאן ממליצים שאין לעצור כך אף נים אפילו נים מאוד פשוט. (בעצם הם מודים שהיה להם באג ב-Design ...)

מדוע מימוש זה לא טוב? עדיף שהאחריות לאופן ביצוע העצירה תיפול על הנים עצמו. כך הוא יוכל לנקות בצורה מסודרת משאבים שהוא מחזיק בהם. (קבצים, מנעולים (נראה בהמשך) וכו').

לכן נגדיר שיטה כללית לעצירת נים. בשיטה זו נגדיר אנו מתודה אשר תעצור את הנים בצורה מסודרת.

<u>הרעיון הכללי, הצעת הגשה מס' 1</u>

```
package example3;

/**
 * @author Guy Oren
 */
public class StoppableThead extends Thread{
	boolean stop;
	public void run(){
		while(!stop){
			// do something
		}
		//do cleaning jobs…

	}

	public void stopMe(){
		stop = true;
	}
}
```

מבקשים מהנים לעצור ע"י שנוי הדגל. ברגע שהנים יגיע לבדיקת תנאי הלולאה הוא יעצור.

בעיה ראשונה

נניח השנים נמצא בתוך sleep ארוך ? במקרה כזה ייקח זמן רב עד אשר הוא יגלה שהוא נתבקש לעצור.

פתרון באמצעות interrupt:

```
package example3;

/**
 * @author Guy Oren
 */
public class StoppableThead extends Thread{
	boolean stop;
	public void run(){
		while(!stop){
			try {
				// do something
				Thread.sleep(10000);
			}
			catch (InterruptedException e) {
			}
		}

	}

	public void stopMe(){
		stop = true;
		interrupt();
	}
}
```

שימו לב שגם בפתרון זה יש הסתברות מאוד קטנה שהבקשה תהיה לפני והנים יכנס לתנומה. (מתי?)
פתרון היה באמצעות סנכרון.

בעיה נוספת שלא נטפל בה במסגרת הקורס אך חשוב להכירה.
המקרה בו הנים נמצא ב-blocking io במקרה זה פקודת ה- interrupt לא תעזור כלל.
באופן כללי הרעיון הוא להוסיף למתודת ה- stop פקודה הסוגרת את ה- Stream .
(שימושי מאוד אם יש לכם נים שיושב על Socket , ואתם רוצים לעוצרו).

החל מ-JAVA 1.4 בשימוש ב-nio אפשר לבצע קלט אסינכרוני (non blocking io) ואז לעצור נים קל יותר.