– BuggyProgram
תוכנית לניתוח ובדיקת RaceFinder

מוגש על-ידי אוהד רז, ת.ז. 038249785

<u>מבוא</u>

השימוש בתכנות מקבילי נפוץ היום עד מאוד; תוכניות רבות עושות שימוש נרחב ב-Thread-ים, הרצים במקביל, במסגרת תוכנית אחת, בכדי לתת פתרון לבעיה/משימה נתונה.

על-אף היתרונות הגלומים בשיטת תכנות זו, היא טומנת בחובה "סכנות" אופייניות, שעל צוות הפיתוח וצוות הבדיקה להיות מודע להם. מצבים של "תחרות" על משאבים משותפים (Race-ים) ו-Deadlock-ים נפוצים ביותר, וקשה מאוד לגלותם.

אי-הדטרמיניסטיות בשיטת תכנות זו, והבעיה לדעת איזה Thread יישורת קודם, או ייגמר קודם, גורם לקושי רב במעקב אחרי זרימת התוכנית, והופך את תהליך הניתוח וה-Debug למסורבל, מסובך וקשה. כך קורה, שלעיתים באגים הנובעים מ-Race או Deadlock, מתגלים רק לאחר שהתוכנה כבר שוחררה לשוק.

רבים מכלי-הבדיקה האוטומטיים לתוכניות מקביליות מעלות את הסבירות למציאת באג על-ידי האוריסטיקות פשוטות ואחידות.

כלי-בדיקה אשר נוקט בהאוריסטיקות מורכבות יותר הוא RaceFinder. מרחב ה-Interleaving-ים האפשריים לבדיקה הוא רחב מאוד, אך רק חלקם יוצרים באגים בפועל. RaceFinder באה לגלות Interleaving-ים פוטנציאליים, אשר ייצרו Race-ים, ובכך יעלו את הסבירות לגלות באגים מקביליים.

<u>BuggyProgram</u>

תוכנית-דמה בשם BuggyProgram (נספח א') נכתבה בשפת Java, במיוחד לצורך בחינתה של RaceFinder.

BuggyProgram היא תוכנית פשוטה, אשר מחוללת ומקצה מספר ראנדומלי ייחודי למשתמש נתון. בנוסף, מתעדת BuggyProgram את רשימת כל המספרים שהוקצו.

ב-"עולם האמיתי", תוכנית שכזו יכולה להוות את הבסיס לאפליקציות להגרלת מספרי לוטו, הקצאת מספרי תעודת זהות, חילול סיסמאות, וכיו"ב.

התוכנית BuggyProgram היא תוכנית מקבילית, מרובת משתמשים, במובן זה שמספר משתמשים יכולים לבקש מספר בו-זמנית, כאשר כל בקשה מנוהלת על-ידי Thread נפרד.

דוגמא מה-"עולם האמיתי" יכולה להיות שרת או אתר אינטרנט.

שתי הבעיות הבאות, מסוג Weak-Reality, העלולות להיווצר, מצוינות להלן:

1. שני משתמשים (או יותר) מקבלים את אותו המספר.
2. המספר שנוצר עבור משתמש מסוים אינו המספר שתועד עבור אותו המשתמש.

<u>הניסוי</u>

RaceFinder הורצה על BuggyProgram, בכדי לבדוק אותה.

בכל אחת מהתוצאות שיובאו להלן, RaceFinder הורצה, כאשר הפרמטר ***Num of Times to Run*** מכוון ל-***100***.

מדד נקבע להסתברות לרעש מסוג ***Yield*** :

<u>High</u>: ***Yield Probability****: 90*

Loop Probability*: 90*

<u>Medium</u>: ***Yield Probability****: 60*

Loop Probability*: 60*

<u>Low</u>: ***Yield Probability****: 30*

Loop Probability*: 30*

כמו-כן, לרעש מסוג ***Barrier***, נקבע כי ה-***Barrier Base Time***, וה- ***Barrier Time Range*** יהיו ***1000***, כל אחד.

הניסוי נערך עם ***Whitenoise*** (כל המשתנים), עם משתנה יחיד (ראנדומלי, או המשתנה המשותף), ועם חמישה משתנים, שהם כמחצית מכלל המשתנים (כולם ראנדומלים, או המשתנה המשותף וארבעה משתנים ראנדומליים אחרים).

הניסויים הורצו על Pentium III, עם מערכת הפעלה Windows 2000, ו-JDK 1.4.

<u>תוצאות</u>

<u>הרצה 1-3</u> :

Concurrency Level*:* *little*

Size of Variable Subset*:* *Whitenoise*

Type of Noise Applied*:* *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	1%	0%	**ללא באגים**
2%	27%	3%	**1 באגים מתוך 3**
98%	72%	97%	**2 באגים מתוך 3**

<u>הרצה 4-6</u> :

Concurrency Level*:* *little*

Size of Variable Subset*:* *Single (random)*

Type of Noise Applied*:* *Yield*

Probability: *High*

after	*before*	*before and after*	
99%	100%	98%	**ללא באגים**
1%	0%	1%	**1 באגים מתוך 3**
0%	0%	1%	**2 באגים מתוך 3**

<u>הרצה 7-9</u> :

Concurrency Level*:* *little*

Size of Variable Subset*:* *Single (dedicated)*

Type of Noise Applied*:* *Yield*

Probability: *High*

after	*before*	*before and after*	
29%	1%	0%	**ללא באגים**
47%	18%	15%	**1 באגים מתוך 3**
24%	81%	85%	**2 באגים מתוך 3**

<u>הרצה 10-12</u> :

Concurrency Level*:* *little*

Size of Variable Subset*:* *50% (random)*

Type of Noise Applied*:* *Yield*

Probability: *High*

after	*before*	*before and after*	
86%	21%	19%	**ללא באגים**
10%	16%	21%	**1 באגים מתוך 3**
4%	63%	60%	**2 באגים מתוך 3**

הרצה 13-15:

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
30%	0%	0%	**ללא באגים**
47%	17%	18%	**1 באגים מתוך 3**
23%	83%	82%	**2 באגים מתוך 3**

הרצה 16-18:

Concurrency Level: *little*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
19%	30%	31%	**ללא באגים**
55%	54%	47%	**1 באגים מתוך 3**
26%	16%	22%	**2 באגים מתוך 3**

הרצה 19-21:

Concurrency Level: *little*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
100%	99%	98%	**ללא באגים**
0%	0%	1%	**1 באגים מתוך 3**
0%	1%	1%	**2 באגים מתוך 3**

הרצה 22-24:

Concurrency Level: *little*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
1%	0%	0%	**ללא באגים**
33%	0%	0%	**1 באגים מתוך 3**
66%	100%	100%	**2 באגים מתוך 3**

הרצה 25-27:

Concurrency Level: *little*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
74%	28%	14%	**ללא באגים**
11%	0%	4%	**1 באגים מתוך 3**
15%	72%	82%	**2 באגים מתוך 3**

הרצה 28-30

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
30%	0%	0%	**1 באגים מתוך 3**
70%	100%	100%	**2 באגים מתוך 3**

הרצה 31-33:

Concurrency Level: *little*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	9%	0%	**ללא באגים**
5%	39%	9%	**1 באגים מתוך 3**
95%	52%	91%	**2 באגים מתוך 3**

הרצה 34-36:

Concurrency Level: *little*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
98%	98%	98%	**ללא באגים**
0%	0%	0%	**1 באגים מתוך 3**
2%	2%	2%	**2 באגים מתוך 3**

הרצה 37-39:

Concurrency Level: *little*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
68%	5%	3%	**ללא באגים**
29%	46%	47%	**1 באגים מתוך 3**
3%	49%	50%	**2 באגים מתוך 3**

הרצה 40-42:

Concurrency Level: *little*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
86%	33%	28%	**ללא באגים**
13%	28%	31%	**1 באגים מתוך 3**
1%	39%	41%	**2 באגים מתוך 3**

הרצה 43-45:

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
58%	2%	2%	**ללא באגים**
34%	29%	30%	**1 באגים מתוך 3**
8%	69%	68%	**2 באגים מתוך 3**

הרצה 46-48:

Concurrency Level: *little*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
10%	0%	0%	**ללא באגים**
48%	18%	18%	**1 באגים מתוך 3**
42%	82%	82%	**2 באגים מתוך 3**

הרצה 49-51:

Concurrency Level: *little*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
96%	97%	100%	**ללא באגים**
0%	2%	0%	**1 באגים מתוך 3**
4%	1%	0%	**2 באגים מתוך 3**

הרצה 52-54:

Concurrency Level: *little*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
9%	0%	0%	**ללא באגים**
49%	8%	17%	**1 באגים מתוך 3**
42%	92%	83%	**2 באגים מתוך 3**

הרצה 55-57:

Concurrency Level: *little*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
68%	10%	25%	**ללא באגים**
17%	12%	11%	**1 באגים מתוך 3**
15%	78%	64%	**2 באגים מתוך 3**

הרצה 58-60:

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
14%	0%	0%	**ללא באגים**
54%	14%	7%	**1 באגים מתוך 3**
32%	86%	93%	**2 באגים מתוך 3**

הרצה 61-63:

Concurrency Level: *little*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
2%	46%	2%	**ללא באגים**
29%	41%	31%	**1 באגים מתוך 3**
69%	13%	67%	**2 באגים מתוך 3**

הרצה 64-66:

Concurrency Level: *little*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
100%	97%	98%	**ללא באגים**
0%	7%	0%	**1 באגים מתוך 3**
0%	2%	2%	**2 באגים מתוך 3**

הרצה 67-69:

Concurrency Level: *little*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
97%	69%	50%	**ללא באגים**
3%	25%	47%	**1 באגים מתוך 3**
0%	6%	3%	**2 באגים מתוך 3**

הרצה 70-72:

Concurrency Level: *little*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
95%	49%	51%	**ללא באגים**
4%	41%	37%	**1 באגים מתוך 3**
1%	10%	12%	**2 באגים מתוך 3**

הרצה 73-75:

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
88%	37%	34%	**ללא באגים**
12%	45%	45%	**1 באגים מתוך 3**
0%	18%	21%	**2 באגים מתוך 3**

הרצה 76-78:

Concurrency Level: *little*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
6%	7%	0%	**ללא באגים**
54%	36%	40%	**1 באגים מתוך 3**
40%	57%	60%	**2 באגים מתוך 3**

הרצה 79-81:

Concurrency Level: *little*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
99%	98%	98%	**ללא באגים**
0%	1%	1%	**1 באגים מתוך 3**
1%	1%	1%	**2 באגים מתוך 3**

הרצה 82-84:

Concurrency Level: *little*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
4%	6%	2%	**ללא באגים**
47%	37%	49%	**1 באגים מתוך 3**
49%	57%	49%	**2 באגים מתוך 3**

הרצה 85-87:

Concurrency Level: *little*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
84%	28%	26%	**ללא באגים**
11%	30%	26%	**1 באגים מתוך 3**
5%	42%	48%	**2 באגים מתוך 3**

הרצה 88-90:

Concurrency Level: *little*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
6%	1%	3%	**ללא באגים**
46%	27%	26%	**1 באגים מתוך 3**
48%	72%	71%	**2 באגים מתוך 3**

הרצה 91-93:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
1%	1%	1%	**30 באגים מתוך 33**
29%	6%	30%	**31 באגים מתוך 33**
70%	93%	69%	**32 באגים מתוך 33**

הרצה 94-96:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	0%	1%	**29 באגים מתוך 33**
5%	1%	3%	**30 באגים מתוך 33**
23%	11%	15%	**31 באגים מתוך 33**
72%	88%	81%	**32 באגים מתוך 33**

הרצה 97-99:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	1%	0%	**23 באגים מתוך 33**
4%	0%	2%	**30 באגים מתוך 33**
16%	21%	17%	**31 באגים מתוך 33**
80%	78%	81%	**32 באגים מתוך 33**

הרצה 100-102:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
8%	5%	3%	**30 באגים מתוך 33**
15%	11%	5%	**31 באגים מתוך 33**
77%	84%	92%	**32 באגים מתוך 33**

הרצה 103-105:

Concurrency Level: *average*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
4%	0%	1%	**30 באגים מתוך 33**
16%	7%	12%	**31 באגים מתוך 33**
80%	93%	87%	**32 באגים מתוך 33**

הרצה 106-108:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	1%	2%	**26 באגים מתוך 33**
1%	0%	0%	**28 באגים מתוך 33**
1%	30%	2%	**29 באגים מתוך 33**
7%	11%	8%	**30 באגים מתוך 33**
20%	25%	19%	**31 באגים מתוך 33**
71%	63%	71%	**32 באגים מתוך 33**

הרצה 109-111:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
2%	0%	3%	**30 באגים מתוך 33**
18%	15%	21%	**31 באגים מתוך 33**
80%	85%	76%	**32 באגים מתוך 33**

הרצה 112-114:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	0%	1%	**29 באגים מתוך 33**
2%	1%	1%	**30 באגים מתוך 33**
4%	20%	17%	**31 באגים מתוך 33**
94%	79%	81%	**32 באגים מתוך 33**

הרצה 115-117:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	0%	1%	**2 באגים מתוך 33**
0%	1%	1%	**4 באגים מתוך 33**
0%	1%	1%	**5 באגים מתוך 33**
0%	1%	3%	**25 באגים מתוך 33**
1%	3%	3%	**30 באגים מתוך 33**
14%	22%	18%	**31 באגים מתוך 33**
85%	72%	76%	**32 באגים מתוך 33**

הרצה 118-120

Concurrency Level: *average*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	0%	1%	**30 באגים מתוך 33**
6%	30%	22%	**31 באגים מתוך 33**
94%	70%	77%	**32 באגים מתוך 33**

הרצה 121-123:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	0%	1%	**30 באגים מתוך 33**
12%	7%	17%	**31 באגים מתוך 33**
88%	93%	82%	**32 באגים מתוך 33**

הרצה 124-126:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	1%	0%	**29 באגים מתוך 33**
2%	2%	1%	**30 באגים מתוך 33**
11%	17%	10%	**31 באגים מתוך 33**
87%	80%	89%	**32 באגים מתוך 33**

הרצה 127-129:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	1%	0%	**28 באגים מתוך 33**
3%	3%	3%	**30 באגים מתוך 33**
19%	24%	15%	**31 באגים מתוך 33**
78%	72%	82%	**32 באגים מתוך 33**

הרצה 130-132:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
1%	0%	0%	**29 באגים מתוך 33**
0%	1%	3%	**30 באגים מתוך 33**
11%	11%	11%	**31 באגים מתוך 33**
88%	88%	86%	**32 באגים מתוך 33**

הרצה 133-135:

Concurrency Level: *average*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	0%	1%	**29 באגים מתוך 33**
2%	2%	4%	**30 באגים מתוך 33**
13%	8%	11%	**31 באגים מתוך 33**
85%	90%	84%	**32 באגים מתוך 33**

הרצה 136-138:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
6%	5%	4%	**30 באגים מתוך 33**
8%	11%	8%	**31 באגים מתוך 33**
86%	84%	88%	**32 באגים מתוך 33**

הרצה 139-141:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
0%	0%	1%	**26 באגים מתוך 33**
6%	4%	2%	**30 באגים מתוך 33**
15%	15%	21%	**31 באגים מתוך 33**
79%	81%	76%	**32 באגים מתוך 33**

הרצה 142-144:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
1%	0%	0%	**29 באגים מתוך 33**
2%	1%	6%	**30 באגים מתוך 33**
18%	27%	15%	**31 באגים מתוך 33**
79%	72%	79%	**32 באגים מתוך 33**

הרצה 145-147:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
0%	1%	0%	**9 באגים מתוך 33**
0%	1%	0%	**10 באגים מתוך 33**
0%	1%	1%	**11 באגים מתוך 33**
0%	1%	0%	**14 באגים מתוך 33**
0%	1%	1%	**16 באגים מתוך 33**
1%	1%	0%	**17 באגים מתוך 33**
3%	0%	0%	**24 באגים מתוך 33**
3%	1%	0%	**26 באגים מתוך 33**
1%	0%	1%	**28 באגים מתוך 33**
2%	3%	0%	**29 באגים מתוך 33**
6%	1%	2%	**30 באגים מתוך 33**
21%	13%	14%	**31 באגים מתוך 33**
63%	77%	81%	**32 באגים מתוך 33**

הרצה 148-150:

Concurrency Level: *average*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
3%	0%	0%	**21 באגים מתוך 33**
3%	0%	0%	**22 באגים מתוך 33**
5%	0%	0%	**23 באגים מתוך 33**
6%	0%	0%	**24 באגים מתוך 33**
5%	0%	0%	**25 באגים מתוך 33**
2%	0%	0%	**26 באגים מתוך 33**
4%	0%	0%	**27 באגים מתוך 33**
8%	0%	0%	**28 באגים מתוך 33**
7%	0%	0%	**29 באגים מתוך 33**
15%	0%	0%	**30 באגים מתוך 33**
17%	12%	16%	**31 באגים מתוך 33**
25%	88%	84%	**32 באגים מתוך 33**

הרצה 151-153:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
5%	7%	10%	**31 באגים מתוך 33**
95%	93%	90%	**32 באגים מתוך 33**

הרצה 154-156:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
0%	1%	0%	**29 באגים מתוך 33**
2%	3%	2%	**30 באגים מתוך 33**
11%	18%	15%	**31 באגים מתוך 33**
87%	78%	83%	**32 באגים מתוך 33**

הרצה 157-159:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
1%	0%	0%	**29 באגים מתוך 33**
1%	0%	0%	**30 באגים מתוך 33**
21%	6%	9%	**31 באגים מתוך 33**
77%	94%	91%	**32 באגים מתוך 33**

הרצה 160-162:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
2%	0%	1%	**30 באגים מתוך 33**
19%	8%	6%	**31 באגים מתוך 33**
79%	92%	93%	**32 באגים מתוך 33**

הרצה 163-165:

Concurrency Level: *average*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
1%	0%	0%	**29 באגים מתוך 33**
13%	3%	4%	**31 באגים מתוך 33**
86%	97%	96%	**32 באגים מתוך 33**

הרצה 166-168:

Concurrency Level: *average*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
1%	1%	1%	**30 באגים מתוך 33**
4%	4%	7%	**31 באגים מתוך 33**
95%	95%	92%	**32 באגים מתוך 33**

הרצה 169-171:

Concurrency Level: *average*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
4%	7%	2%	**30 באגים מתוך 33**
18%	17%	22%	**31 באגים מתוך 33**
78%	76%	76%	**32 באגים מתוך 33**

הרצה 172-174:

Concurrency Level: *average*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
1%	0%	0%	**21 באגים מתוך 33**
1%	0%	0%	**22 באגים מתוך 33**
1%	0%	0%	**23 באגים מתוך 33**
1%	0%	0%	**24 באגים מתוך 33**
1%	0%	0%	**25 באגים מתוך 33**
6%	0%	0%	**26 באגים מתוך 33**
4%	0%	0%	**27באגים מתוך 33**
3%	0%	0%	**28 באגים מתוך 33**
11%	0%	0%	**29 באגים מתוך 33**
22%	1%	0%	**30 באגים מתוך 33**
24%	19%	10%	**31 באגים מתוך 33**
25%	80%	90%	**32 באגים מתוך 33**

הרצה 175-177:

Concurrency Level: *average*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
0%	2%	0%	**21 באגים מתוך 33**
0%	2%	1%	**22 באגים מתוך 33**
0%	1%	1%	**25 באגים מתוך 33**
0%	0%	1%	**27באגים מתוך 33**
1%	1%	1%	**28 באגים מתוך 33**
0%	4%	0%	**29 באגים מתוך 33**
5%	2%	1%	**30 באגים מתוך 33**
23%	12%	19%	**31 באגים מתוך 33**
71%	76%	76%	**32 באגים מתוך 33**

הרצה 178-180:

Concurrency Level: *average*

Size of Variable Subset: *50%*

(dedicated + 4 randoms)

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
9%	0%	0%	**28 באגים מתוך 33**
7%	0%	0%	**29 באגים מתוך 33**
16%	4%	1%	**30 באגים מתוך 33**
29%	14%	17%	**31 באגים מתוך 33**
39%	82%	82%	**32 באגים מתוך 33**

הרצה 181-183:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	0%	1%	**327 באגים מתוך 333**
1%	0%	0%	**329 באגים מתוך 333**
6%	7%	4%	**330 באגים מתוך 333**
26%	29%	27%	**331 באגים מתוך 333**
67%	64%	68%	**332 באגים מתוך 333**

הרצה 184-186:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
1%	1%	0%	**328 באגים מתוך 333**
1%	3%	1%	**329 באגים מתוך 333**
12%	16%	17%	**330 באגים מתוך 333**
38%	48%	36%	**331 באגים מתוך 333**
48%	32%	46%	**332 באגים מתוך 333**

הרצה 187-189:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	0%	1%	**328 באגים מתוך 333**
1%	0%	0%	**329 באגים מתוך 333**
3%	7%	1%	**330 באגים מתוך 333**
31%	27%	30%	**331 באגים מתוך 333**
65%	66%	69%	**332 באגים מתוך 333**

הרצה 190-192:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
1%	0%	0%	**327 באגים מתוך 333**
0%	0%	1%	**329 באגים מתוך 333**
8%	3%	7%	**330 באגים מתוך 333**
34%	22%	24%	**331 באגים מתוך 333**
57%	75%	68%	**332 באגים מתוך 333**

הרצה 193-195:

Concurrency Level: *lot*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *High*

after	*before*	*before and after*	
0%	2%	1%	**329 באגים מתוך 333**
2%	1%	3%	**330 באגים מתוך 333**
21%	31%	26%	**331 באגים מתוך 333**
77%	74%	70%	**332 באגים מתוך 333**

הרצה 196-198:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	0%	2%	**329 באגים מתוך 333**
4%	6%	8%	**330 באגים מתוך 333**
21%	21%	25%	**331 באגים מתוך 333**
75%	73%	65%	**332 באגים מתוך 333**

הרצה 199-201:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*Before and after*	
3%	2%	1%	**329 באגים מתוך 333**
11%	8%	16%	**330 באגים מתוך 333**
41%	55%	40%	**331 באגים מתוך 333**
45%	35%	43%	**332 באגים מתוך 333**

הרצה 202-204:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
1%	0%	2%	**329 באגים מתוך 333**
5%	11%	7%	**330 באגים מתוך 333**
34%	34%	38%	**331 באגים מתוך 333**
60%	55%	53%	**332 באגים מתוך 333**

הרצה 205-207:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
0%	0%	1%	**313 באגים מתוך 333**
0%	0%	1%	**319 באגים מתוך 333**
0%	0%	1%	**327 באגים מתוך 333**
1%	1%	6%	**329 באגים מתוך 333**
4%	5%	11%	**330 באגים מתוך 333**
30%	30%	31%	**331 באגים מתוך 333**
65%	64%	49%	**332 באגים מתוך 333**

הרצה 208-210

Concurrency Level: *lot*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *100%*

after	*before*	*before and after*	
1%	1%	0%	**329 באגים מתוך 333**
11%	7%	11%	**330 באגים מתוך 333**
41%	28%	27%	**331 באגים מתוך 333**
47%	64%	62%	**332 באגים מתוך 333**

הרצה 211-213:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
0%	0%	1%	**329 באגים מתוך 333**
4%	3%	7%	**330 באגים מתוך 333**
20%	32%	17%	**331 באגים מתוך 333**
76%	65%	75%	**332 באגים מתוך 333**

הרצה 214-216:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
1%	4%	3%	**329 באגים מתוך 333**
6%	13%	9%	**330 באגים מתוך 333**
51%	44%	53%	**331 באגים מתוך 333**
42%	39%	35%	**332 באגים מתוך 333**

הרצה 217-219:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
2%	0%	2%	**329 באגים מתוך 333**
4%	2%	3%	**330 באגים מתוך 333**
27%	22%	18%	**331 באגים מתוך 333**
67%	76%	77%	**332 באגים מתוך 333**

הרצה 220-222:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
3%	1%	2%	**329 באגים מתוך 333**
10%	2%	5%	**330 באגים מתוך 333**
33%	32%	28%	**331 באגים מתוך 333**
54%	65%	65%	**332 באגים מתוך 333**

הרצה 223-225:

Concurrency Level: *lot*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Medium*

after	*before*	*before and after*	
1%	1%	1%	**329 באגים מתוך 333**
4%	6%	2%	**330 באגים מתוך 333**
22%	22%	27%	**331 באגים מתוך 333**
73%	71%	70%	**332 באגים מתוך 333**

הרצה 226-228:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
1%	0%	2%	**329 באגים מתוך 333**
1%	7%	7%	**330 באגים מתוך 333**
31%	31%	18%	**331 באגים מתוך 333**
67%	62%	73%	**332 באגים מתוך 333**

הרצה 229-231:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
0%	0%	1%	**328 באגים מתוך 333**
5%	1%	2%	**329 באגים מתוך 333**
10%	14%	10%	**330 באגים מתוך 333**
43%	42%	49%	**331 באגים מתוך 333**
42%	43%	38%	**332 באגים מתוך 333**

הרצה 232-234:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
1%	0%	0%	**326 באגים מתוך 333**
3%	0%	0%	**327 באגים מתוך 333**
5%	0%	1%	**328 באגים מתוך 333**
15%	0%	2%	**329 באגים מתוך 333**
19%	10%	4%	**330 באגים מתוך 333**
34%	33%	37%	**331 באגים מתוך 333**
23%	57%	56%	**332 באגים מתוך 333**

הרצה 235-237:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
0%	1%	0%	**325 באגים מתוך 333**
0%	1%	2%	**326 באגים מתוך 333**
1%	2%	2%	**328 באגים מתוך 333**
2%	4%	3%	**329 באגים מתוך 333**
7%	9%	12%	**330 באגים מתוך 333**
37%	31%	30%	**331 באגים מתוך 333**
53%	52%	51%	**332 באגים מתוך 333**

הרצה 238-240:

Concurrency Level: *lot*

Size of Variable Subset: *50%*

(dedicated + 4 randoms)

Type of Noise Applied: *Barrier*

Probability: *60%*

after	*before*	*before and after*	
1%	0%	0%	**326 באגים מתוך 333**
2%	0%	0%	**327 באגים מתוך 333**
8%	0%	0%	**328 באגים מתוך 333**
10%	2%	2%	**329 באגים מתוך 333**
19%	6%	10%	**330 באגים מתוך 333**
26%	30%	27%	**331 באגים מתוך 333**
34%	62%	61%	**332 באגים מתוך 333**

הרצה 241-243:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
2%	1%	7%	**330 באגים מתוך 333**
17%	28%	22%	**331 באגים מתוך 333**
81%	71%	71%	**332 באגים מתוך 333**

הרצה 244-246:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
2%	2%	0%	**329 באגים מתוך 333**
6%	9%	11%	**330 באגים מתוך 333**
41%	41%	47%	**331 באגים מתוך 333**
51%	48%	42%	**332 באגים מתוך 333**

הרצה 247-249:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
6%	3%	3%	**330 באגים מתוך 333**
29%	34%	25%	**331 באגים מתוך 333**
65%	63%	72%	**332 באגים מתוך 333**

הרצה 250-252:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
1%	0%	0%	**329 באגים מתוך 333**
5%	2%	2%	**330 באגים מתוך 333**
32%	24%	27%	**331 באגים מתוך 333**
62%	74%	71%	**332 באגים מתוך 333**

הרצה 253-255:

Concurrency Level: *lot*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Yield*

Probability: *Low*

after	*before*	*before and after*	
5%	2%	7%	**330 באגים מתוך 333**
19%	24%	24%	**331 באגים מתוך 333**
76%	74%	69%	**332 באגים מתוך 333**

הרצה 256-258:

Concurrency Level: *lot*

Size of Variable Subset: *Whitenoise*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
2%	1%	0%	**329 באגים מתוך 333**
3%	7%	4%	**330 באגים מתוך 333**
28%	27%	16%	**331 באגים מתוך 333**
67%	65%	80%	**332 באגים מתוך 333**

הרצה 259-261:

Concurrency Level: *lot*

Size of Variable Subset: *Single (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
4%	2%	3%	**329 באגים מתוך 333**
10%	13%	7%	**330 באגים מתוך 333**
40%	41%	38%	**331 באגים מתוך 333**
46%	44%	52%	**332 באגים מתוך 333**

הרצה 262-264:

Concurrency Level: *lot*

Size of Variable Subset: *Single (dedicated)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
3%	0%	0%	**327 באגים מתוך 333**
5%	0%	0%	**328 באגים מתוך 333**
10%	1%	0%	**329 באגים מתוך 333**
29%	11%	5%	**330 באגים מתוך 333**
35%	31%	33%	**331 באגים מתוך 333**
20%	57%	62%	**332 באגים מתוך 333**

הרצה 265-267:

Concurrency Level: *lot*

Size of Variable Subset: *50% (random)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
0%	0%	1%	**325 באגים מתוך 333**
0%	1%	0%	**326 באגים מתוך 333**
1%	0%	1%	**327 באגים מתוך 333**
1%	2%	0%	**328 באגים מתוך 333**
5%	3%	3%	**329 באגים מתוך 333**
13%	17%	7%	**330 באגים מתוך 333**
25%	20%	25%	**331 באגים מתוך 333**
55%	57%	63%	**332 באגים מתוך 333**

הרצה 268-270:

Concurrency Level: *lot*

Size of Variable Subset: *50% (dedicated + 4 randoms)*

Type of Noise Applied: *Barrier*

Probability: *30%*

after	*before*	*before and after*	
4%	0%	0%	**327 באגים מתוך 333**
7%	0%	1%	**328 באגים מתוך 333**
17%	1%	0%	**329 באגים מתוך 333**
28%	4%	9%	**330 באגים מתוך 333**
32%	36%	29%	**331 באגים מתוך 333**
12%	59%	61%	**332 באגים מתוך 333**

<u>ניתוח ומסקנות</u>

לכאורה, הכלי RaceFinder הוא כלי יעיל, ולראיה – מספר הבאגים שמצא.

על-אף הישגים נאים אלו, בולט בדו"ח זה היעדרותו של כלי בדיקה נוסף, לצורכי השוואה. מכיוון שכלי בדיקה נוסף אינו כלול בדו"ח זה, ננסה להבין ולהעריך את יעילותו של RaceFinder, בפעולתו על BuggyProgram, באמות מידה תיאורטיות.

התוכנית BuggyProgram התבססה על עקרון ה-Scheduling של Java. Java נוקטת בעיקרון Scheduling פשוט אשר נקרא – fixed priority scheduling, ואשר שואף למינימיזציה של Turnaround Time, תחת שמירה על סדר עדיפויות (priorities). כלומר, ה-Scheduling של ה-CPU הוא preemptive לגמרי – אם מגיע Thread שצריך לרוץ, ולו יש priority גבוהה מזו של ה-Thread שרץ כרגע, ריצתו של ה-Thread הרץ תופסק מיד, וה-Thread עם ה-priority העדיף יחל מידית בריצתו.

Java לא תפסיק את ריצתו של Thread שרץ עבור Thread אחר, אשר לו priority זהה לזו של ה-Thread הרץ (או הנמוכה ממנו). במילים אחרות – Java לא מבצעת Time-Slice. כאשר לכל ה-Thread-ים אשר מוכנים לריצה (runnable) יש את אותה ה-priority, יבחר ה-Scheduler את ה-Thread הבא לריצה במנגנון פשוט – Round Robin, אשר לכשעצמו, אינו Preemptive.

מתוך הכרה במדיניות ה-Scheduling של Java, התוכנית BuggyProgram נכתבה ללא שימוש ב-"פרימיטיבים" של סינכרון, כגון sleep, wait, yield וכו'.

הכלי RaceFinder "שותל" פרימיטיביים אלו ב-ByteCode של התוכנית אותה הוא בודק, ומכאן יעילותו.

אמנם ידע מספיק בנושא חסר לי, אך נוטה אני להאמין שגם כלי בדיקה אחרים פועלים בשיטה זו.

על-פי מפתחיה, יתרונה הגדול של RaceFinder על פני תוכניות בדיקה אחרות הוא היכולת לשלוט היכן "לשתול" והיכן לא "לשתול" את אותם הפרימיטיביים, וזאת בניגוד לכלי בדיקה אחרים, אשר "שותלים" פרימיטיבים אלו בכל מקום בתוכנית. המצגת של RaceFinder מציגה את העוצמה הטמונה ביכולת זו.

בהיעדר הגדרה מפורשת, ומחוסר ידיעה לקראת מה עתידה התוכנית לעבור, על איזה כלי תיבדק וכו', כללה BuggyProgram רק משתנה משותף אחד. לפיכך – היא תוכנית מנוונת, שאין בה בכדי להפעיל, להציג ולמצות את יכולותיה האמיתיות של RaceFinder.

זוהי מסקנה חשובה מאוד, ועולה הרושם כאילו העבודה הייתה לחינם. יש בהחלט לערוך ניסויים על תוכניות מורכבות מעט יותר, ולבטח, אשר בהם יותר ממשתנה משותף יחיד.

כל שנותר כעת הוא לנסות ולהעריך מהם הפרמטרים האופטימליים להרצת RaceFinder. נעלה את השאלות הבאות, ננסה לענות על כל אחת מהן.

- ***Before, After או Before and After?***

בבואנו לבדוק את נקודות ה-Peak, כלומר, את נקודות השיא, מתקבלת התמונה הבאה:

Before - בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 6-4). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 22-24).

After - בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 19-21). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-98% מהמקרים (הרצה 1-3).

Before and After - בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100%

מהמקרים (הרצה 49-51). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 22-24).
כעת נבדוק את ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

Before - מקסימלית – 63.5%; מינימאלית (ללא באגים כלל) – 10.8%.

After - מקסימלית – 53.72%; מינימאלית (ללא באגים כלל) – 16.96%.

Before and After – מקסימלית – 65.18%; מינימאלית (ללא באגים כלל) – 9.78%.

על-פי תוצאות אלו, ניתן לומר כי **After** אינו פרמטר מתאים, היות והוא הניב את התוצאות ה-"רעות" ביותר מבין שלושת הפרמטרים האפשריים. – הוא נפל בתוצאותיו במציאת מירב הבאגים (98% לעומת 100% שהשיגו שני הפרמטרים הנוספים). זאת ועוד, הרי שהוא גם נצפה כפרמטר ה-"קופצני" והבלתי יציב ביותר.

שני הפרמטרים הנוספים, **Before** ו-**Before and After**, השיגו תוצאות טובות יותר, כאשר ידו של הפרמטר **Before and After** על העליונה – הוא התגלה כפרמטר יציב, אשר מגלה את מירב הבאגים באחוזים נאים.
על-פי תוצאות אלו, הייתי קובע כי הפרמטר **Before and After** הינו הפרמטר ה-"טוב" והעדיף ביותר – הוא מגלה את כמות הבאגים המקסימלית באחוזים יפים, והוא יציב.

- ***Lot, Average או Little?***

שוב, מחוסר הכרה מה משמעותם של פרמטרים אלה, נקבעו פרמטרים "לא טובים" ל-Lot (333), ל-Average (33), ול-Little (3).
ובכל זאת, אילו התוצאות אשר נצפו:

בבואנו לבדוק את נקודות ה-Peak, מתקבלת התמונה הבאה:

Lot - בנקודה הנמוכה ביותר, נמצאו לפחות כ-97% באגים אפשריים ב-1% מהמקרים (הרצה 115-117). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-81% מהמקרים (הרצה 241-243).

Average – בנקודה הנמוכה ביותר, נמצאו לפחות כ-5% באגים אפשריים ב-1% מהמקרים (הרצה 115-117). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-97% מהמקרים (הרצה 165-163).

Little – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 6-4). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית, ב-100% מהמקרים (הרצה 28-30).
נבדוק את ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

Lot – מקסימלית – 59%; מינימאלית (5% מהבאגים) – 0.0002%.

Average – מקסימלית – 80%; מינימאלית (5% מהבאגים) – 0.0001%.

Little – מקסימלית – 42%; מינימאלית (ללא באגים כלל) – 38%.

על-פי תוצאות אלו, ניתן לומר כי **Little** אינו פרמטר מתאים, היות ובמספר הרצות, לא נתגלו כל באגים בתוכנית.

זאת ועוד, הרי ש-**Little** נמצא כפרמטר "קופצני" ביותר ואשר תלוי וקשור לערכיהם של פרמטרים אחרים – לעיתים הוא מצא את 100% הבאגים בכל המקרים, ולעיתים לא מצא כל באגים ב-100% מהמקרים. זאת בניגוד, למשל, ל-**Average**, אשר נצפה כפרמטר יציב הרבה יותר.

מתוצאות אלו, הייתי קובע כי **Average** הינו הפרמטר ה-"טוב" ביותר – הוא מגלה את כמות הבאגים המקסימלית באחוזים יפים, והוא יציב.

- ***WhiteNoise, Single (random), Single (dedicated), 50% (random), או 50% (dedicated and random)?***

נבדוק בנקודות ה-Peak (נקודות השיא):

WhiteNoise – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-46% מהמקרים (הרצה 61-63). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-98% מהמקרים (הרצה 1-3).

Single (random) – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 4-6). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-89% מהמקרים (הרצה 124-126).

Single (dedicated) – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-97% מהמקרים (הרצה 67-69). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 22-24).

50% (random) – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-95% מהמקרים (הרצה 70-72). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-93% מהמקרים (הרצה 160-162).

50% (dedicated and random) – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-88% מהמקרים (הרצה 73-75). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 28-30).

ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

WhiteNoise – מקסימלית – 71.56%; מינימאלית (ללא באגים כלל) – 3%.

Single (random) – מקסימלית – 41.56%; מינימאלית (ללא באגים כלל) – 32.80%.

Single (dedicated) – מקסימלית – 63.22%; מינימאלית (ללא באגים כלל) – 6.37%.

50% (random) – מקסימלית – 59.41%; מינימאלית (ללא באגים כלל) – 15.28%.

50% (dedicated and random) – מקסימלית – 66.65%; מינימאלית (ללא באגים כלל) – 5.09%.

כפי שניתן היה להניח, הפרמטר **Single (random)** אינו פרמטר טוב, היות ובכשליש מההרצות הוא לא גילה כל באגים. תוצאה זו אינה מפתיעה, היות וייתכן בהחלט כי הפרימיטיבים של הסנכרון הופעלו על משתנים שאין להם כל קשר למשתנים משותפים. באותו האופן ניתן גם להסביר את התוצאות שהניב הפרמטר **50% (random)**.

מפתיע היה לגלות את התוצאות שהניבו הפרמטר **Single (dedicated)** והפרמטר **50% (dedicated and random)**, בהשוואה לפרמטר **WhiteNoise**. מפתחי RaceFinder טענו ש-"שתילת" פרימיטיבים של סנכרון "במקומות הנכונים" היא זו שתניב את התוצאות הטובות ביותר, וזאת על שום ש-"שתילה" של פרימיטיבים של סנכרון לפני ו/או אחרי כל משנה, באשר הוא, עלולה "לפתור" באג שאכן קיים בתוכנית. התוצאות שנצפו עולות *<u>בסתירה</u>* לטענות אלו. מהתוצאות עולה כי הפרמטר **WhiteNoise** הוא הפרמטר ה-"טוב" ביותר; אמנם, הפרמטרים **Single (dedicated)** ו-**50% (dedicated and random)** גילו את מירב הבאגים בנקודה הגבוהה ביותר ב-100% מהמקרים (לעומת 98% שהשיג **WhiteNoise**), אך **WhiteNoise** התגלה כפרמטר "יציב" הרבה יותר.

על-כן, הייתי קובע כי **WhiteNoise** הינו הפרמטר ה-"טוב" ביותר, שוב, *<u>בסתירה</u>* לטענתם של מפתחי RaceFinder. זהו פרמטר אשר מגלה את כמות הבאגים המקסימלית באחוזים יפים, והוא יציב.

- ***Yield או Barrier?***

נבדוק בנקודות ה-Peak (נקודות השיא):

Yield – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 4-6). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-98% מהמקרים (הרצה 1-3).

Barrier – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 19-21). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 22-24).

ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

Yield – מקסימלית – 61.36%; מינימאלית (ללא באגים כלל) – 14.72%.

Barrier – מקסימלית – 59.6%; מינימאלית (ללא באגים כלל) – 10.3%.

מהתוצאות עולה כי הפרמטר **Yield** הוא הפרמטר ה-"טוב" ביותר; אמנם, **Barrier** גילה את מירב הבאגים בנקודה הגבוהה ביותר ב-100% מהמקרים (לעומת 98% שהשיג **Yield**), אך **Yield** נצפה כפרמטר "יציב" הרבה יותר.

על-כן, הייתי קובע כי **Yield** הינו הפרמטר ה-"טוב" ביותר, אשר מגלה את כמות הבאגים המקסימלית באחוזים יפים, ואשר הוא יציב.

- ***בהנחה שבחרנו ב-Yield – מה עדיף – Low, Medium, או High?***

נבדוק בנקודות ה-Peak (נקודות השיא):

Low – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 64-66). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-97% מהמקרים (הרצה 163-165).

Medium – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-98% מהמקרים (הרצה 34-36). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-95% מהמקרים (הרצה 31-33).

High – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 4-6). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-98% מהמקרים (הרצה 1-3).

ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

Low – מקסימלית – 56.56%; מינימאלית (ללא באגים כלל) – 20.34%.

Medium – מקסימלית – 62.09%; מינימאלית (ללא באגים כלל) – 13.07%.

High – מקסימלית – 65.42%; מינימאלית (ללא באגים כלל) – 10.76%.

התוצאות מראות כי, כצפוי, הפרמטר **High** מניב את התוצאות ה-"טובות" ביותר – הן באחוזים נאים בגילוי מרבית הבאגים, והן ביציבות.

- ***ובהנחה שבחרנו ב-Barrier – מה עדיף – 30%, 60%, או 100%?***

נבדוק בנקודות ה-Peak (נקודות השיא):

30% – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-99% מהמקרים (הרצה 79-81). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-95% מהמקרים (הרצה 166-168).

60% – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 49-51). בנקודה הגבוהה ביותר, נמצאו

כמות הבאגים המקסימלית ב-93% מהמקרים (הרצה 58-60).

100% – בנקודה הנמוכה ביותר, לא נמצאו כל באגים, ב-100% מהמקרים (הרצה 19-21). בנקודה הגבוהה ביותר, נמצאו כמות הבאגים המקסימלית ב-100% מהמקרים (הרצה 22-24).

ממוצע אחוז מציאת הכמות המקסימלית וממוצע הכמות המינימלית של באגים:

30% – מקסימלית – 56.31%; מינימאלית (ללא באגים כלל) – 10.4%.

60% – מקסימלית – 60.27%; מינימאלית (ללא באגים כלל) – 9.54%.

100% – מקסימלית – 62.22%; מינימאלית (ללא באגים כלל) – 10.98%.

על-פי תוצאות אלו, ניתן לומר כי הפרמטר **30%** אינו פרמטר מתאים, היות והוא התגלה כפרמטר החלש ביותר ביציבותו וכן מציאת הבאגים המרביים שלו לא הייתה הטובה ביותר.

הפרמטר **60%** התגלה כפרמטר יציב אמנם, אך מציאת הבאגים המרביים שלו הייתה הנמוכה ביותר מבין כל שלושת הפרמטרים.

לפיכך, הייתי קובע כי **100%** הינו הפרמטר ה-"טוב" ביותר – הוא גילה בנקודות השיא את מירב הבאגים ב-100% מהמקרים, והוא פרמטר מספק ביציבותו.

היות והפרמטרים הנ"ל אינם תלויים, אין צורך לבצע בדיקה ואנליזה עם שילובים משילובים שונים של פרמטרים אלו ואחרים.

סיכום

מפתחי RaceFinder הציעו גישה חדשה למציאת באגים מקביליים – "שתילת" פרימיטיבים של סנכרון בנקודות אסטרטגיות בקוד. המפתחים הבטיחו כי כלי בדיקה שאכן ישתמש בחכמה בשתילת הפרימיטיבים של הסנכרון יחשוף באגים רבים, אשר לא היו מתגלים בעזרת הכלים אשר מצויים כיום בשוק (כאלה ששותלים פרימיטיבים על-פי שימוש בהאוריסטיקות פשוטות).

לצורך בדיקת RaceFinder ולצורך ניתוחה, נכתבה התוכנית BuggyProgram, ובה באג מקבילי.

RaceFinder הורצה על BuggyProgram, ותוצאות ההרצה הובאו לעיל.

במבט ראשון עולה הרושם כי אכן RaceFinder מקיימת את אשר הבטיחו מפתחיה, שהרי היא מצאה הרבה באגים.

עם זאת, בולטת בהיעדרותה השוואה למול תוצאות שהניב של כלי בדיקה אחר, כך שאין מדד השוואתי אמיתי לעדיפותה של RaceFinder על כלי בדיקה נוספים.

מהתוצאות עולה כי הפרמטרים היעילים ביותר להרצת RaceFinder הם:

- ***Before and After***,
- ***Average***,
- ***WhiteNoise***,
- ***Yield***, ו-
- ***High***.

תוצאות אלו מפריכות, למעשה, את טענת מפתחי RaceFinder. עולה כי אם נשתול פרימיטיבים של סנכרון לכל המשתנים, גם לפני וגם לאחר השימוש בהם, נקבל את התוצאות הטובות ביותר, בסתירה לאמור במצגת של RaceFinder.

עם זאת, אבקש לציין ולסייג דברי אלו: התוכנית BuggyProgram נכתבה ללא ידיעה והכרה לקראת הבדיקות שתעבור, ויכולות כלי הבדיקה שיבדוק אותה. התוכנית הינה תוכנית מנוונת ביותר, המציעה משתנה משותף אחד ויחיד. לפיכך, ל-RaceFinder לא ניתנה האפשרות להראות את יכולותיה (לפחות, על-פי טענת מפתחיה), וייתכן והתוצאות אשר הובאו לעיל עושות עוול לכלי בדיקה זה.

יש בהחלט לבצע ניסויים חדשים עם תוכניות מפותחות יותר, אשר לבטח כוללות יותר ממשתנה משותף יחיד.

נספח א'

קוד המקור של התוכנית BuggyProgram:

```
/*********************************************************************/
/*                                                                   */
/* BuggyProgram                                                      */
/*                                                                   */
/* Submitted by Raz Ohad, 038249785                                  */
/*                                                                   */
/*********************************************************************/

// ___________________________________________________________________
//
//   Module Name:
//     BuggyProgram
//
//   Extends:
//     None
//
//   Implements:
//     None
//
//   Inner Class Of:
//     None
//
//   Module Description:
//     This module includes handling the sample of the buggy program.
//     This "buggy program" generates and assigns a random number to a
//     specific user, and keeps track of all numbers generated.
//
//     In "real world", the application can be used for issuing lottery
//     numbers, identification numbers, passwords, and the like.
//
//     It is a multi-user, in the sense that two or more people can request
//     numbers at the same time, where each request is serviced by a thread.
//     A practical, real world example of this would be a web site.
//
//     Two possibilities could happen:
//     1) Two people get the same number.
//     2) The number generated for a particular user may not be the same
//        when the time comes to record the number.
//___________________________________________________________________


// ============================   Package   ============================
package buggyprogram;


// ======================   Java Imported Files   ======================
import java.io.*;


// ======================   Class definition   =======================
public class BuggyProgram{

  // ==========================   Constants   ==========================
```

```
  public static final int    LITTLE_CONCURRENCY=   3,
                             AVERAGE_CONCURRENCY=  33,
                             LOT_CONCURRENCY=      333,
                             MAX_DIGITS=           3,
                             INVALID=              -3;
  public static final String PROGRAM_NAME=         "BuggyProgram";


  // ============================   Variables   ============================
  protected static final StringBuffer buffer=       new StringBuffer();
  protected              long         randomNumber= INVALID;
  protected              long[]       history=      null,
                                      generated=    null;
  protected static       int          mismatch=     0;
            static       int          numOfUsers=   AVERAGE_CONCURRENCY;
            static       String       pattern=      "None;"


// ============================   Methods   ============================

// ____________________________________________________________________
//
//   Name: BuggyProgram (constructor)
//    Input:  None
//    Output: None
//    Description:  The constructor.
//                  Initializes the history array, creates users
//                  (according to NUM_OF_USERS), and calls activeUsers,
//                  to starts the users' running.
// ____________________________________________________________________

public BuggyProgram(){
   int    i=    0;
   User[] user= new User[numOfUsers];

   history=   new long[numOfUsers];
   generated= new long[numOfUsers];

   for (i= 0; i < numOfUsers; ++i){
      history[i]=   INVALID;
      generated[i]= INVALID;
   }

   for (i= 0; i < numOfUsers; ++i){
      user[i]= new User(i);
   }

   acivateUsers(user);

   for (i= 0; i < numOfUsers; ++i);
      if (generated[i] != history[i]){
         ++mismatch;
      }
   }

   if (mismatch != 0){
      pattern= "Weak-Reality";
   }
}
```

```
// ____________________________________________________________________
//
//  Name: main
//    Input:  String array - command line arguments
//    Output: None
//    Description:  Sets the buggy program, by calling the program
//                  constructor.
// ____________________________________________________________________

public static void main(String args[]){
   int        i=              0;
   String     outputFilename= null;
   FileWriter outputFile=     null;

   if (args.length > 2 || args.length < 1){
      System.out.println("Illegal arguments.");
      System.out.println("Arguments should be:");
      System.out.println("  1. The name of the output file, and");
      System.out.println("  2. Optional: Parameter of concurrency (little, " +
                         "average, lot).\n");
      System.exit(1);
   }

   outputFilename= args[0];

   if (args.length == 2){
      if (args[1].toLowerCase().equals("little")){
         numOfUsers= LITTLE_CONCURRENCY;
      } else {
         if (args[1].toLowerCase().equals("average")){
            numOfUsers = AVERAGE_CONCURRENCY;
         } else {
            if (args[1].toLowerCase().equals("lot")){
               numOfUsers= LOT_CONCURRENCY;
            } else {
               System.out.println("Unrecognized parameter of concurrency.\n" +
                                  "Should be: little, average, or lot.\n");
               System.exit(1);
            }
         }
      }
   } else {
      numOfUsers= AVERAGE_CONCURRENCY;
   }

   BuggyProgram buggyProgram= new BuggyProgram();

   buffer.append("<" + PROGRAM_NAME + ", " +
                       mismatch + " out of " + numOfUsers + ", " +
                       pattern +
                ("\>\n");
   try {
      outputFile= new FileWriter(outputFilename);
      outputFile.write(buffer.toString());
   } catch (IOException ex){
      System.out.println("File \"" + outputFilename + "\" is possibly " +
                         "corrupted.\n");
      System.exit(1);
   } catch (NullPointerException ex){
```

```
      System.out.println("File \"" + outputFilename + "\" cannot be " +
                         "accessed.\n");
      System.exit(1);
   } catch (Exception ex){
      System.out.println("File \"" + outputFilename + "\" cannot be " +
                         "accessed.\n");
      System.exit(1);
   } finally {
      if (outputFile != null){
         try {
            outputFile.close();
         } catch (IOException ex){
            System.out.println("Could not close the file \"" +
                               OutputFilename + "\".\n");
         }
      }
   }
}


// ____________________________________________________________________
//
//   Name: acivateUsers
//      Input:  User array - the users
//      Output: None
//      Description: Starts the users running.
// ____________________________________________________________________


public void acivateUsers(User[] user){
   for (int i= 0; i < numOfUsers; ++i){
      user[i].start();
   }

   for (int i= 0; i < numOfUsers; ++i){
      try {
         user[i].join();
      } catch (InterruptedException ex){
         System.out.println("interrupted!!!");
      }
   }
}




   // ____________________________________________________________________
   //
   //   Module Name:
   //      User
   //
   //   Extends:
   //      Thread
   //
   //   Implements:
   //      None
   //
   //   Inner Class Of:
   //      BuggyProgram
   //
```

```
//   Module Description:
//      This module includes handling a user in the sample of the buggy
//      program.
//
//      A user may ask to generate a number (unique) for him, and to
//      record it.
// ____________________________________________________________________


// ======================   Class definition   =====================
public class User extends Thread{

   // ========================   Constants   ========================


   // ========================   Variables   ========================
   int userNumber;


   // =========================   Methods   =========================

   // ____________________________________________________________________
   //
   //   Name: User (constructor)
   //      Input:  int – the user ID
   //      Output: None
   //      Description: The constructor.
   //                   Sets the userNumber member field to be the
   //                   user ID, passed as an argument.
   // ____________________________________________________________________

   public User(int userNumber){
      this.userNumber= userNumber;
   }


   // ____________________________________________________________________
   //
   //   Name: run
   //      Input:  None
   //      Output: None
   //      Description: This "starter" method generates a unique
   //                   random number for the user, and records it in
   //                   the history array.
   // ____________________________________________________________________

   public void run(){
      int i= 0;

      while (i != numOfUsers){
         generate();

         for (i= 0; i < numOfUsers; ++i){
            if (history[i] == randomNumber){
               break;
            }
         }
      }

      record();
   }
```

```
        // ____________________________________________________________________
        //
        //   Name: generate
        //      Input:  None
        //      Output: None
        //      Description: This synchronized method generates a random
        //                   number, with the specified maximum number of
        //                   digits (according to MAX_DIGITS).
        // ____________________________________________________________________

        protected synchronized void generate(){
           generated[userNumber]= randomNumber= (long) (Math.random() *
                                                Math.pow(10, MAX_DIGITS));
        }


        // ____________________________________________________________________
        //
        //   Name: record
        //      Input:  None
        //      Output: None
        //      Description: This synchronized method records the random
        //                   number, which has been generated, to the
        //                   history array.
        // ____________________________________________________________________

        protected synchronized void record(){
           history[userNumber]= randomNumber;
        }
    }
}
```