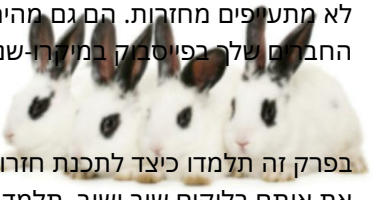
בלוקים חוזרים



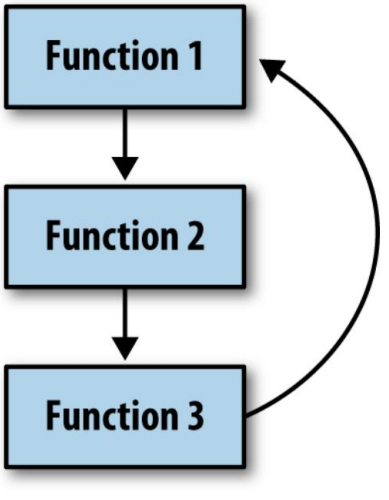
איור 20-1.

אם יש משהו שמחשבים טובים בו, זה לחזור על דברים -כמו ילדים קטנים, הם אף פעם לא מתעייפים מחזרות. הם גם מהירים מאוד ויכולים לעשות דברים כמו לעבד את כל רשימת החברים שלך בפייסבוק במיקרו-שנייה.

בפרק זה תלמדו כיצד לתכנת חזרות עם בלוקים חוזרים מיוחדים במקום להעתיק ולהדביק את אותם בלוקים שוב ושוב. תלמד כיצד לשלוח הודעת SMSלכל מספר טלפון ברשימה וכיצד להוסיף רשימה של מספרים. תלמד גם שחסימות חוזרות יכולות לפשט אפליקציה משמעותית.

שליטה בביצוע אפליקציה: הסתעפות ו בלולאה

בפרקים הקודמים, למדת שאתה להגדיר התנהגות של אפליקציה עם קבוצה של מטפלי אירועים -אירועים והפונקציות שצריכות להיות להורג בתגובה. למדת גם שהתגובה לאירוע היא לרוב לא רצף ליניארי של פונקציות ויכולה להכיל בלוקים שמתבצעים רק בתנאים מסוימים.



איור .2-20בלוקים חוזרים גורמים לתוכנית לעבור לולאה

בלוקים חוזרים הם הדרך השנייה שבה an האפליקציה מתנהגת בצורה לא ליניארית. ממש כאילו ועוד אם בלוקים מאפשרים לתוכנית להסתעף, בלוקים חוזרים מאפשרים לתוכנית להסתעף; כלומר, לבצע קבוצה של פונקציות ואז לקפוץ בחזרה למעלה בקוד ולעשות זאת שוב, כפי שמוצג באיור .1-20 כאשר אפליקציה מופעלת, מונה תוכניות הפועל מתחת למכסה המנוע של האפליקציה עוקב אחר הפעולה הבאה שתתבצע. עד כה, בדקתם אפליקציות שבהן מונה התוכניות מתחיל בראש מטפל באירועים ו(בתנאי)

מבצע פעולות מלמעלה למטה. עם בלוקים חוזרים, מונה התוכנית מתגבש בלולאות, ומבצע ברציפות את אותן פעולות.

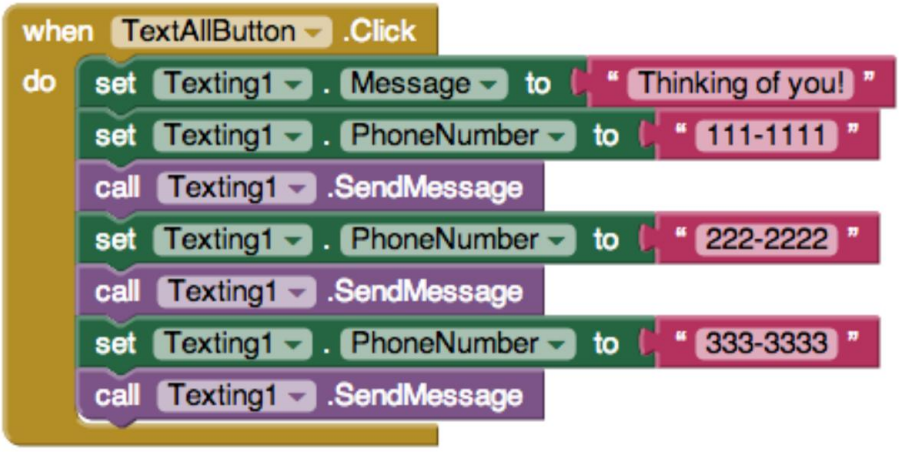
App Inventor מספק מספר בלוקים חוזרים, כולל עבור כל וזמן , בהם נתמקד בפרק זה. foreach משמש לציון פונקציות שיש לבצע בכל פריט ברשימה. אז אם יש לך רשימה של מספרי טלפון, אתה יכול לציין שיש לשלוח טקסט לכל מספר ברשימה

בלוק ה- while הוא כללי יותר מאשר עבור כל אחד מהם. עם זה, אתה יכול לתכנת בלוקים שחוזרים כל הזמן עד שמצב שרירותי כלשהו משתנה. אתה יכול להשתמש בלוקים בעוד כדי לחשב נוסחאות מתמטיות כגון הוספת n המספרים הראשונים או חישוב הפקטוריאלי של n. אתה יכול גם להשתמש בזמן כאשר אתה צריך לעבד שתי רשימות בו זמנית; עבור כל תהליכים רק רשימה בודדת בכל פעם.

איטרציה של פונקציות ברשימה עם עבור כל אחת

פרק 18 מדגים אפליקציה שמתקשרת באופן אקראי למספר טלפון אחד ברשימה. התקשרות אקראית לחבר אחד עשויה להצליח לפעמים, אבל אם יש לך חברים כמו שלי, הם לא תמיד עונים. אסטרטגיה שונה תהיה לשלוח "חושב עליך!" שלח הודעה לכל החברים שלך וראה מי מגיב ראשון (או הכי מקסים!).

אחת הדרכים ליישם אפליקציה כזו היא פשוט להעתיק את הבלוקים לשליחת טקסט לסינגל מספר ולאחר מכן הדבק אותם עבור כל חבר שאליו תרצה לשלוח הודעת טקסט, כפי שמוצג באיור 20-2.



איור 20-3. העתקה והדבקה של הבלוקים עבור כל מספר טלפון שיש לשלוח לו הודעת טקסט

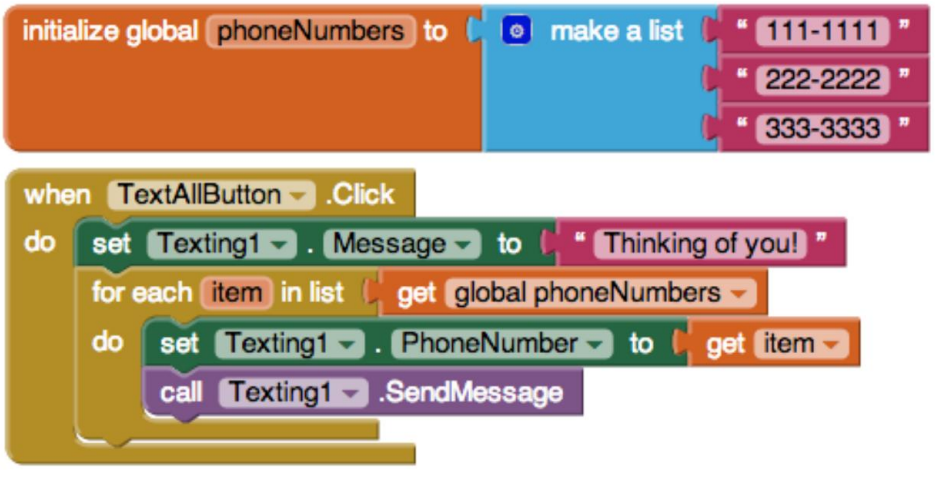
שיטת העתק-הדבק מסוג "כוח אכזרי" היא fine אם יש לך רק כמה בלוקים לחזור עליהם. אבל, אם אתה מתמודד עם כמויות גדולות של נתונים או נתונים שישתנו, אתה לא

רוצה לשנות את האפליקציה שלך בשיטת העתק-הדבק בכל פעם שאתה מוסיף או מסיר מספר טלפון מהרשימה שלך.

עבור כל בלוק מספק פתרון טוב יותר. אתה מגדיר משתנה phoneNumbers

עם כל המספרים ולאחר מכן עטפו aעבור כל בלוק סביב עותק בודד של הבלוקים שברצונכם לבצע. איור 20-3מציג את הפתרון עבור כל הודעה להודעות טקסט

קְבוּצָה.



איור .20-4שימוש ב- עבור כל בלוק כדי לבצע את אותם בלוקים עבור כל פריט ברשימה

אתה יכול לקרוא את הקוד הזה בתור, "עבור כל פריט (מספר טלפון) ברשימה מספרי טלפון, הגדר את מספר הטלפון של אובייקט הטקסט לפריט ושלח את הודעת הטקסט."

בחלק העליון של עבור כל בלוק, אתה מציין את הרשימה שתעובד. לבלוק יש גם משתנה מציין מיקום שמגיע עם עבור כל אחד. כברירת מחדל, מציין מיקום זה נקרא "פריט". אתה יכול להשאיר אותו כך או לשנות את שמו. משתנה זה מייצג את הפריט הנוכחי המעובד ברשימה.

אם רשימה כוללת שלושה פריטים, הבלוקים הפנימיים יבוצעו שלוש פעמים. אומרים שהבלוקים הפנימיים כפופים, או מקוננים בתוך, עבור כל בלוק. אנו אומרים שמונה התוכנית "מתגלגל" בחזרה כשהוא מגיע לבלוק התחתון בתוך

לכל אחד.

מבט מקרוב על לולאה

הבה נבחן את המכניקה של כל בלוקים בפירוט, מכיוון שהבנת לולאות היא בסיסית לתכנות. כאשר המשתמש מקיש על TextAllButtonועל

מופעל מטפל באירועים, הפעולה הראשונה שבוצעה היא הגדר Texting1.Messageלחסימה, מה שמגדיר את ההודעה ל"חושב עליך!" בלוק זה מבוצע פעם אחת בלבד.
לאחר מכן מתחיל עבור כל בלוק. לפני ביצוע הבלוקים המקוננים של aעבור כל אחד מהם, פריט משתנה מציין המיקום מוגדר למספר הראשון ברשימת מספרי הטלפון .(1111–111) זה קורה אוטומטית; עבור כל אחד משחררים אותך מהצורך להתקשר לפריט ידנית לפריט בחר רשימה. לאחר שהפריט הראשון נבחר לפריט המשתנה, הבלוקים בתוך עבור כל אחד מבוצעים בפעם הראשונה. ה
מאפיין SMS1.PhoneNumberמוגדר לערך של פריט (1111–111)וההודעה נשלחת.

לאחר הגעה לבלוק האחרון בתוך aעבור כל אחד (גוש ,(Texting.SendMessage האפליקציה "נכנסת" חזרה לראש ה- forכל אחד ומכניסה אוטומטית את הפריט הבא ברשימה (2222–222)לפריט המשתנה. לאחר מכן חוזרות על שתי הפעולות בתוך עבור כל אחת , ושולחות את ה"חושב עליך!" טקסט למספר .2222–222לאחר מכן, האפליקציה חוזרת שוב בלולאה ומגדירה את הפריט לפריט האחרון ברשימה .(3333–333)הפעולות חוזרות על עצמן בפעם השלישית, שליחת הטקסט השלישי.
מכיוון שהפריט הסופי ברשימה עבר עיבוד, ה- עבור כל לולאה נעצר ב-
הנקודה הזו. בשפת תכנות, אנו אומרים שהבקרה "קופצת" מחוץ ללולאה, מה שאומר שמונה התוכניות ממשיך להתמודד עם הבלוקים שמתחת לכל אחד. בדוגמה זו, אין בלוקים מתחתיו, כך שהמטפל באירוע מסתיים.

כתיבת קוד בר תחזוקה

למשתמש של האפליקציה, עבור כל פתרון שתואר זה עתה מתנהג בדיוק כמו שיטת "הכוח האכזרי" של העתקה ולאחר מכן הדבקת בלוקים להודעות טקסט. מנקודת מבטו של מתכנת, לעומת זאת, הפתרון לכל פתרון ניתן לתחזוקה וניתן להשתמש בו גם אם הנתונים (רשימת הטלפונים) מוזנים באופן דינמי.
תוכנה הניתנת לתחזוקה היא תוכנה שניתן לשנות בקלות מבלי להכניס באגים. עם הפתרון עבור כל אחד , אתה יכול לשנות את רשימת החברים שנשלחו להם טקסטים על ידי שינוי רק במשתנה הרשימה -אינך צריך לשנות את ההיגיון של התוכנית שלך (מטפל באירועים) כלל. הפוך זאת לשיטת ,brute-forceמחייבת אותך להוסיף בלוקים חדשים במטפל האירוע כאשר חבר חדש נוסף.
בכל פעם שאתה משנה את ההיגיון של תוכנית, אתה מסתכן בהחדרת באגים.
לא פחות חשוב, עבור כל פתרון יעבוד גם אם רשימת הטלפונים הייתה דינמית; כלומר, כזה שבו משתמש הקצה יכול להוסיף מספרים לרשימה. בניגוד לדוגמא שלנו, שיש לה שלושה מספרי טלפון מסוימים הרשומים בקוד, רוב האפליקציות עובדות עם נתונים דינמיים שמגיעים ממשתמש הקצה או ממקור אחר. אם עיצבת מחדש את האפליקציה הזו כך שמשתמש הקצה יוכל להזין את מספרי הטלפון, תצטרך להשתמש ב-a עבור כל פתרון, מכיוון שכאשר אתה כותב את התוכנית, אתה לא יודע אילו מספרים לשים בפתרון .brute-force-ה

שימוש עבור כל אחד כדי להציג רשימה

כאשר אתה רוצה להציג את הפריטים של רשימה בטלפון, אתה יכול לחבר את הרשימה למאפיין הטקסט של תווית, כפי שמוצג באיור .20-4

איור .20-5הדרך הפשוטה להציג רשימה היא לחבר אותה ישירות לתווית

כאשר אתה מחבר רשימה ישירות למאפיין טקסט של תווית, פריטי הרשימה מוצגים בתווית כשורה אחת של טקסט, מופרדים ברווחים ומוכללים בסוגריים: (333–3333 222–2222 111–1111)

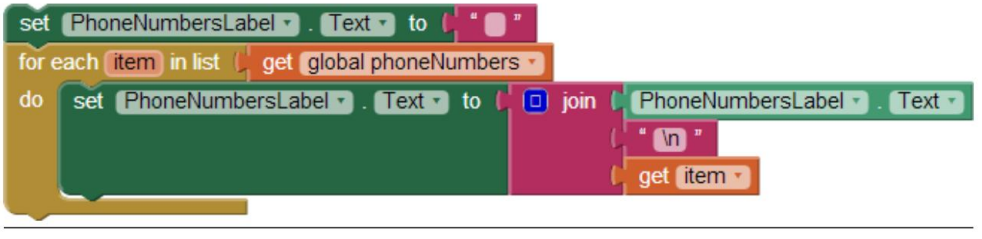
המספרים עשויים להשתרע על יותר משורה אחת או לא, תלוי כמה יש. המשתמש יכול לראות את הנתונים ואולי להבין שזו רשימה של מספרי טלפון, אבל זה לא מאוד אלגנטי. פריטי רשימה מוצגים בדרך כלל בשורות נפרדות או עם פסיקים המפרידים ביניהם.

כדי להציג רשימה כמו שצריך, אתה צריך בלוקים שהופכים כל פריט רשימה ליחיד ערך טקסט עם העיצוב הרצוי. אובייקטי טקסט מורכבים בדרך כלל מאותיות, ספרות וסימני פיסוק. עם זאת, טקסט יכול גם לאחסן תווי בקרה מיוחדים, שאינם ממפים לתו שאתה יכול לראות. כרטיסייה, למשל, מסומנת ב-\t. (למידע נוסף על תווי בקרה, עיין בתקן Unicode לייצוג טקסט בכתובת http://www.unicode.org/standard/standard.html.)

ברשימת מספרי הטלפון שלנו, אנו רוצים תו חדש, המסומן ב-\n.
כאשר \n מופיע בגוש טקסט, זה אומר "עבור לשורה הבאה לפני שתציג את הפריט הבא." לפיכך, אובייקט הטקסט "111–1111\n222–2222\n333–3333"יופיע כ:

```
111–1111
222–2222
333–3333
```

כדי לבנות אובייקט טקסט כזה, אנו משתמשים ב-a עבור כל בלוק ו"מעבדים" כל פריט על ידי הוספתו יחד עם תו חדש למאפיין , PhoneNumberLabel.Text כפי שמוצג באיור .20-5



איור A .20-6עבור כל בלוק המשמש להצגת רשימה עם פריטים בשורות נפרדות

בואו נתחקה אחר הבלוקים כדי לראות איך הם עובדים. כפי שנדון בפרק ,15מעקב מראה כיצד כל משתנה או תכונה משתנה עם ביצוע הבלוקים. עם aעבור כל אחד, אנו רואים את הערכים לאחר כל איטרציה; כלומר, בכל פעם שהתוכנית עוברת את עבור כל לולאה.

לפני כל אחד , ,PhoneNumbersLabelמאותחל לטקסט הריק. כאשר עבור כל אחד מתחיל, האפליקציה ממקמת אוטומטית את הפריט הראשון ברשימה (111-1111)בפריט המשתנה של מציין המיקום. הבלוקים של כל אחד לאחר מכן מצטרפים עם PhoneNumbersLabel.Text(הטקסט הריק), n\והפריט, ומגדירים את התוצאה ל- .PhoneNumbersLabel.Textלפיכך, לאחר האיטרציה הראשונה של ה- עבור כל אחד מהם, המשתנים הרלוונטיים מאחסנים את הערכים המוצגים בטבלה 20-1.

טבלה .20-1הערכים לאחר האיטרציה הראשונה

פריט	PhoneNumbersLabel.Text
111–1111	\n111–1111

מכיוון שהגענו לתחתית של כל אחד , לולאות בקרה מגובות, ו
הפריט הבא ברשימה (222–2222)מוכנס לפריט המשתנה. כאשר הבלוקים הפנימיים חוזרים על עצמם, הטקסט משרשר את הערך של (1111 –111\n) PhoneNumbersLabel.Textעם ,n\ולאחר מכן עם פריט, שהוא כעת .222–2222לאחר האיטרציה השנייה הזו,
המשתנים מאחסנים את הערכים המוצגים בטבלה .20-2

טבלה .20-2הערכים לאחר האיטרציה השנייה

פריט	PhoneNumbersLabel.Text
222–2222	\n111–1111\n222–2222

לאחר מכן, הפריט השלישי ברשימה ממוקם בפריט, והגוש הפנימי חוזר על עצמו א
פעם שלישית. הערך הסופי של המשתנים, לאחר איטרציה אחרונה זו, מוצג ב
טבלה .20-3

טבלה .20-3ערכי המשתנים לאחר האיטרציה הסופית

פריט	PhoneNumbersLabel.Text
333–3333	\n111–1111\n222–2222\n333–3333

לכן, לאחר כל איטרציה, התווית הופכת גדולה יותר ומכילה עוד מספר טלפון אחד (ועוד שורה חדשה אחת). בסוף הערך עבור כל אחד, PhoneNumbersLabel.Text מוגדר כך שהמספרים יופיעו כדלקמן: 111–1111

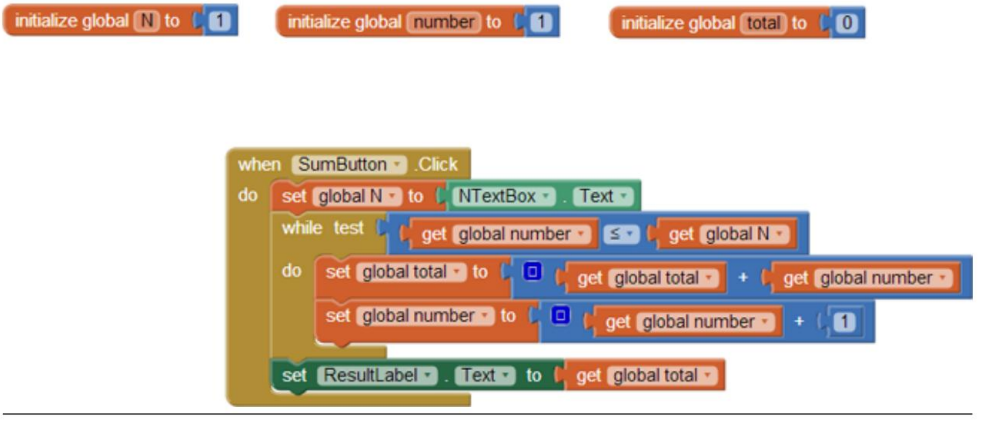
222–2222
333–3333

בלוק ה-while-do

בלוק ה- while-do הוא קצת יותר מסובך לשימוש מאשר עבור כל אחד. היתרון של בלוק ה- do while- טמון בכלליותו: עבור כל חזרה על רשימה, אבל בעוד יכול לחזור כל עוד כל תנאי שרירותי הוא נכון.
כפי שלמדת בפרק 18, תנאי בודק משהו ומחזיר ערך של נכון או לא נכון. בלוקים while-do כוללים מבחן מותנה, בדיוק כמו אם בלוקים. אם הבדיקה של זמן מה מוערכת כאמיתית, האפליקציה מבצעת את החסימות הפנימיות, ולאחר מכן חוזרת בלולאה ובודקת מחדש את הבדיקה. כל עוד הבדיקה מוערכת כנכונה, הבלוקים הפנימיים חוזרים על עצמם. כאשר הבדיקה מוערכת כ-false, האפליקציה יוצאת מהלולאה (כמו שראינו עם עבור כל בלוק) וממשיכה עם הבלוקים מתחת ל- while-do.

שימוש while-do בכדי לחשב נוסחה

הנה דוגמה לבלוק while-do שחוזר על פעולות. מה לדעתך עושים הבלוקים באיור 20-6? דרך אחת לגלות זאת היא לעקוב אחר כל בלוק (ראה פרק 15 למידע נוסף על מעקב), מעקב אחר הערך של כל משתנה תוך כדי תנועה.



איור 20-7. האם אתה יכול להבין מה הבלוקים האלה עושים?

הבלוקים בתוך לולאת while-doיחזרו על עצמם בעוד שמספר המשתנה קטן או שווה למשתנה N.
עבור אפליקציה זו, Nמוגדר למספר שמשתמש הקצה מקליד בתיבת טקסט .(NTextBox)נניח
שהמשתמש מקליד 3.המשתנים של האפליקציה ייראו כמו טבלה 20-4כאשר תגיע לראשונה לבלוק ה-
while-do .

טבלה .20-4ערכי משתנים כאשר תגיעו לראשונה while-doל-

	Nסך הכל	מספר
		3 1

הבלוק while-doבודק קודם את התנאי: האם מספר קטן או שווה ל-(□) ?N
בפעם הראשונה ששואלים שאלה זו, הבדיקה נכונה, ולכן הבלוקים המקוננים בתוך בלוק while-do
מבוצעים. סך מוגדר לעצמו (0)פלוס מספר ,(1)והמספר מוגדל. לאחר האיטרציה הראשונה של
הבלוקים בתוך ה- ,while-doהמשתנה
הערכים הם כמפורט בטבלה 20-5.

טבלה .20-5ערכי המשתנים לאחר האיטרציה הראשונה של הבלוקים בתוך בלוק while

	Nסך הכל	מספר
3	2	1

באיטרציה השנייה, הבדיקה "N□רפסמ"עדיין נכונה ,(3□2)כך שהבלוקים הפנימיים
מוצאים להורג שוב. סך מוגדר לעצמו (1)פלוס מספר .(2)מספר מוגדל.
כאשר איטרציה שנייה זו מסתיימת, המשתנים מכילים את הערכים המפורטים בטבלה 20-6.

טבלה .20-6ערכי המשתנים לאחר האיטרציה השנייה

	Nסך הכל	מספר
3	3	3

האפליקציה חוזרת שוב בלולאה ובודקת את המצב. שוב, זה נכון ,(3□3)אז הבלוקים מבוצעים בפעם
השלישית. כעת, הסכום מוגדר לעצמו (3)בתוספת מספר ,(3)כך שהוא הופך ל-6. מספר מוגדל ל-4, כפי
שמוצג בטבלה 20-7.

טבלה .7-20הערכים לאחר האיטרציה השלישית

	Nסך הכל	מספר
3	4	6

לאחר איטרציה שלישית זו, האפליקציה חוזרת שוב פעם אחת לראש הזמן
לַעֲשׂוֹת. כאשר הבדיקה "N רפסמ"פועלת הפעם, היא בודקת ,3 4אשר מוערכת לשווא.
לפיכך, הבלוקים המקוננים של while-doאינם מבוצעים שוב, ומטפל האירועים משלים.

אז מה עשו הבלוקים האלה? הם ביצעו את אחת הפעולות המתמטיות הבסיסיות ביותר: ספירת מספרים. לא משנה מה המספר שהמשתמש יקליד, האפליקציה תדווח על סכום המספרים ,N..1 כאשר Nהוא המספר שהוזן. בדוגמה זו, Nהוא ,3כך שהאפליקציה הגיעה עם סך של .1+2+3=6אם המשתמש היה מקליד ,4האפליקציה הייתה מחשבת .10

סיכום

מחשבים טובים בלחזור על אותה פונקציה שוב ושוב. חשבו על כל חשבונות הבנק שמעובדים לצבירת ריבית, כל הציונים המעובדים לחישוב ממוצעי הציונים של התלמידים, ועוד אינספור דוגמאות יומיומיות שעבורן מחשבים משתמשים בחזרה כדי לבצע משימה.

פרק זה חקר שניים מהבלוקים החוזרים של .App Inventorעבור כל בלוק
מחיל קבוצה של פונקציות על כל רכיב ברשימה. על ידי שימוש בו, אתה יכול לעצב קוד עיבוד שעובד על רשימה מופשטת במקום נתונים קונקרטיים. קוד כזה ניתן לתחזוקה יותר; ואם הנתונים שיש לעבד הם דינמיים, זה נדרש.
בהשוואה לכל אחד, while-doהוא כללי יותר: אתה יכול להשתמש בו כדי לעבד רשימה, אבל אתה יכול להשתמש בו גם לעיבוד סינכרוני של שתי רשימות או לחשב נוסחה. עם ,while-doהחסימות הפנימיות מבוצעות ברציפות כל עוד מצב מסוים נכון. לאחר ביצוע החסימות בתוך הזמן , לולאות בקרה מגובות ומצב הבדיקה נוסה שוב. רק כאשר הבדיקה מוערכת false-כמסתיים בלוק ה- . while-do