



MIT
APP INVENTOR

פעולות

מצגת מספר 6

מה למדנו בפרק הקודם?

- מהו ביצוע חוזר - לולאה
- מתי נשתמש בלולאות
- סוגי לולאות
- מבנה של לולאות
- שילוב רכיב שעון באפליקציה

נושאי השיעור

- אתגרים מורכבים
- פעולות
- זימון פעולה עם ובלי פרמטרים
- רכיב Notifier
- סנסור ACCELERATOR

אתגרים מורכבים - ניהול זמנים

נניח ואתם עורכים שיפוץ בביתכם.

יש המון חלקים בבית שצריכים לשפץ - חשמל, פרקט, דשא בגינה, שיפוץ המקלחת והשירותים, ארונות מטבח והחלפת דלתות.

ניהול כל העבודות הינו קשה, במיוחד שלכל עבודה צריכים להגיע אנשי עבודה המתמחים בסוג העבודה המסוים.

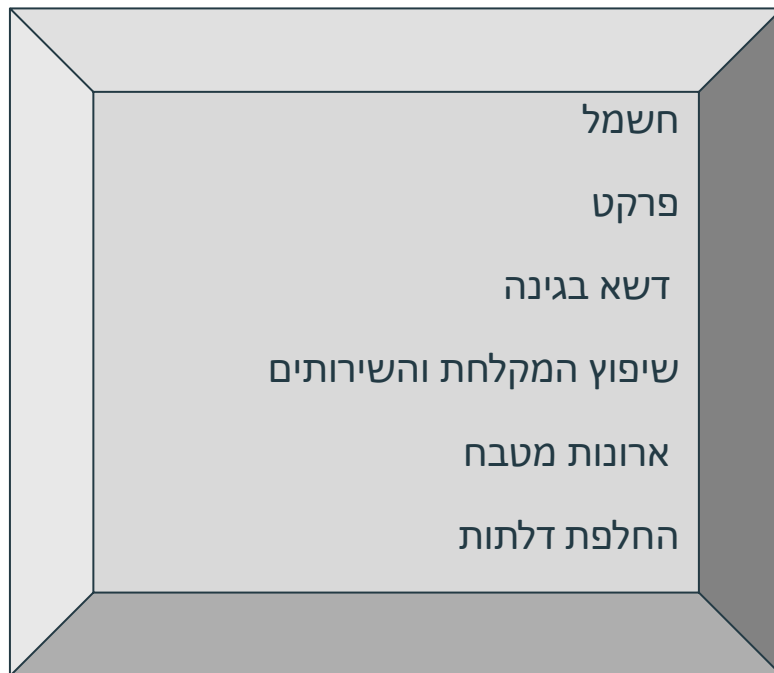
מה שנעשה הוא לפרק את כל השיפוץ לעבודות משניות בתוך השיפוץ.

איך זה יראה ??



אתגרים מורכבים - ניהול זמנים

כפי שאמרנו קודם -ניקח את כל עבודות השיפוץ כמכסה אחת.



אתגרים מורכבים - ניהול זמנים

נפרק את כל השיפוע הגדול לעבודות נפרדות, כל דבר יעשה בנפרד.

שירותים
ומקלחת

דשא
בגינה

דלתות

פרקט

ארונות
מטבח

חשמל

אתגרים מורכבים ניהול זמנים

כאשר נפרק את כלל השיפוע לעבודות נפרדות יהיה לנו קל יותר לנהל את כלל השיפוע אם זה בתכנון הזמנים וניהול אנשי המקצוע.

לכל עבודה תהיה את הזמן שלה ואנשי העבודה שלה.

בעולם התכנות - עבודת השיפוע היא התכנית הראשית שלנו, בכל פעם התכנית שלנו(עבודת השיפוע) תזמן עבודה אחרת לשיפוע הבית (פעולה)



מהי פעולה?

לעיתים, האפליקציה שאנו מפתחים מבצעת משימות מורכבות כך שלכל משימה יש תת משימות.

כדי להתמודד עם משימה מורכבת וגדולה נוכל לחלק כל משימה גדולה ומורכבת ולחלק אותה לתת משימות קטנות.

תתי המשימות נקראות **פעולות**.

לכל תת משימה- פעולה יש מטרה, אופן ביצוע וזמן ביצוע (מתי תקרה).

דוגמא לפעולה: האם מספר מתחלק ב-5 ללא שארית (נעזר באופרטור מודולו).

על מנת להשתמש בפעולות נשתמש בתת קטגוריה Procedure במסך ה-Blocks, ניצור פרוצדורה (פעולה) ונוכל לתפעל אותה.

The image shows the Scratch interface. On the left is the 'Blocks' palette with a 'Built-in' category expanded, showing various block categories: Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. The 'Procedures' category is highlighted. On the right is the 'Viewer' area showing three procedure blocks in the script area. The first two are 'to procedure' blocks with 'do' and 'result' outputs. The third is a 'call' block with a dropdown menu showing 'שם הפעולה'. Three arrows point from the Hebrew labels on the right to these blocks.

Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures

Viewer

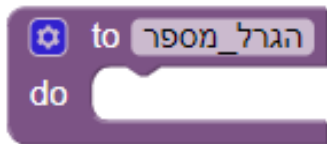
- to procedure do
- to procedure result
- call שם הפעולה

הגדרת פעולה

הגדרת פעולה המחזירה תשובה

תפעול הפעולה (זימון, הפעלה)

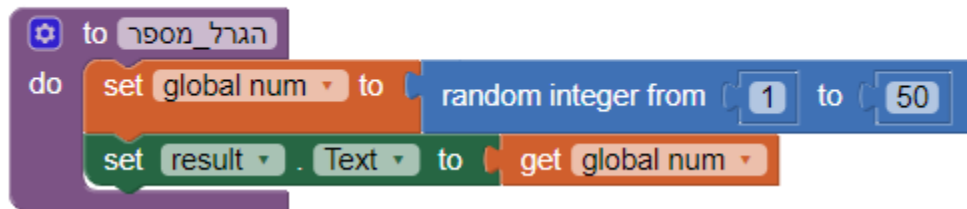
הגדרת הפעולה



נגדיר פעולה עם שם ייחודי ובתוכה נגדיר מה מטרת הפעולה.

בדוגמה מטה, נרצה שבפעולה יוגרל מספר אקראי בין 1-50, שערך המספר שהוגרל ייהפוך לערך המשתנה.

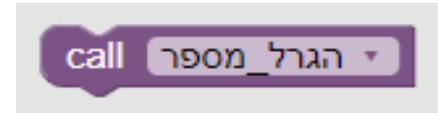
initialize global num to 0



זימון הפעולה

בעת הגדרת הפעולה נגדיר מה מטרת הפעולה, על מנת להשתמש בפעולה נצטרך לזמן את הפעולה.

איך נזמן פעולה?



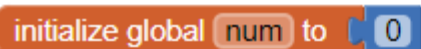
נגרור את הבלוק call כדי לזמן את הפעולה, נגרור את הבלוק למקום בו נרצה שהפעולה תתרחש.

תרגיל כיתה

צרו פרויקט שבו Button ו-Label שבתכונת הטקסט שלו רשום "Procedure".
רשמו בתכונת הטקסט של ה-Button "Click Me".

מטרת הפרויקט להשתמש בפעולה שמגרילה מספר אקראי בטווח של 15-40 בעת לחיצת כפתור.

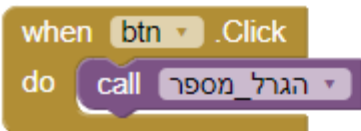
בנוסף, על המשימה לשנות את גודל הטקסט של ה-Label למספר שעלה בהגרלה!

A Scratch code block with an orange background. It contains the text "initialize global num to" followed by a blue block with the number "0".

נאתחל משתנה עם ערך 0.

A Scratch code block with a purple background and a gear icon. It starts with "to הגרל מספר" (when green flag clicked). Inside a "do" loop, there are two blocks: "set global num to random integer from 15 to 40" and "set result FontSize to get global num".

נגדיר את הפעולה שלנו, הפעולה תגדיר מספר בין 15-40 ותשנה את גודל הטקסט ב-Label למספר שעלה בהגרלה.

A Scratch code block with a yellow background. It starts with "when btn Click" and contains a "do" loop with a block "call הגרל מספר".

בכל פעם אשר נלחץ על הכפתור נקרא לפעולה והיא תתבצע.

פעולה המקבלת ערכים

עד כה ראינו הגדרה לפעולה אשר מבצעת מה שנכתוב לה לבצע, אך היא לא מקבלת נתונים בעת הגדרתה.

נוכל להגדיר פעולה אשר בעת הגדרתה תקבל נתונים/פרמטרים (טקסטים, מספרים וכו.).

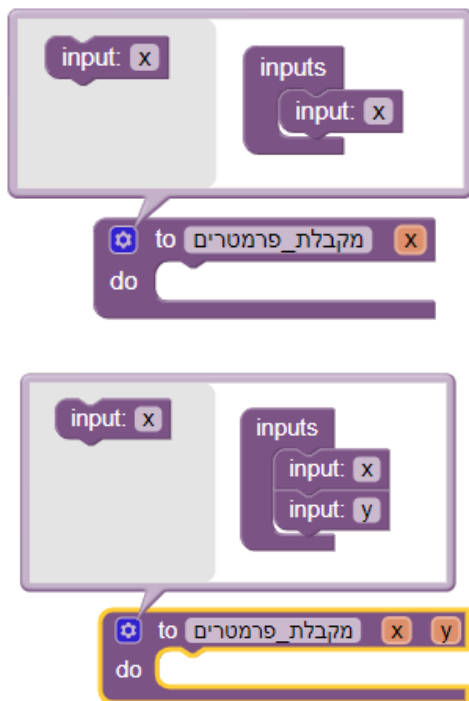
לדוגמה, נוכל להגדיר פעולה שתקבל בהגדרתה את המספר 22 והמספר 7.

הפעולה תוכל להשתמש או לא להשתמש במספרים, נהוג להשתמש בפרמטרים שמקבלים בפעולה בעת הגדרת פעולה שמקבלת נתונים.

נוכל להגדיר פעולה עם כמה פרמטרים שנרצה.

על מנת להוסיף פרמטרים נלחץ על גלגל השיניים ונגרור input לתוך inputs.

כל נתון ישמש כמשתנה באפליקציה שלנו.



תרגיל כיתה

צרו פרויקט שבו 2 משתנים בעלי ערך כלשהו (אם בתור משתנים סטטים או כקלט בתיבת טקסט).

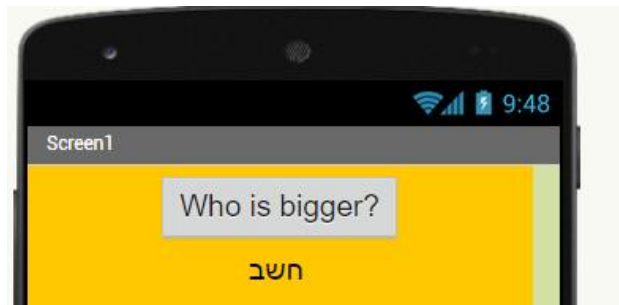
גרו למסך כפתור ושנו את שמו לשם `who is bigger`.

בנוסף, גרו Label שבו תכונת הטקסט ריקה.

לבסוף, גרו פעולה למסך ה-Block וקראו לה "חשב".

על הפעולה לקבל את 2 המשתנים, בעת לחיצת כפתור על הפעולה לבדוק איזה מי מהמשתנים גדול יותר ולרשום הודעה מתאימה.

נסו בעצמכם!



נגרור כפתור + לייבל.
נשנה את תכונות הטסקט שלהם

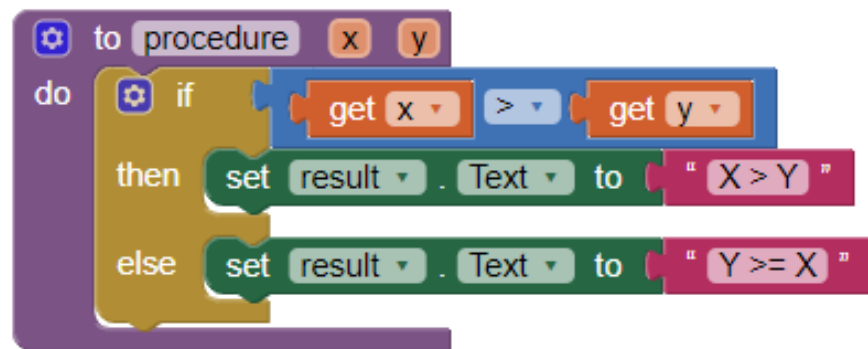
initialize global x to 22

initialize global y to 18

when btn.Click
do
call procedure
x get global x
y get global y

נאתחל שני משתנים סטטים וניתן להם
ערך

בעת לחיצה על כפתור נקרא לפעולה ונשלח
אליה את X ו-Y.
שימו לב - לפני כן נצטרך להגדיר את הפעולה.



נגדיר את הפעולה עם משתנים X ו-Y.
נבדוק האם $X > Y$ ונדפיס הודעה אם כן.
אחרת נדפיס הודעה שונה.

פעולה המחזירה ערך

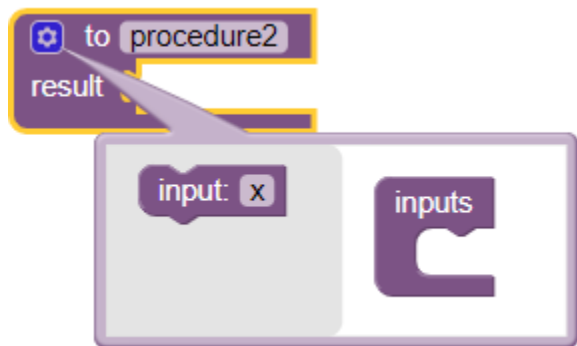
עד כה ראינו פעולות שמוגדרות ללא פרמטרים ועם פרמטרים.

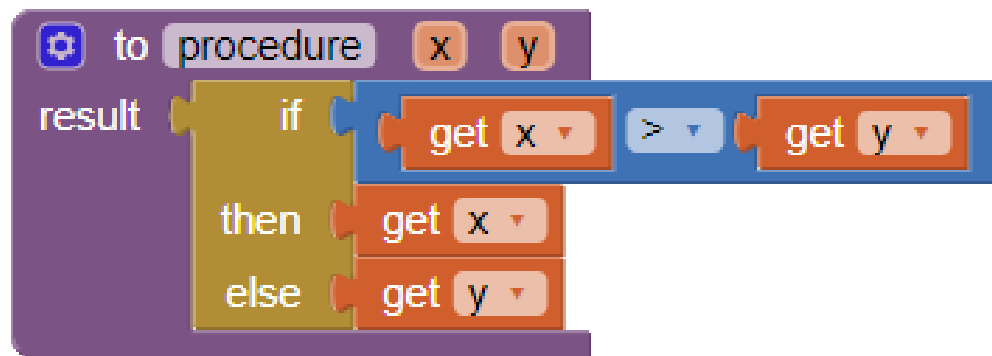
פעולה יכולה לקבל פרמטרים וגם להחזיר תשובה בסופה.

לדוגמא, ניקח את הדוגמא הקודמת, הפעולה קיבלה את המספרים 22 ו-7, תבדוק מי המספר הגדול מביניהם ותחזיר את המספר 22.

איך נראית פעולה המחזירה ערך?

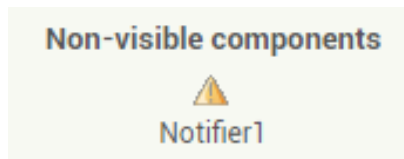
ההבדל הוא שלעומת הפעולות הקודמות בפעולה זו נקבל תשובה - `.result`.



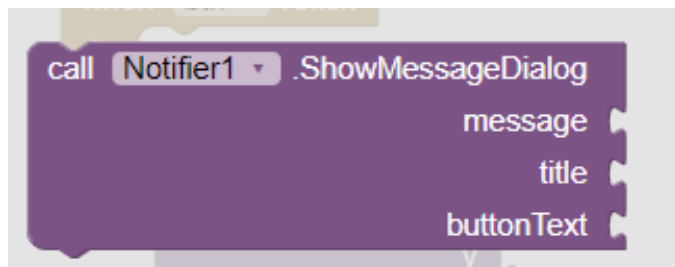


נבדוק מי הערך הגדול יותר,
ונחזיר מהפעולה את הערך
הנכון.

רכיב Notifier



רכיב Notifier הוא רכיב המאפשר לנו לשלוח הודעה או אזהרה למסך.
רכיב זה הוא רכיב בלתי נראה, כלומר אחרי שנגרור אותו למסך לא נראה אותו ויזואלית.



בדרך כלל נשתמש ברכיב עם תכונה זו.
בתכונה זו הרכיב מורכב מההודעה עצמה,
כותרת וכפתור.
בכל אחד מהם יהיה כתוב טקסט.

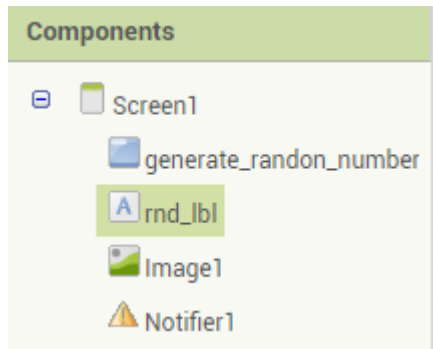
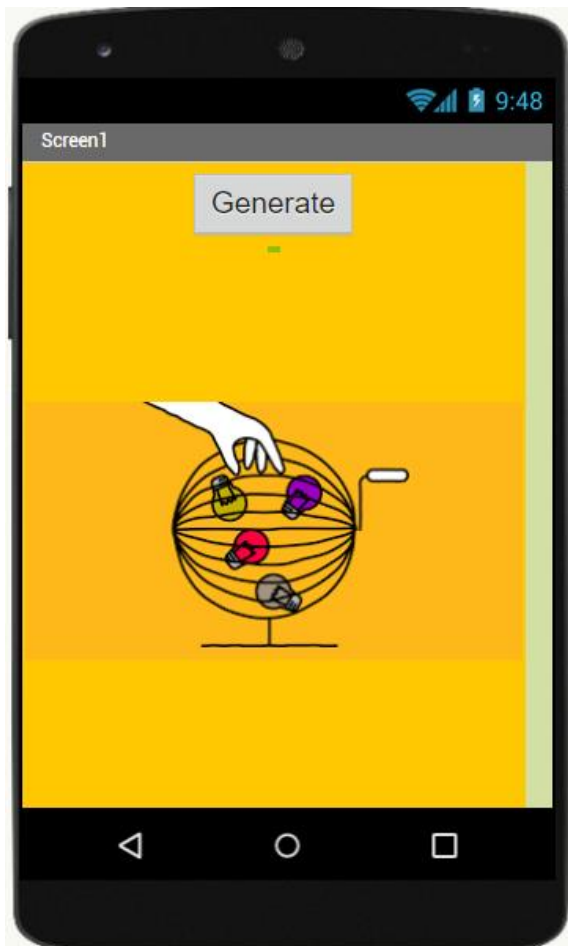
תרגיל כיתה

צרו פרויקט שבו : Button, Label ו-Notifier.
בעת לחיצה על כפתור יוגרל מספר אקראי בין 1-100.

על התוכנית להציג את המספר שהוגרל ב-Label.

בנוסף, אם המספר שהוגרל גדול מ-50 רכיב ה-Notifier יציג הודעה שהמספר גדול מ-50, אחרת יוציא הודעה שהמספר קטן שווה מ-50.

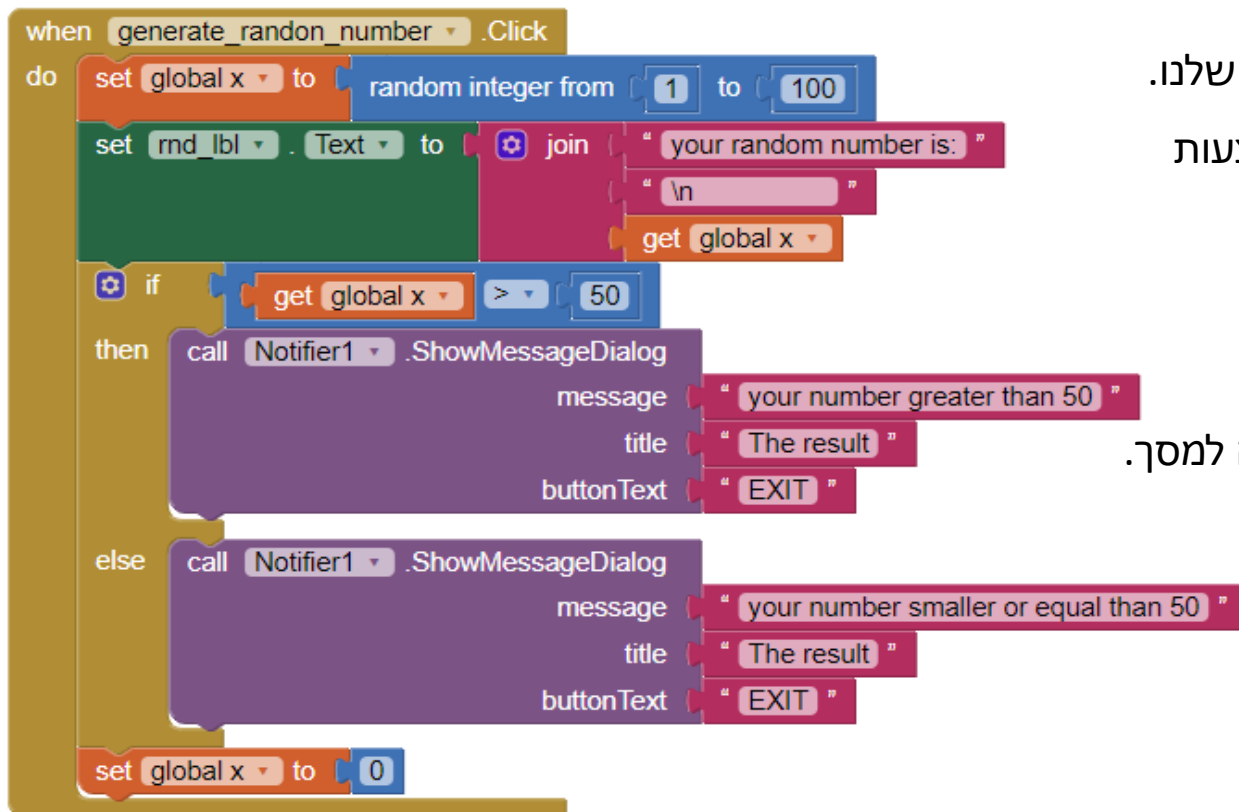
ניתן להוסיף תמונה לאפליקציה ().



נוסיף את הרכיבים הללו, נשנה את השמות ותכונות
הטקסט.

initialize global x to 0

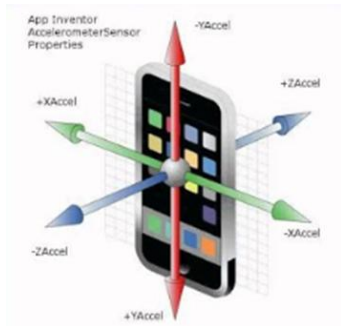
נאתחל משתנה ל-0



נגריל מספר אקראי למשתנה X שלנו.
נציג את המספר שהוגרל באמצעות
פעולת איחוד.

אם המספר גדול מ-50, רכיב ה-
Notifier יקפיץ הודעה מתאימה למסך.

אחרת, כלומר המספר קטן
שווה מ-50, רכיב ה-Notifier
יקפיץ הודעה מתאימה למסך.



סנסור ACCELERATOR

רכיב המאפשר לאפליקציה לחשוש בתזוזות ושינויים.

לדוגמה, הסנסור יכול לזהות Shaking של המסך.

הרכיב נמצא תחת תת תפריט Sensors, הרכיב שנגרור למסך הוא : AccelerometerSensor .

הרכיב הוא רכיב בלתי נראה, כלומר לאחר שנגרור אותו למסך לא נראה אותו ויזואלית.

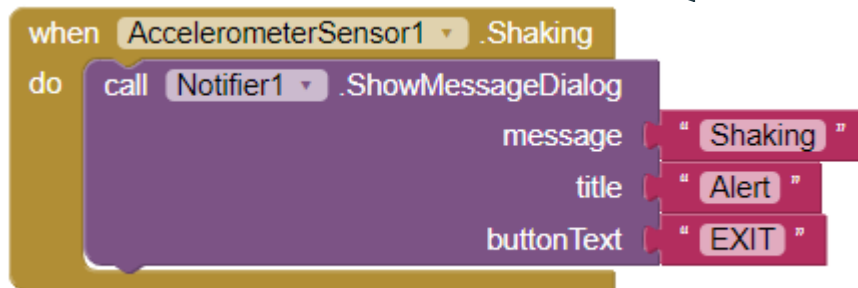
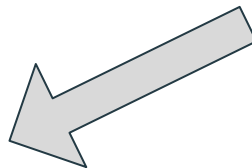
Non-visible components



AccelerometerSensor1

דוגמא

ניצור תכנית פשוטה שמזהה ניעור ושינוי ציר X, Y של המכשיר שלנו.



סיכום - מה למדנו?

- בעזרת פעולות נוכל לנהל את כל הקוד בצורה נקייה ומסודרת
- נוכל לזמן פעולה עם ובלי פרמטרים בהתאם למה שנצטרך
- נוכל לשלב רכיב Notifier שיאפשר לנו לשלוח הודעה או אזהרה למסך

תרגיל מסכם

צרו פרויקט שמטרתו להיות "משחק הניחוש".
מטרת הפרויקט להגריל מספר בין 1-50, על המשתמש לנחש מה המספר שהוגרל ב-5 ניחושים.
בכל ניחוש, האפליקציה תודיע למשתמש (בעזרת רכיב Notifier) אם המספר שהוגרל שווה למספר שהוגרל, קטן ממנו או גדול ממנו.
לאחר 5 ניסיונות ניחוש, אם המשתמש לא הצליח לנחש את המספר אז נגלה למשתמש מה המספר ונציג לו הודעת כישלון.

THANK
you