הנדסת תוכנה - מועד ב' 2018

חלק ראשון - חיי מערכת תוכנה

1. בתוכנה הפופולארית "תנו לגדול בשקט" ישנם 10,000 משתמשים רשומים המשתמשים במערכת בתדירות של 10 שעות בשבוע.

לאחרונה התגלה באג במערכת התוכנה, התקלה נותחה ותוקנה.

חישבו ומצאו שעלות תיקון התקלה הגיעה ל-500,000 ש"ח.

אילו בשלב הגדרת הדרישות היו מוסיפים דרישה שהייתה מונעת את התקלה, כמה להערכתך הייתה עלות הטיפול? **(5 נק')**

א. כ-500,000 ש"ח.

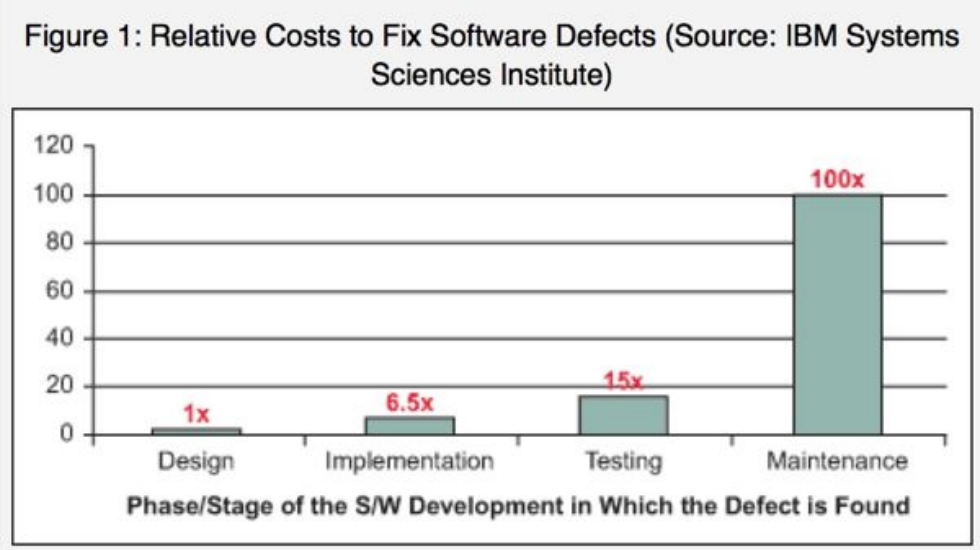
ב. כ-50,000 ש"ח.

ג. כ-5,000-10,000 ש"ח.

ד. 0 ש"ח, או סכום זניח אחר.

ה. אין שום בסיס להערכה שכזו ולא ניתן לחשב את העלויות.

תשובה



לפי חוברת המבחנים התשובה היא "ג" - זה גם מתאים לגרף המצורף הלקוח מהמצגת של שלבי הייזום והדרישות.

עלות התיקון בשלב ה-Maintenance (השלב אחרי שהתוכנה כבר בשימוש) הייתה 500,000 ש"ח, לכן אם היינו מטפלים בבאג בשלב הדרישות (Design בגרף) עלות התיקון הייתה יכולה להיות חלקי 100, כלומר 5,000.

2. הגדר מהי דרישה פונקציונלית ומהי דרישה לא פונקציונלית.

תן דוגמא לכל אחת מ-2 סוגי הדרישות המוזכרות לעיל. (6 נק')

תשובה

דרישה פונקציונלית - מה המערכת אמורה לעשות/להגיב מנקודת המבט של המשתמש.

לדוגמא:

המערכת תאפשר להזין הזמנות מלקוח למוצרים שבמלאי.

היא תייצר מספר הזמנה חד ערכי בעת שמירת ההזמנה.

כל הזמנה תאפשר לציין למוצר יחידת מידה וכמות.

כל שינוי ביחידת מידה מחייב רישום כפריט נפרד בהזמנה.

לכל פריט בהזמנה יש לרשום יחידת מידה, כמות ומחיר ליחידת מידה.

ניתן להזמין מוצרים הן דרך מרכז ההזמנות והן בצורה ישירה באינטרנט.

עבור כל פריט שיוצג יש להציג את שמו, הברקוד שלו וצבעו).

דרישה לא פונקציונלית - דרישות המגדירות תכונות נוספות של הפתרון שצריכות להתמלא תוך כדי מילוי הדרישות הפונקציונליות. או: דרישות ותנאים המגבילים את חופש בחירת כיווני הפתרון.

לדוגמא: זמני תגובה/ביצוע, נפחי פעילות, אמינות, אבטחת מידע, אופני מימוש, תדירות ביצוע, עמידה בעומסים ,שימושיות.

לדוגמא:

המערכת תתבסס על מחשב מסוג X.

המערכת תהיה זמינה ** שעות ביממה.

עלויות הפיתוח לא יהיו גבוהות מ- $200M.

תאריך היעד של המערכת הוא 01/01/2018.

יש להשתמש בחישוב הפרמיה החודשית כפי שמחושב במערכת הקיימת.

יש להחזיר תשובה לחיפוש תוך 2 שניות לכל היותר.

3. חברת תוכנה מפתחת מוצרי תוכנה עבור לקוחותיה.

כל המוצרים מבוססים על מוצר בסיסי. כאשר מגיע לקוח חדש לחברה מציגים לו את המוצר הבסיסי וביחד איתו כותבים את מפרט הדרישות להתאמת המוצר לצרכיו הספציפיים. פיתוח מוצר הבסיס הינו באחריות "צוות בסיס" אחד בחברה, ובכל פרויקט נמצא צוות פיתוח ייעודי להתאמת המוצר. חלק מהפיצ'רים שנוספו לצרכי התאמות נכנסים, עם הזמן, למוצר הבסיסי.

לפי תיאור זה, מהו מודל העבודה לפיו מפותחים מוצרי החברה? **(4 נק')**

א. Code and Fix.

ב. מודל ספירלי.

ג. מודל V.

ד. Extreme Programming.

<u>תשובה</u>

לפי חוברת המבחנים התשובה היא "א" , Code and Fix.

4. נתונה רשימת הפעולות בפרויקט מסוים: (8 נק')

פעולה	תלויות	משך אופטימי	משך פסימי	משך סביר
A	-	5	15	13
C	-	2	18	7
E	C	1	3	2
F	C	4	8	6
G	E, A	1	21	2
H	E, F	2	16	3
I	H	3	5	4

א. מהו הנתיב הקריטי בפרויקט? מהו משכו? נמקו בעזרת הסבר, גרף או תרגילים מתאימים. (4 נק')

ב. תנו דוגמא לפעולה עם slack = 0 והסבירו מדוע. (2 נק')

ג. תנו דוגמא לפעולה עם slack > 0 והסבירו מדוע. (2 נק')

<u>תשובה</u>

א. הנטיב הקריטי הוא C-F-H-L. משך הנטיב 23.

הנטיב הקריטי הוא הנטיב שמכיל את הפעולות שה-Slack שלהם הוא 0, והוא הנטיב הארוך ביותר.

כדי למצוא אותו צריך למצוא את ה-Slack של כל הפעולות. לכן נצטרך למצוא את לכל הפעולות את הערכים הבאים: Duration, Early Start, Early Finish, Late Start, Late Finish.

בעזרתם נחשב את ה-Slack.

Duration - כמות הזמן שלוקח לבצע את הפעולה (משך הפעילות).

Early Start - נקודת הזמן המוקדמת ביותר בה יכולה להתחיל הפעולה בהתחשב בתלויות.

Early Finish - נקודת הזמן המוקדמת ביותר בה יכולה להסתיים הפעולה בהתחשב בתלויות.

Late Start - נקודת הזמן המאוחרת ביותר בה יכולה להתחיל הפעולה בלי להאריך את משך הפרויקט.

Late Finish - נקודת הזמן המאוחרת ביותר בה יכולה להסתיים הפעולה בלי להאריך את משך הפרויקט.

Slack - כמות הזמן שהפעולה יכולה להתעכב בלי להאריך את משך הפרויקט.

Duration (לפי נוסחת PERT) = (1/6) * (Optimistic + 4*Likely + Pessimistic)

Early Finish = Early Start + Duration

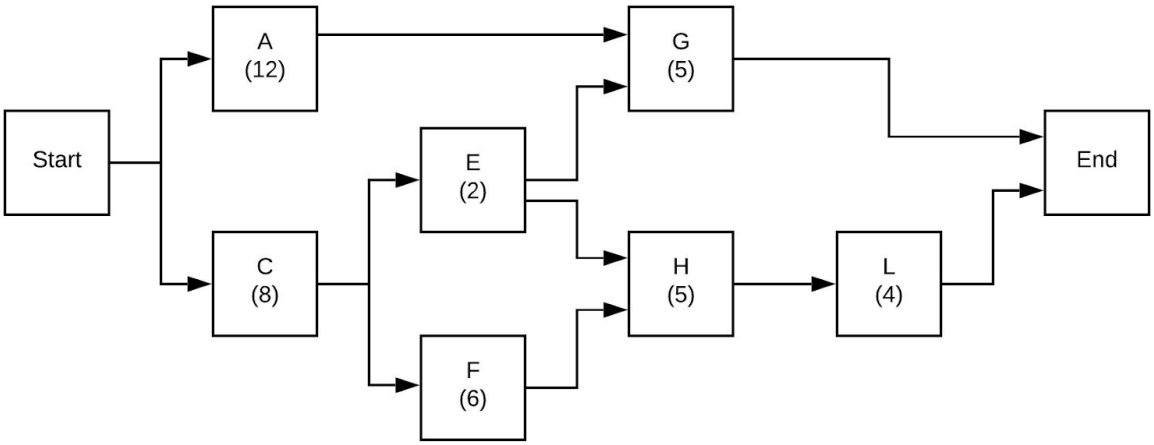
Late Start = Late Finish - Duration

Late FInish = ה-LS המינימלי מבין הפעולות התלויות בפעולה

Slack = Late Start - Early Start

תרשים רשת של הפרויקט:

(המספר מתחת לשם הוא משך הפעולה)



	Early Start	Early Finish	Late Start	Late FInish	Slack
Start	0	0	0	0	0
A	0	12	6	18	6
C	0	8	0	8	0
E	8	10	12	14	4
F	8	14	8	14	0
G	12	17	18	23	6
H	14	19	14	19	0
L	19	23	19	23	0
End	23	23	23	23	0

ב. דוגמא לפעולה עם slack = 0 היא פעולה L.

L היא הפעולה האחרונה בפרויקט ולכן אם היא תתעכב, משך הפרויקט יגדל.

ג. דוגמא לפעולה עם slack > 0 היא פעולה A.

הפעולה היחידה שתלויה ב-A היא פעולה G, פעולה G מתחילה כאשר פעולה E מסתיימת, אך פעולה A מסתיימת לפני פעולה E. לכן אם פעולה A תתעכב, היא עדיין תוכל להסתיים לפני פעולה E (או באותו הזמן) ומשך הפרויקט לא ישתנה.

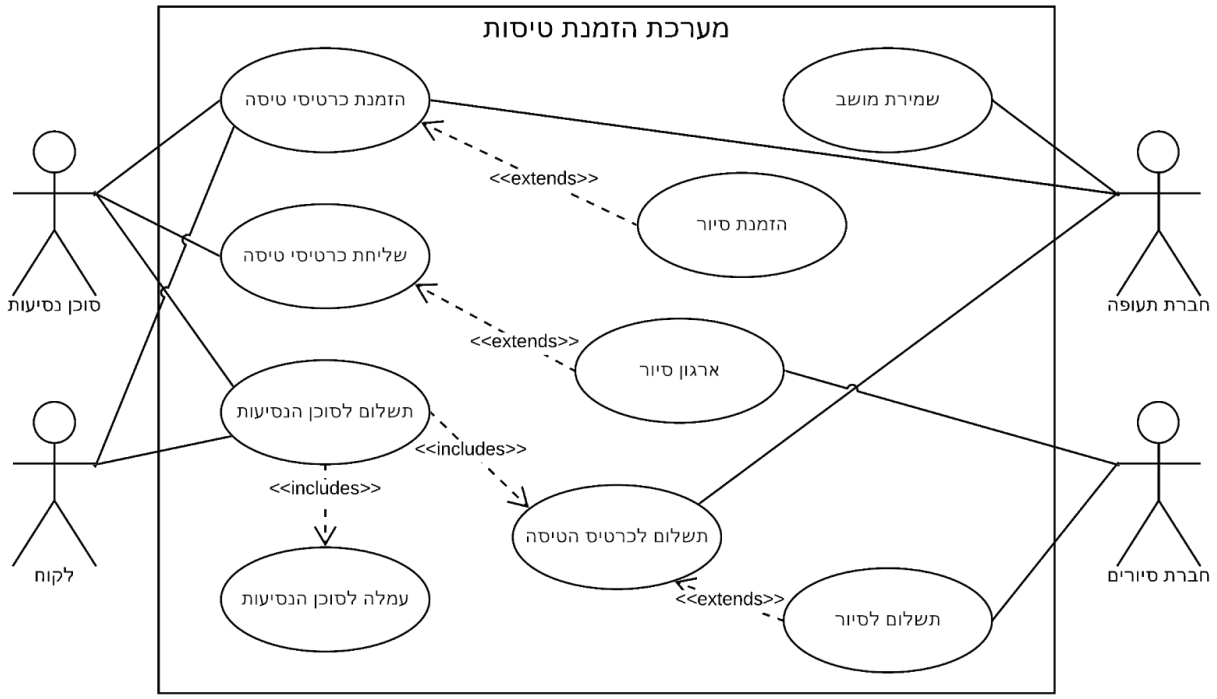
5. מה מייחד את מודל העבודה XP בהיבט של הבדיקות? (4 נק')

<u>תשובה</u>

XP משתמש בשיטת פיתוח מונחה-בדיקות שעיקריה הם כתיבת דרישות המערכת כסט של בדיקות הניתנות להרצה, ופיתוח הבדיקות קודם לפיתוח הפונקציונליות.

כלומר, יוצרים בדיקות שבודקות את כל הדרישות שלנו מהקוד, וממשיכים לכתוב קוד ולתקן אותו עד שהקוד עובר את כל הבדיקות, לעומת הדרך הרגילה בה קודם כותבים קוד, ורק אז כותבים בדיקות.

6. נתון Use Case Diagram ובעקבותיו 10 משפטים. לגבי כל משפט סמן "נכון" אם הוא נובע מהאמור בתרשים, "לא נכון" אם הוא נמצא בסתירה לתרשים ו"לא יודע" אם לא ניתן לדעת מהתרשים האם הוא נכון או לא. (15 נק')



	משפט	תשובה
1	לקוח לא יכול להזמין כרטיסי טיסה ישירות, סוכן הנסיעות הוא זה שמזמין עבורו.	**לא נכון**. קיים קו המקשר בין הלקוח לבין "הזמנת כרטיסי טיסה".
2	תשלום על כרטיס הטיסה, תמיד יכלול תשלום לסוכנות הנסיעות.	**נכון**. "תשלום לכרטיס הטיסה" מוכל ב"תשלום לסוכן הנסיעות".
3	לקוח לא יכול להזמין טיסה ללא הרשמה בסוכנות הנסיעות.	**לא ידוע**. הרשמה בסוכנות הנסיעות לא מוזכרת בדיאגרמה.
4	לא ניתן להזמין סיור כי לא הוגדר מי השחקן שלו.	**לא נכון**. "הזמנת סיור" מרחיב את "הזמנת כרטיסי טיסה" ששחקניו מוגדרים. (לפי ההגיון "הזמנת סיור" אמור להיות מחובר ל"חברת סיורים", אך מבחינה טכנית עדיין ניתן לגשת ל-use case הנ"ל)
5	חברת התעופה לא יכולה לשריין מושב טיסה ללא סוכנות הנסיעות.	**לא נכון**. "שמירת מושב" מחוברת רק ל"חברת תעופה" ולא מחוברת ל"סוכן נסיעות".
6	לקוח יכול לשלם רק עבור טיסה שהוא הזמין.	**לא ידוע**. לקוח א' יכול לגשת ל"הזמנת כרטיס טיסה" ולקוח ב' יכול לגשת ל"תשלום לכרטיס הטיסה", ואולי לקוח ב' יכול לשלם על הכרטיס שלקוח א' הזמין, אולי לא. אי אפשר לדעת בלי מידע נוסף.
7	לא ניתן להזמין כרטיס נוסף אם הוזמן כרטיס וטרם שולם.	**לא ידוע**. לקוח יכול לגשת ל"הזמנת כרטיס טיסה" גם אלף פעמים לפני שהוא ניגש ל"תשלום לכרטיס הטיסה" לפי כל מה שידוע לנו, לא מפורטת שום מגבלה או בדיקה מסוג כלשהו.

8	לא ניתן לשלם על סיור ללא תשלום על כרטיס הטיסה.	**נכון**. "תשלום לסיור" מרחיב את "תשלום לכרטיס הטיסה", כלומר, אם משלמים על כרטיס טיסה, יש אפשרות גם לשלם על סיור, אך אי אפשר להגיע לאפשרות הזאת בלי לעבור קודם ב"תשלום לכרטיס הטיסה".
9	לעולם השחקנים שמצד שמאל מייצגים את ה-input והשחקנים מימין את ה-output.	**לא נכון**. "חברת תעופה" יכולה לגשת ל"שמירת מושב" אשר לא מקושר לשום שחקן אחר, כלומר השחקן הראשי (input) במקרה זה הוא "חברת תעופה". לעומת שחקן משני (output) שעושה משהו רק כתגובה למשהו ששחקנים אחרים עשו.
10	הקשר בין השחקן "חברת תעופה" ל-use case "תשלום כרטיס טיסה" משמעו שהלקוח משלם ישירות לחברת התעופה.	**נכון**. קיים קשר ישיר בין "תשלום לכרטיס הטיסה" לבין "חברת תעופה" בלי use cases אחרים ביניהם.

7. השאלה מתעסקת בתבניות עיצוב שירדו מהחומר ב2020.

חלק שני - אנדרואיד

מצורפים כמה קבצי קוד ללא שמות הקבצים/מחלקות הרלוונטיים.

בחלק זה תשאלו על הבנתכם את הקודים השונים ואת הקשרים ביניהם. כמו כן, יכול להיות שיהיו שאלות בחלק זה שלא יהיו קשורות ישירות לקוד המופיע לפניכם. קבצי קוד אלו דומים לקבצים שראיתם בקורס, אך לא זהים. בסוף כל סעיף ושאלה מופיע הניקוד הרלוונטי.

1. קובץ מס' 1: **(13 נק')**

```
<application android:icon="@mipmap/ic_launcher" android:label="app">
<activity android:name="APA"
android:label="@string/app_name"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
            <intent-filter>
                  <action android:name="android.intent.action.MAIN" />
                  <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
      </activity>
</application>
<uses-sdk android:minSdkVersion="5" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

א. מהו סוג הקובץ המדובר?

יש לתת תיאור מדויק של סוג הקובץ ומטרתו, ובמידה ומדובר על תת סוג, יש להזכיר גם אותו.

למען הבהירות, לא מספיק להגיד שקובץ מסוים הוא קובץ pdf (אם אכן מדובר על שכזה), אלא יש להגיד איזה קובץ pdf ספציפי. **(5 נק')**

ב. הסבירו את שורה 3. **(3 נק')**

ג. הוצע להחליף את שורה 14 (המודגשת) בשורה הבאה:

```
<uses-feature android:name="android.hardware.bluetooth" />
```

מהי משמעות החלפה זו? **(5 נק')**

<u>תשובה</u>

א. זהו קובץ manifest הכתוב ב-xml.

קובץ ה-manifest מכיל מידע חשוב על האפליקציה. לדוג': שם האפליקציה, גרסת האנרואיד המינימלית שהאפליקציה תומכת בה, אילו activities קיימים באפליקציה, מה ה-activity ההתחלתי, אילו הרשאות האפליקציה מבקשת ועוד הרבה.

ב. label קובע את השם של ה-activity שיופיע ב-action bar בחלק העליון של המסך של האפליקציה.

ג. uses-permission אומר שהאפליקציה מבקשת רישות להשתמש ברכיב מסוים, אך גם מכשירים אשר לא מכילים את הרכיב יוכלו לראות את האפליקציה בחנות של גוגל.

uses-feature אומר שהאפליקציה משתמשת ברכיב מסוים, והחנות של גוגל תראה את האפליקציה רק למכשירים אשר מכילים את הרכיב.

2. קובץ מס' 2: **(15 נק')**

```
...
protected void onCreate(Bundle savedInstanceState)
{
       super.onCreate(savedInstanceState);
3      setContentView(R.layout.activity_main);
       mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);
       String userFavoriteFood = getUserFavoriteFood();
       if (userFavoriteFood == null) {
              askFavoriteFood();
       } else {
              setUserFavoriteFood(userFavoriteFood);
       }
}


private void recordImageView()
{
       String id = getCurrentImageId();
       String name = getCurrentImageTitle();
14     Bundle bundle = new Bundle();
15     bundle.putString(FirebaseAnalytics.Param.ITEM_ID, id);
16     bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, name);
17     bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "image");
18     mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
}
...
```

א. הסבירו את הביטוי המודגש בשורה 3. **(4 נק')**

ב. הסבירו את המתרחש בשורות 14-18 (כולל). **(5 נק')**

ג. מתודה דומה ל-putString היא putExtra.

היכן משתמשים במתודה זו? מה מטרתה? וכיצד משתמשים בה? **(6 נק')**

<u>תשובה</u>

א. R זה מזהה של הקובץ עם כל המשאבים של האפליקציה (קבצי xml, תמונות, מחרוזות...), layout מתייחס לתקייה המכילה את קבצי ה-xml שמתארים את מראה ה-activities של האפליקציה. activity_main הוא אחד מקבצי ה-layout של האפליקציה.

ב. bundle הוא אובייקט משמש להעברת מידע בין activities, הוא יכול להכיל אובייקטים מסוגים שונים בו זמנית (string, int, float...). במקרה זה אנו שמים בתוך הbundle שיצרנו שלוש מחרוזות הקשורות למאכל האהוב על המשתמש (ID, שם ותמונה).

FirebaseAnalytics מאפשר לנו לשלוח מידע אנליטי לגבי אירועים שמתרחשים באפליקציה שלנו לשרת ה-firebase שלנו. במקרה זה הודענו על כך שהתרחש אירוע מסוג SELECT_CONTENT (המשתמש בחר ערך מסוים, במקרה זה המאכל האהוב עליו) בעזרת logEvent, וצירפנו את ה-bundle שיצרנו שמכיל מידע על הערך שהמשתמש בחר (במקרה זה מידע על המאכל האהוב עליו).

ג. putString מוסיף ל-bundle ערך מסוג מחרוזת. putExtra מצרף ערך מסוים (או bundle של ערכים) ל-intent, ואז אפשר לגשת ל-extras האלו ב-activity שה-intent מוביל אליה, בעזרת getIntent().getExtras() שיחזיר אוביקט המכיל את המידע שהוספנו בעזרת putExtra.

2. קובץ מס' 2: **(22 נק')**

```
...
StorageReference storageRef = storage.getReference();
StorageReference forestRef = storageRef.child("images/forest.jpg");
forestRef.getMetaData().addOnSuccessListener(new OnSuccessListener<StorageMetadata>() {
        @Override
        public void onSuccess(StorageMetadata storageMetadata) { }
        }).addOnFailureListener(new OnFailureListener() {
        @Override
9       public void onFailure(@NonNull Exception exception) {}
        });
...
```

א. הסבירו את המתרחש בשורות 1-2 (כולל). הסבירו את הקשר בין שורות אלו למבנה מסד הנתונים איתו עבדנו בקורס. **(7 נק')**

ב. בשורה 3 מופיע ביטוי הקשור לנושא שלמדנו בקשר למסד הנתונים בקורס (Metadata).
הסבירו את המושג, ואת משמעותו בעניין מסד הנתונים. **(5 נק')**

ג. בשורה 9 מופיע ביטוי מודגש. בקורס למדנו את הביטוי המוזכר וביטוי נוסף הקשור אליו.
הסבירו את שני הביטויים, ואת השימוש השונה ביניהם. **(5 נק')**

ד. לגבי אחד מהביטויים הנזכרים בסעיף ג' למדנו כי יש צורך לעשות בדיקת קלט בצמוד לשימוש בו. מהי הבדיקה ולמה נצרכת? (אין צורך לכתוב קוד במפורש, אלא להסביר מהי הבדיקה וכו') **(5 נק')**

<u>תשובה</u>

א. FirebaseStorage הוא מסד נתונים שהוא חלק מ-Firebase שמטרתו לאחסן קבצים גדולים כמו תמונות וסרטונים, לעומת ה-Realtime Database או Firestore שמטרתם לאחסן מידע יותר פשוט (אוביקטים המורכבים ממחרוזות, מספרים וכו').

המשתנה storage הוא מהסוג FirebaseStorage, שזהו אוביקט המייצג את שרת האחסון שלנו.

```
FirebaseStorage storage = FirebaseStorage.getInstance();
```

StorageReference הוא סוג אובייקט המצביע על תיקיה או קובץ בתוך האחסון שלנו.

במקרה זה storageRef מצביע על התיקיה הראשית המכילה את כל הקבצים המאוחסנים שלנו.

forestRef מצביע על קובץ תמונה ספציפי באחסון שנמצא במיקום "images/forest.jpg".

ב. לכל קובץ באחסון שלנו יש metadata - מידע כללי שקשור אליו (שם הקובץ, גודל, סוג, תאריך יצירה ועוד).

המתודה getMetadata מחזירה לנו את המידע הזה.

ג. NonNull אומר לקומפיילר שהפרמטר שמתקבל פה לא יכול להיות null.

ביטוי נוסף הקשור אליו הוא not-null, שמצביע על כך שערך בשדה במסד נתונים לא יכול להיות null.

ד. נצטרך לבדוק שערך שאנחנו מנסים להכניס לשדה שהוגדר כ-not-null הוא באמת לא null לפני שאנחנו מכניסים אותו.

שאלת בונוס (**3 נק'**): **(לא בטוח אם זה רלוונטי לנו)**

במהלך הקורס, למדנו על האובייקט InstrumentationRegistry המתקבל כקלט ע"י אובייקט אחר.

האובייקט האחר מקבל את InstrumentationRegistry ע"י המתודה

InstrumentationRegistry.getInstrumentation().

קבלתו מאפשרת לבצע מספר פעולות. הסבירו איזה אובייקט מקבל את InstrumentationRegistry

כקלט, וכמו כן 6 פעולות הניתנות לביצוע ע"י אובייקט זה.

<u>תשובה</u>

זהו מופע של רגיסטר במכשיר שמחזיק את כל התהליכים שרצים במכשיר.