נכתב ע"י: אביה צ'ריקר :)

הנדסת תוכנה שיעור 11

מבוא לעקרונות תיכון ותבניות מונחי עצמים OODP

למדנו כמה נושאים בנושאי בדיקות.

איכות תוכנה – כוללת 2 סוגי מרכיבים:

1. מרכיבים חיצוניים – איך הלקוח רואה את התוכנה (בעלי ערך ללקוח, נכונות, יעילות, שמישות, עמידות, הרחבתיות- למקרה שירצה להוסיף פיצ'רים שזה יהיה קל).
2. מרכיבים פנימיים – יותר עניינים של הנדסת תוכנה (הקוד מודולרי, יש עקיבות בין החלקים, קל לקרוא וכו').

איך מגיעים לתוכנה איכותית?

צימוד – מדד כמה הדברים קשורים אחד לשני/תלויים – המטרה: כמה שיותר נמוך.

לכידות – אם עוסקים באותו עניין נרצה שכל אחד יקרה לכל אחד – המטרה: כמה שיותר גבוהה.

איך משיגים אותם ? עובדים עפ"י עקרונות ידועים, תבניות מוסכמות, הרגלים קבועים.

Legacy code – קוד שכבר קיים וצריך להסתדר איתו.

- עלות של שלב התוכנה זה 60% מהעלות הכוללת ש60% מזה זה מה שהלקוח ביקש לשנות.

גישות להתמודד עם קוד כזה:

1. לשנות ולהתפלל – לנסות דברים.
2. לכסות את הקוד בטסטינג (ששומרים עליי) ואז רק לעשות את השינויים.

כשכותבים קוד צריך לבדוק:

- אם הוא בדוק – האם כתבו טסטים עליו
- אם הוא בדיק – האם אפשר לכתוב עליו בדיקות.

נכתב ע"י: אביה צ'ריקר :)

<u>דרך להתמודד עם שינויים לאורך זמן:</u>

לעשות refactoring (כגון לעשות את כל הטסטינג שעשינו עד כה) כל פעם שמשנים משהו. המטרה שהטסטינג יכסו אותנו כמה שיותר וכך נדע שהקוד עושה את כל מה שהוא צריך

בדיקות אפיון –

אנחנו מבינים שאנחנו לא רוצים לבנות קוד חדש בלי טסטינג, מצד שני קיבלנו קוד ארוך שאין בו טסטינג, מצד שלישי – אנחנו לא יכולים לכתוב טסטינג בלי להבין את הקוד שכתבנו.

איך יוצאים מהמלכוד?

- באזורים שביקשו מאיתנו לשנות – נכתוב טסטינג שמתאר מה הקוד עושה כיום וככה מתקדמים לאט לאט.

<u>אין יודעים שמתודה שקיבלנו היא איכותית ?</u>

1. זה צריך להיות כמו סיפור שהולך ומתפתח ולא משפטים שמתערבבים
2. שיש לה מעט ארגומנטים – הרבה ארגומנטים (7-8) קשה לבדוק כי צריך לבדוק את כל האפשרויות של כל הארגומנטים וקשה לעקוב במהלך הבדיקות
3. ארגומנטים בוליאנים צריכים להיות דגל צהוב
4. אם אנחנו שמים לב שאנחנו מעבירים את אותם ארגומנטים לאותם פונקציות – סימן שאפשר להפוך אותם לקלאס חדש ובו להשתמש

<u>מה הכי חשוב עבור טסטינג מבין האפשרויות הבאות?</u>

1. שיהיה קצר
2. עושה דבר אחד
3. מעט ארגומנטים
4. להמשיך באותה רמה של אבסטרקציה

התשובה: 3

הרעיון בריפרקטורינג – קיבלנו קוד לא טוב ובמספר צעדים אנו הופכים אותו לקוד טוב

נכתב ע"י: אביה צ'ריקר :)

מה מהבאים אנחנו לא מנסים לעשות כאשר עושים רפרקטורינג לשיפור מתודה?

1. להוריד מסובכיות של הקוד
2. לתקן קוד
3. לתקן באגים
4. לשפר טסטינג

התשובה: 3 - לתקן באגים – קודם כל נעשה טסטים שמוצאים את הבאג ואח"כ נחשוב איך לתקן באג זה

איך עושים תחזוקה בפרויקטים יותר גדולים?

תחזוקה – הרבה פעמים הלקוח משלם סכום מסויים לתחזוקה של הקוד הזה והרבה פעמים זה פחות יקר

מה התפקיד של אחראי התחזוקה – זה בעצם תפקיד פיתוח, צריך לראיין אנשים לעבוד בצוות שלו, לתת להם הערכות, לתעד את התהליך שהם עושים וכו'

מה ההבדל בין פיתוח רגיל לפיתוח של תחזוקה?

בפיתוח של תחזוקה –

יש תוכנה שעובדת ופשוט רוצים להגיב לשינויים, בנוסף, רוצים שהיא לא תבוא על חשבון דרישות קודמות.

נכתב ע"י: אביה צ'ריקר :)

תרגיל 5

באופן כללי – בתרגיל נעשה Kata

הרעיון: לוקחים משהו קטן ועושים אותו שוב ושוב עד שנכנס לראש

הדרישה בתרגיל : לתעד את התהליך שעשינו באמצעות קומיטים, הטוויסט – שמשתמשים פה הרבה בגיט רברס – לחזור כמה קומיטים אחורה (סומכים על גיט ששומר את הגירסאות אחורה וחוזרים אליהם)

אפשר גם לעשות ב pair-programming

בתרגיל:

קיבלנו קוד שעובד עבור חברה אחת (mastercrupt) ואנו רוצים שיעבוד גם עבור חברה אחרת (גרסא עבור SO)

יש לנו 2 טסטינגים כבר.

פותחים פרויקט ג'אווה, עושים את הפרויקט ליד הפרויקט של החברה mastercrupt שעבורו עשינו clone