

מעבדה מספר 4

ניהול תצורה באמצעות GIT, בדיקות קבלה

חלק א': Git

1. מהו ניהול תצורה?

ניהול תצורה כולל שיטות, תהליכים וכלים לארגון ולמעקב אחר שינויים בתוצרים של תהליך פיתוח. כלי בקרת תצורה מאפשרים לצוות הפיתוח לעבוד במקביל על חלקים שונים בתוכנה, לשמור ולנהל גרסאות שונות, ציבוריות ופרטיות. קיימים כלים שונים. במעבדה זו נלמד לעבוד עם GIT על IntelliJ.

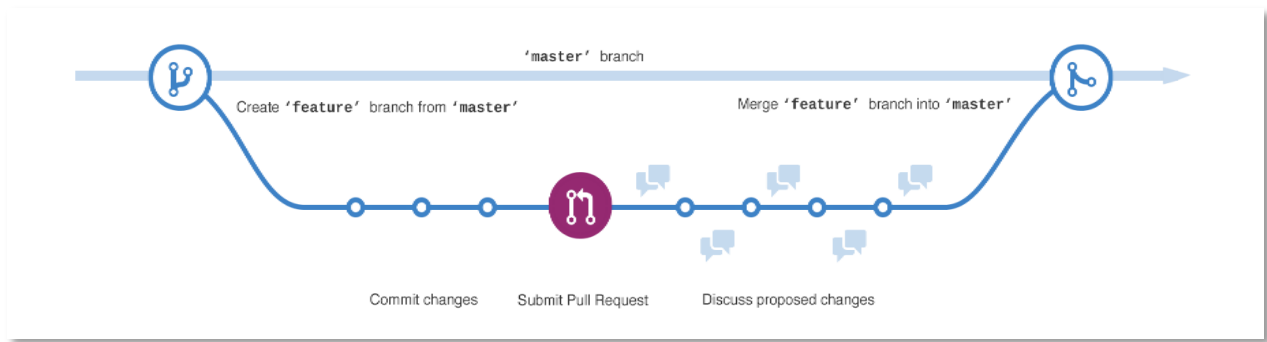
מטרת תהליך ניהול התצורה היא לאפשר לתת מענה לשאלות הבאות, ואחרות:

- מהי הגרסה הנכונה של מודול התוכנה שאני צריך להמשיך לקודד?
- כיצד אשתף את חברי לצוות בגרסת הקוד האחרונה שכתבתי ובדקתי?
- איך אנחנו יכולים לחזור לגרסה האחרונה שעבדה לנו לפני שהכנסנו שינויים רבים/תיקונים שגרמו לפונקציות רבות שעבדו להפסיק לעבוד?
- מהי הגרסה האחרונה שכוללת את כל השינויים האחרונים שעשינו?
- אילו שינויים הוכנסו בגרסה האחרונה שלנו?
- ואם עובדים בפיתוח מערכת מסחרית שנמצאת כבר אצל לקוחות:
 - מי יכול לספק לי עותק מדויק של גרסה 4.1 של התוכנה שסופקה ללקוח לפני שנה?
 - איזו גרסת תוכנה מותקנת אצל לקוח X?
 - היכן ניתן למצוא את רשימת כל הלקוחות אצלם מותקנת גרסה 6.8 של התוכנה שלנו?

2. מושגים בסיסיים בניהול תצורה:

2.1 Repository – אוסף המשמש לרוב לארגון פרויקט וכולל מחיצות, קבצים, תמונות ומסמכים הנדרשים לפרויקט. מומלץ לכלול גם קובץ README או כל קובץ אחר עם תיאור כללי של הפרויקט ומידע עליו.

2.2 Branching – הסתעפויות הן דרך לעבוד על גרסאות שונות של אותו repository באותו זמן. ברירת המחדל היא שלכל repository יש ענף אחד הנקרא master ומהווה ענף שלם של הפרויקט. משתמשים בענפים (הסתעפויות) כדי לערוך חלקים של הפרויקט לפני שמכלילים אותם בענף ה-master. כאשר יוצרים ענף של ענף ה-master, מעתיקים את ה-master כפי שהוא ברגע יצירת הענף. אם מישהו אחר עורך שינויים על ה-master בזמן זה, ניתן למשוך את השינויים לענף החדש. בתרשים שלהלן, ניתן לראות שעורכים מספר שינויים על הענף המסתעף לפני שהוא ממוזג חזרה לענף ה-master.



2.3 עותק עבודה (Working Copy)

אזור עבודה מקומי בו המפתחים של הפרויקט עובדים על כתיבה ועדכון קבצי הפרויקט.

2.4 Check-out

פעולה שיוצרת עותק עבודה מקומי מהמאגר. פעולה זו יכולה להתבצע על גרסה מבוקשת או על הגרסה האחרונה.

2.5 Commit

כתיבת העתק של השינויים שנעשו בעותק העבודה לתוך המאגר.

2.6 שינוי Change

מייצג שינוי מסוים בפריט (מסמך כלשהו) השייך למאגר המנוהל בבקרת תצורה.

2.7 Merge

מיזוג שני סטים של שינויים, שנעשו בקובץ או במספר קבצים, לתוך גרסה אחת.

2.8 Conflict

קונפליקט קורה כאשר שני שינויים סותרים נעשו לאותו פריט (מסמך, קובץ תוכנה וכו'). הפתרון נעשה בצורה ידנית: המשתמשים צריכים להחליט מהו השינוי הנכון.

ייתכנו שמות מעט שונים למושגים אלה בכלי בקרת תצורה שונים.

בקורס נעבוד עם Git ועם GitHub.

Git היא מערכת בקרת גרסאות מבוססת.

היא מאפשרת מעקב אחר שינויים בכל סט של קבצים. מטרותיה כוללות מהירות, שלמות נתונים ותמיכה בזרימות עבודה מבוססות ולא ליניאריות.

GitHub הוא שירות של ניהול גרסאות ושירות אחסון, מבוסס רשת, עבור מיזמי פיתוח תוכנה, שבהם משתמשים במערכת Git לניהול תצורה. GitHub מספק שירות זה בתשלום למאגרים פרטיים ושירות חינמי למיזמי קוד פתוח. במאי 2011 הוכר GitHub כשירות אחסון הקוד הפופולרי ביותר למיזמי קוד פתוח.

התחברות ל-Classroom

יש לעקוב אחרי השלבים הבאים:

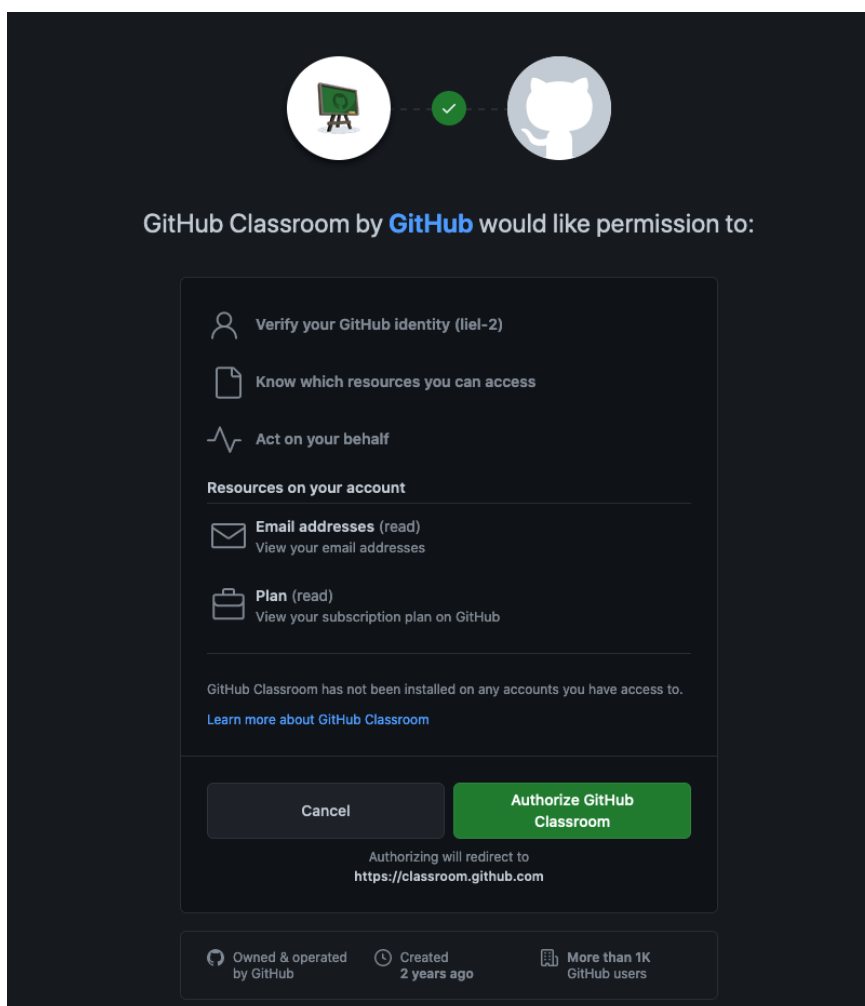
1. ניצור חשבון ב-GitHub.com

2. נכנס ללינק הבא:

[Github Classroom](#)

זהו לינק של המשימה שלנו למעבדה הזאת.

נקבל דף חדש בדומה לזה. נקליק על Authorize GitHub Classroom.



3. חשוב!

רק אחד מהצוות יצור קבוצה חדשה על ידי הקלדת שם הקבוצה (מה שבא לכם, אין הגבלה) ולחיצה על הכפתור Create Team. שאר חברי הקבוצה המגישה מתבקשים להצטרף לקבוצה קיימת על ידי בחירת הקבוצה מהרשימה ולחיצה על כפתור Join. הכוונה היא שלכל צוות המגיש מעבדה תהיה רק קבוצה אחת.
כך נראה הדף:



SWEng-Tutorial-winter2022

Accept the group assignment —

hellogit-swwinter2022

Before you can accept this assignment, you must create or join a team. Be sure to select the correct team as you won't be able to change this later.

Join an existing team

Shir's demo group 1 Join

student

רשימת קבוצות

יצירת קבוצה חדשה

OR Create a new team

Create a new team

+ Create team

4. כעת אנו אמורים להגיע לדף הדומה לזה:



SWEng-Tutorial-winter2022


Accept the assignment —

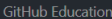





hellogit-swwinter2022


Once you accept this assignment, you will be granted access to the `hellogit-swwinter2022-shirsneh` repository in the `SWEng-Tutorial` organization on GitHub.

Accept this assignment

נלחץ על Accept this assignment. יופיע העמוד הבא:






You accepted the assignment, **hellogit-swwinter2022**. We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Note: You may receive an email invitation to join [SWEng-Tutorial](#) on your behalf. No further action is necessary.




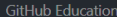





Join the GitHub Student Developer Pack


Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

ולאחר כמה שניות/דקות הוא ישתנה למסך הבא (אם לא – יש ללחוץ על refresh):





You're ready to go —

Shir's demo group


You accepted the assignment, **hellogit-swwinter2022**.

Your team's assignment repository has been created:

<https://github.com/SWEng-Tutorial/hellogit-swwinter2022-shir-s-demo-group>

We've configured the repository associated with this assignment (update).

Note: You may receive an email invitation to join [SWEng-Tutorial](#) on your behalf. No further action is necessary.



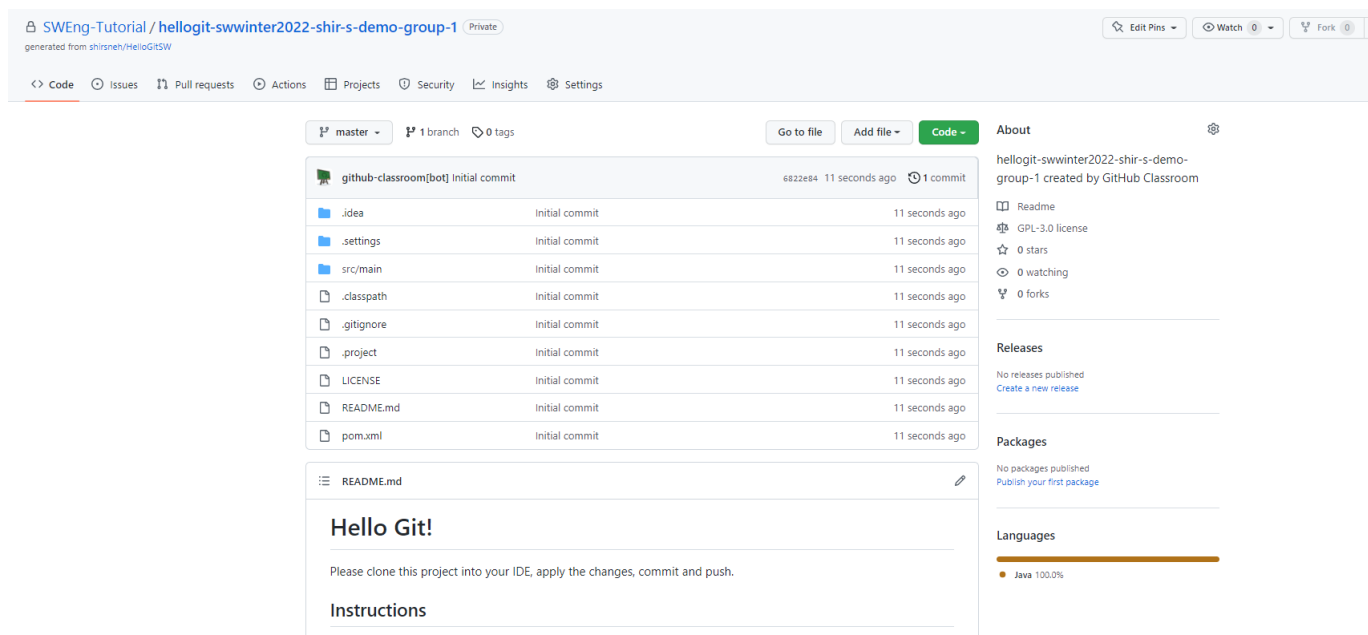
Join the GitHub Student Developer Pack

Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

כאן אפשר לראות את הלינק של הפרויקט שלנו ב-GitHub, וכן לינק שני שיוביל לעמוד הקורס בגיטהאב.

5. נקליק על הלינק הראשון שמופיע. נקבל דף דומה לזה:



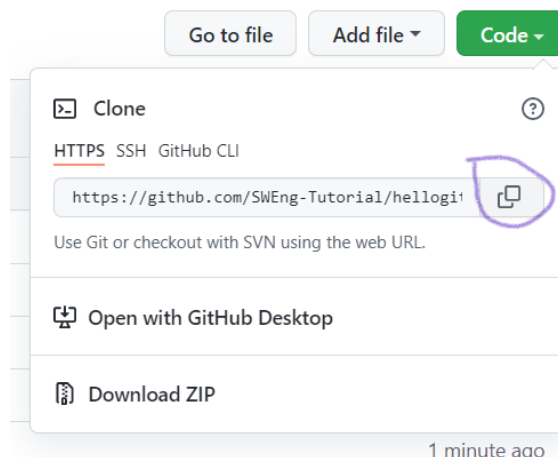
6. לרשותכם פרויקט חדש ב-GitHub. הוא מבוסס על קוד JavaFX בסיסי – חלון עם הודעה ושני תפריטים. מומלץ לא לסגור את הדף, כיוון שנצטרך לקחת את הכתובת של המאגר משם בקרוב.

בסיום השלב הזה כל חברי הצוות אמורים להיות בעלי חשבון ב-GitHub, חברי הקבוצה (Team) ב-ClassRoom ובעלי פרויקט התחלתי.

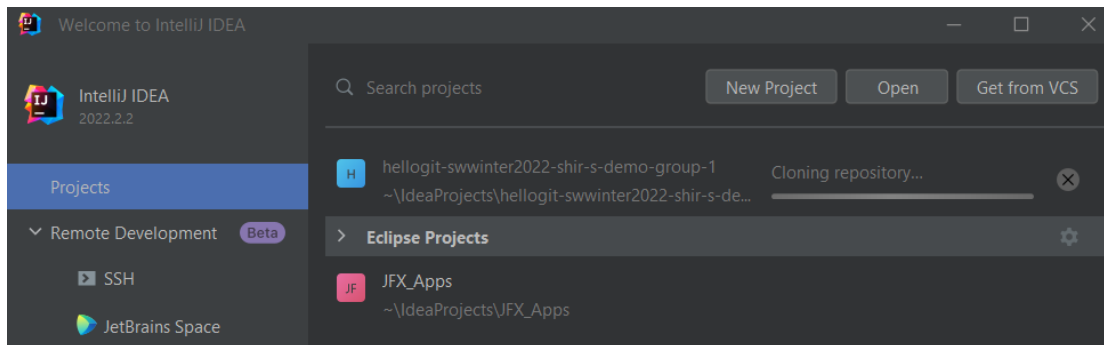
GitHub in IntelliJ

כעת נעבור על כמה שלבים על מנת להגדיר את IntelliJ לעבודה עם הפרויקט שכרגע יצרנו.

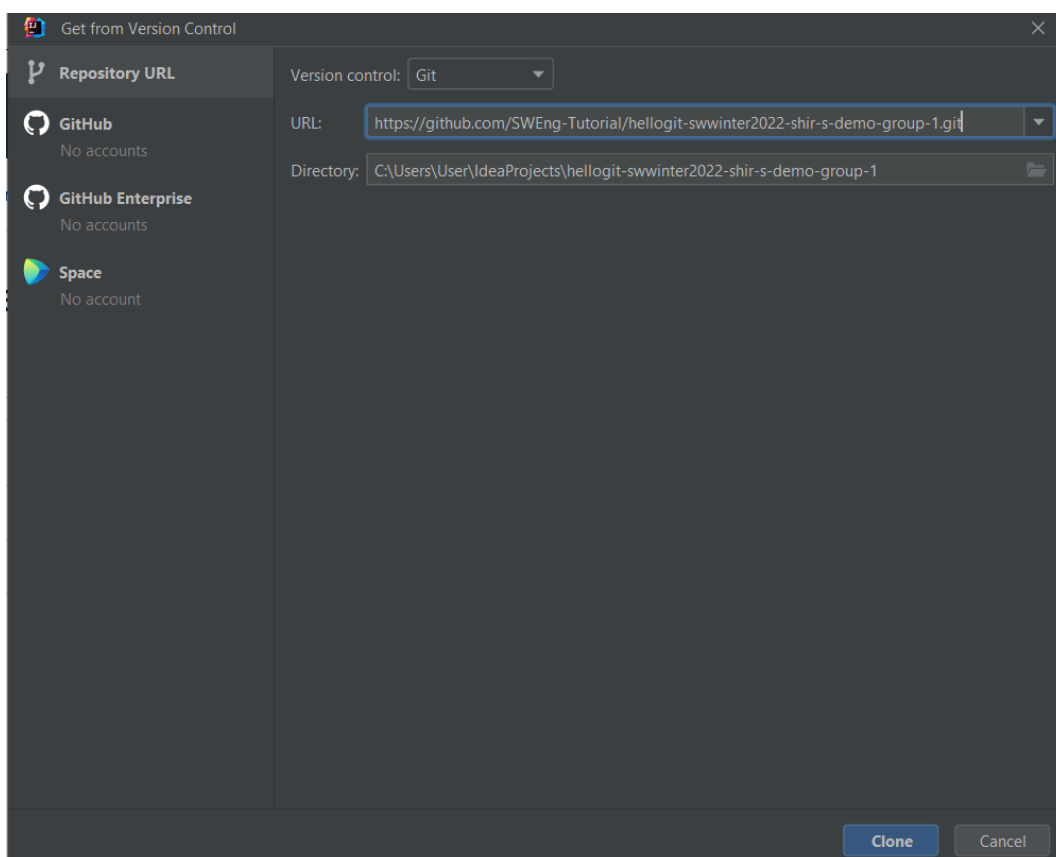
1. ב-GitHub, בעמוד של ה-repository, נלחץ על Code (הכפתור הירוק). נבחר באפשרות HTTPS או SSH, בהתאם למה שנחל לנו (בתור התחלה מומלץ HTTPS ולאחר מכן SSH):



2. כרגע אנחנו מעוניינים להוריד את הפרויקט בשביל לעבוד עליו במחשב מקומי. (יש גם אופציה לערוך את הקוד ישר באתר). בחלון הראשי, נלחץ על Get from VCS



3. בחלון שנפתח נדביק את ה-URL שקיבלנו בשלב 1. נוכל לשנות את התיקייה שאליה הפרויקט ירד, אם נרצה בכך.

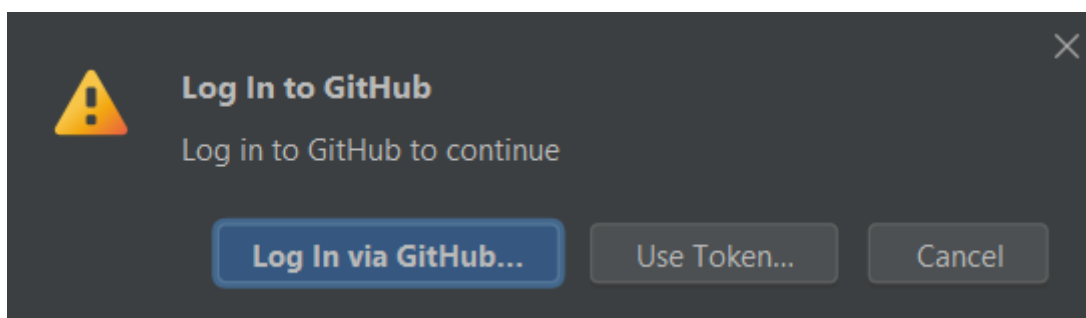


הערה: בשלב זה (וממלץ לכל אורך הקורס) נתעלם מהאופציה להתחבר ישירות ל-

.GitHub

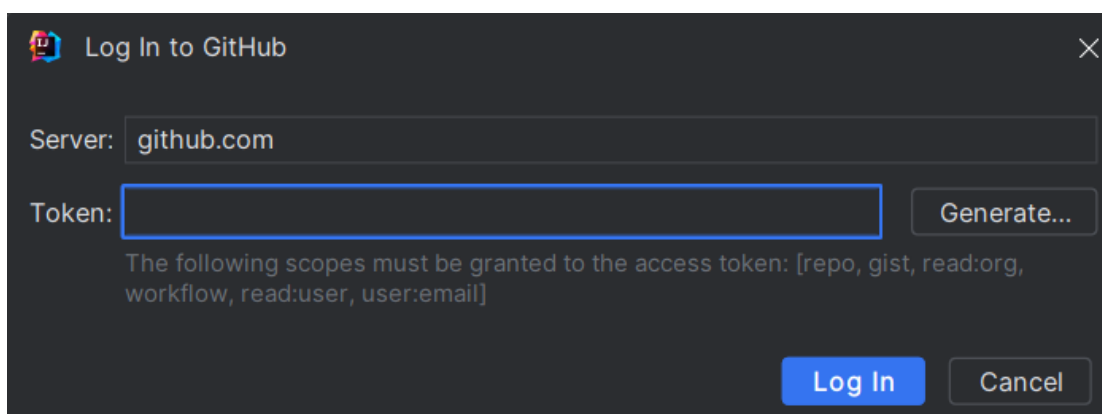
4. נלחץ על Clone.

5. אם בחרנו באפשרות של HTTPS, סביבת העבודה תבקש מאיתנו להיכנס.

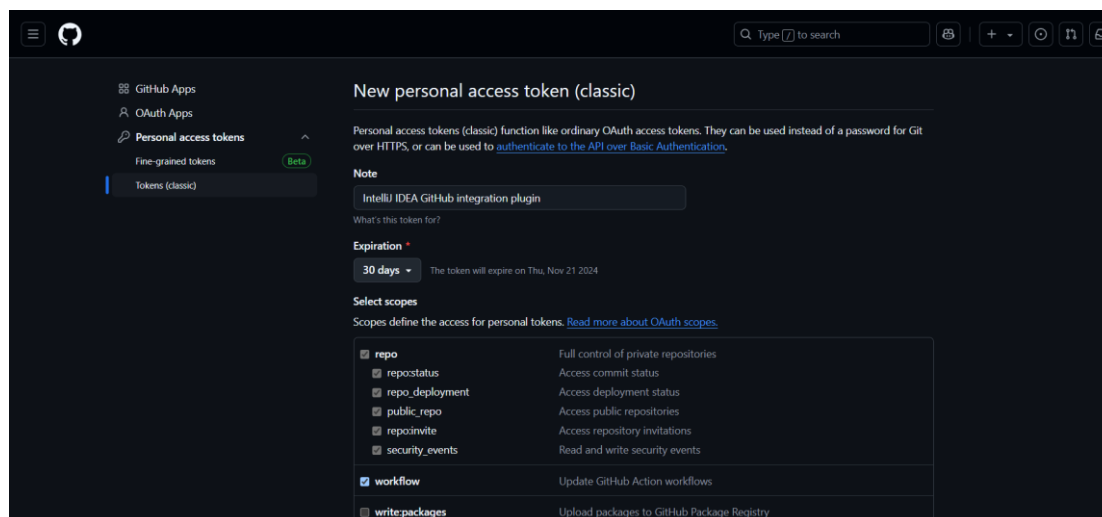


נבחר באופציה Use Token... .

ואז נלחץ על Generate



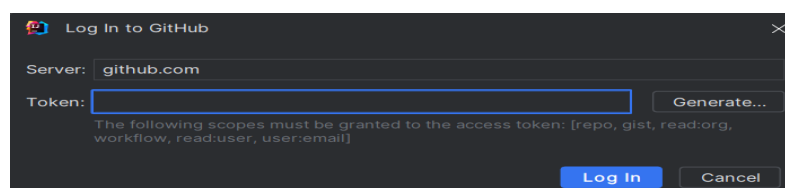
בחלון הדפדפן שייפתח



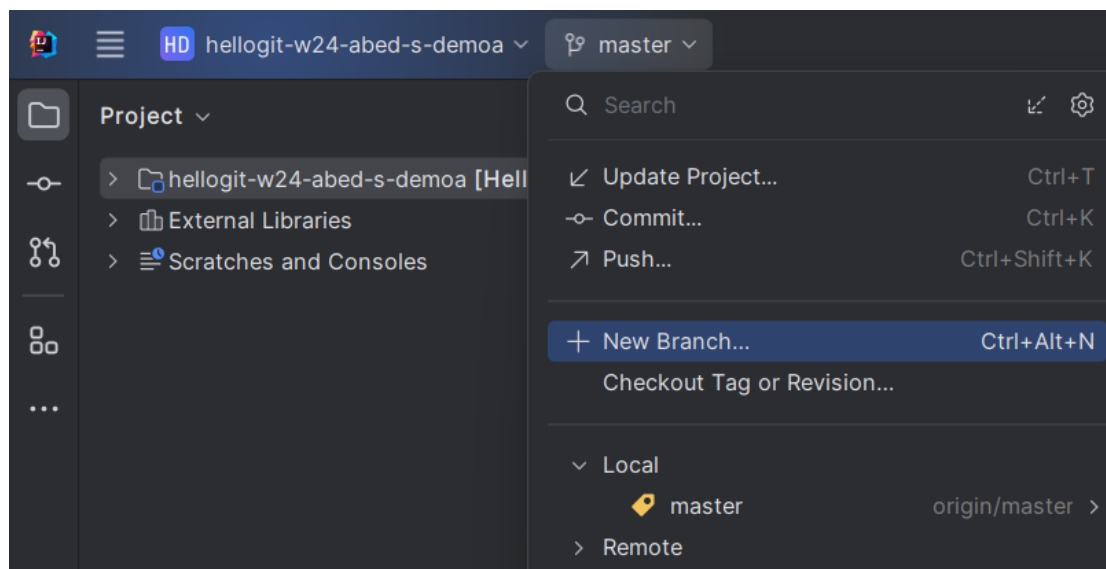
לשנות רק את note ואז לרדת עד הסוף וללחוץ על Generate token שבירוק

ואז נפתח חלון חדש שיש בו את הtoken שמצד שמאל יש לו V להעתיק ושם בתוך token

ואז ללחוץ על login

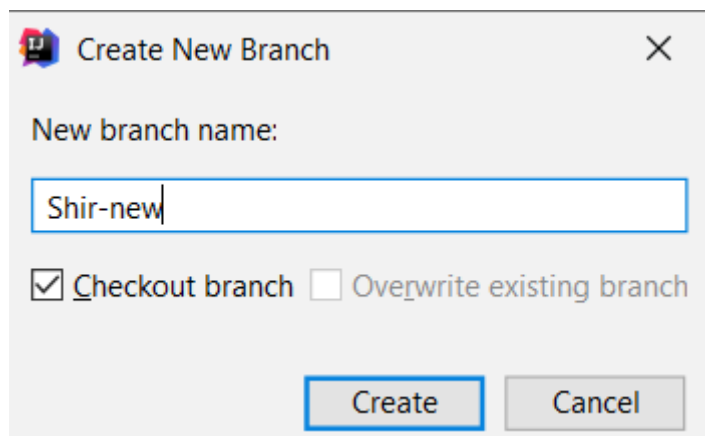


6. סביבת העבודה תשאל אותנו האם לייבא את הפרויקט כפרויקט Eclipse או כפרויקט Maven. נבחר לייבא כפרויקט Maven ונלחץ על OK. כעת ייפתח חלון הפרויקט. קרא בעיון את קובץ ה-README – ההנחיות שם מאוד חשובות (אם לא קרה אז לדלג).
7. ניצור Branch חדש ע"י לחיצה על שם ה-branch המוכי בשורת הסטטוס (למעלה – בצד שמאל):

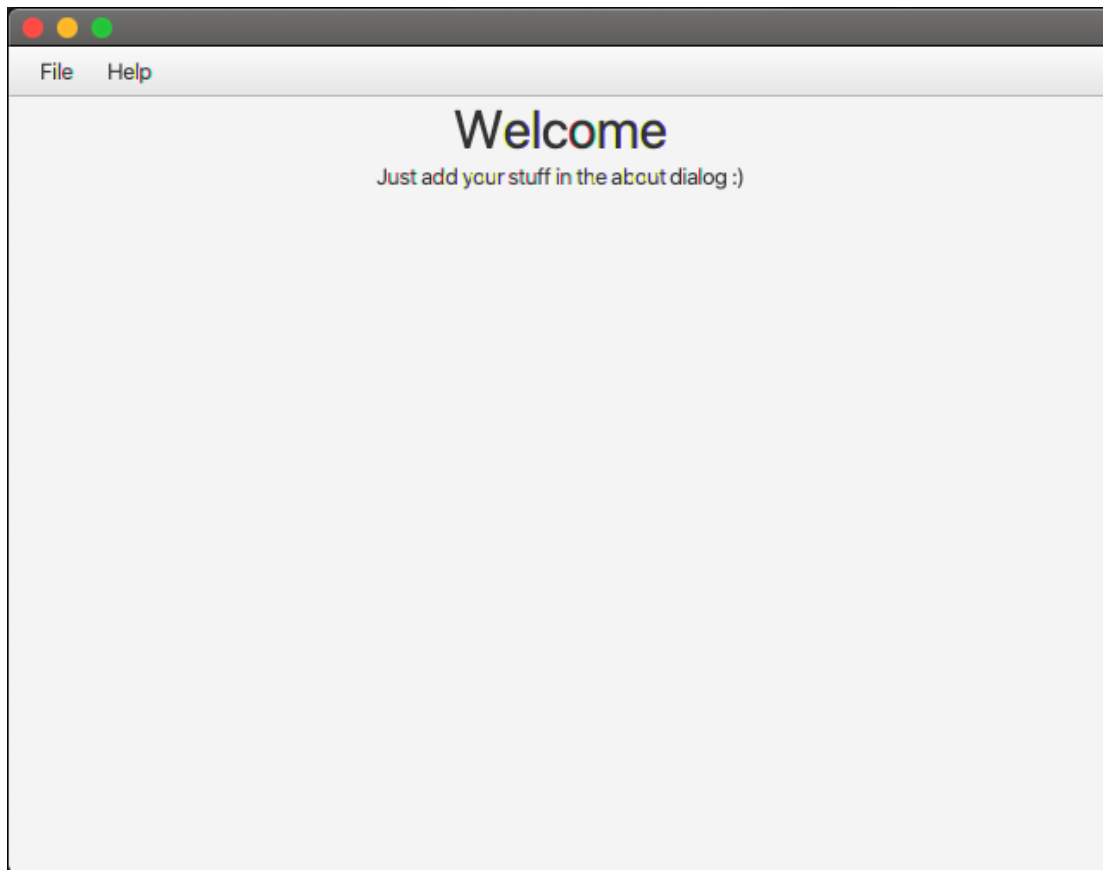


בתפריט שייפתח, נבחר ב-branch new:

8. נבחר שם ל-Branch, למשל:

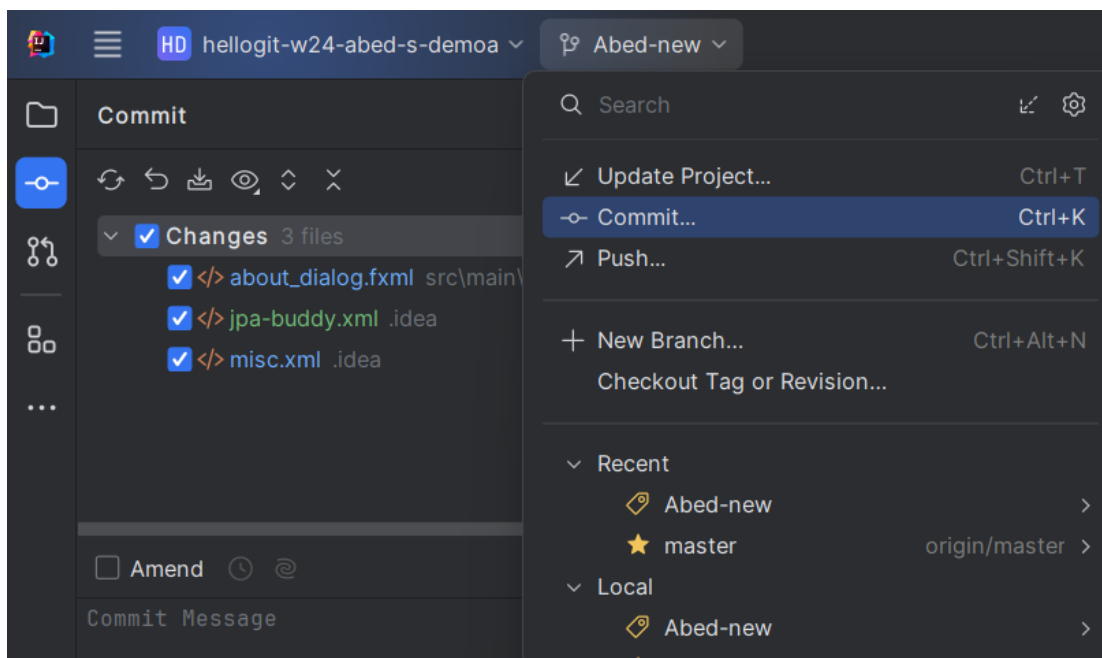


- ונלחץ על Create. השם שבחרנו אמור להופיע בשורת הסטטוס - זה אומר שכרגע אנחנו עובדים על ה-Branch שיצרנו. אם לא, תוכלו להחליף בלחיצה על אותו מקום ובחירה בשם ה-branch שעליו אתם רוצים לעבוד.
9. נוסיף קונפיגורציית הרצה על מנת שנוכל להריץ את הפרויקט – זה נעשה כמו במעבדה הקודמת.
10. נריץ את הפרויקט – ייפתח המסך הבא:

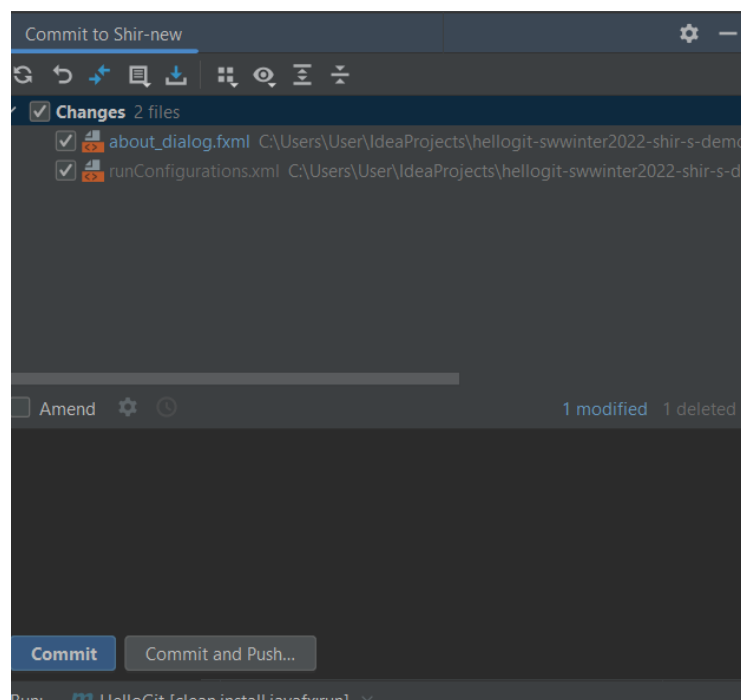


אין צורך לשנות בו דבר. המסך שעליכם לשנות נפתח בלחיצה על Help->About ונמצא בקובץ `about_dialog.fxml` ב-`resources`.
 עכשיו נוכל לעשות את השינויים בקוד ולהוסיף את השם שלנו. שימו לב שלא מספיק לעשות זאת בהערה אלא צריך להוסיף אלמנט GUI מתאים – למשל `Label`.
 לאחר שסיימנו לערוך את הקובץ עלינו להעלות את השינויים חזרה ל-GitHub ולעדכן את שאר חברי הקבוצה על כך שבצענו שינויים.

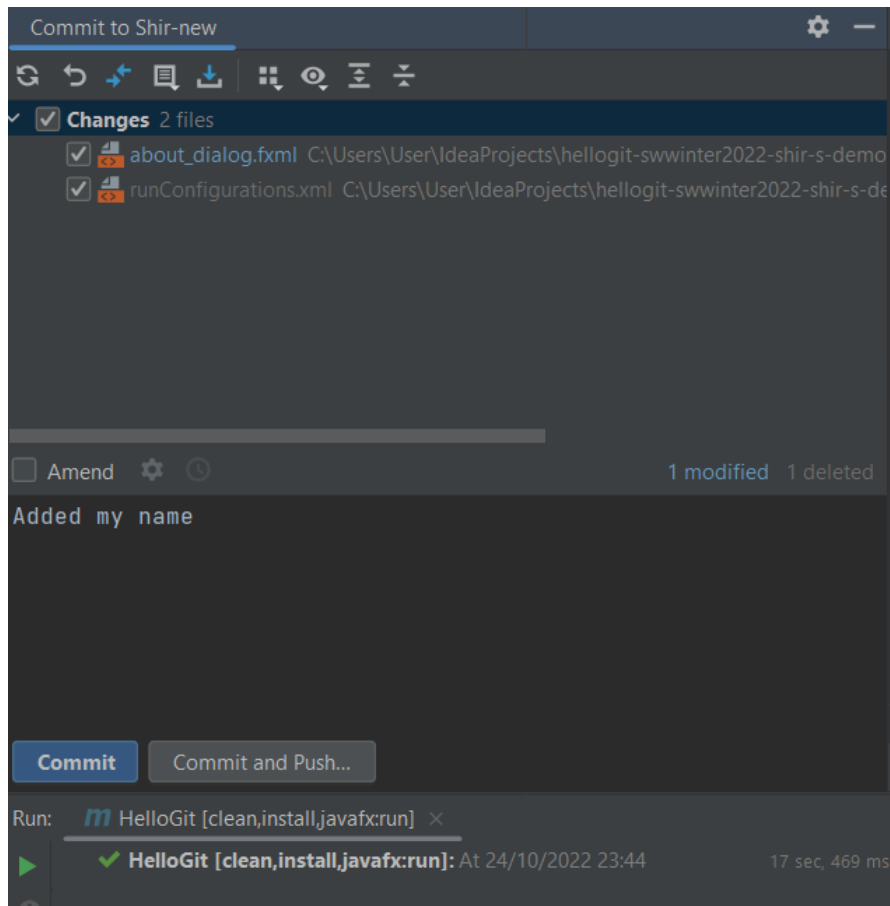
11. נבחר את commit :



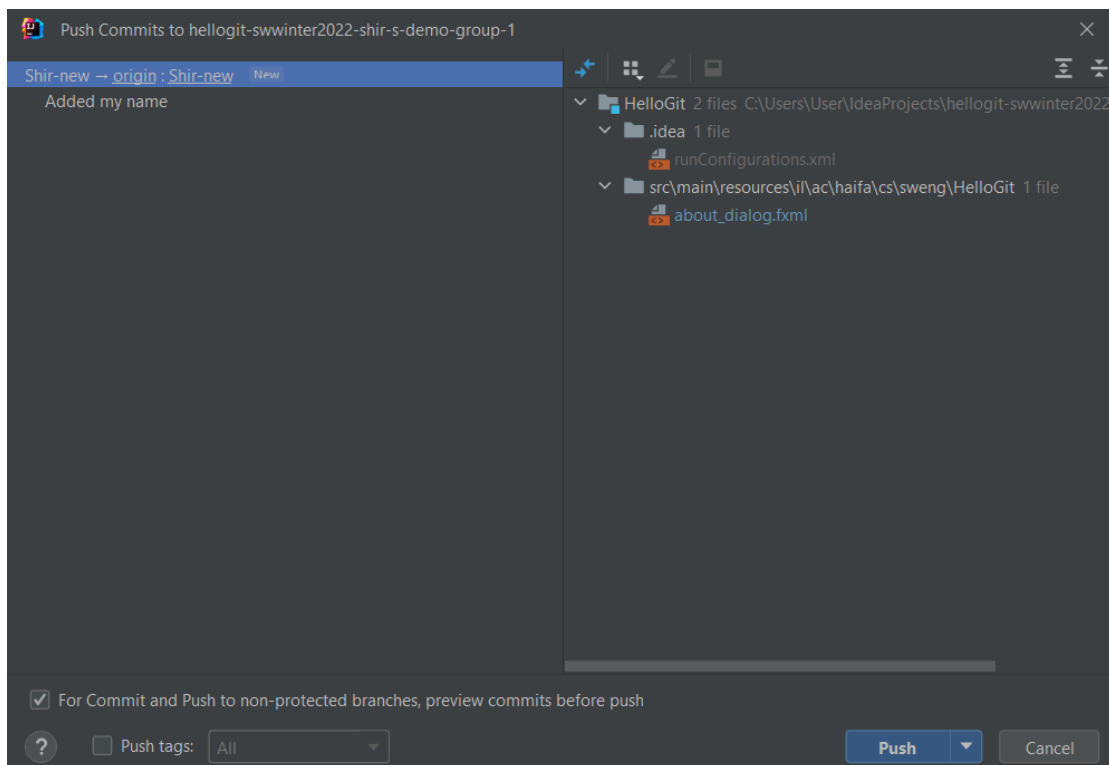
12. ייפתח החלון הבא:



בחלק העליון ניתן להסתכל ברשימת השינויים וכן לסמן כאלה המיועדים ל-staging – בעצם, כאלה שיהיו ב-commit הקרוב. בחלק התחתון, לאחר שסיימנו לסמן את הקבצים, נכתוב תיאור קצר של ה-commit שלנו (בשדה Commit Message).



- כעת נוכל לעשות Commit מקומי (כפתור Commit) או לעשות Commit מקומי וגם להעלות (לדחוף) אותו ל-Github. נלחץ במקרה הזה על Commit and Push... על מנת שנוכל להעלות מיד את הקוד למאגר שלנו.
13. בהנחה שלחצנו על Commit and Push (אם לא – תמיד ניתן ללחוץ בתפריט של IntelliJ - איפה שהשם של ה branch יש גם שם push), ייפתח לנו המסך הבא:



נלחץ על Push. השינויים יועלו למאגר ב-GitHub.

אז עד עכשיו הספקנו ליצור חשבון ב-GitHub, להוריד (או בשפה של Git – clone) את הפרויקט למחשב, ליצור branch חדש, לערוך שינויים לקובץ ב-branch החדש ו"לדחוף" את השינויים בחזרה ל-GitHub.

כרגע נעבור חזרה לדפדפן, לדף של הפרויקט, ונעשה ריענון לדף. אתם אמורים לראות משהו כזה:

בשביל זה אנחנו רוצים להשתמש ב-GitHub, וזה למה אנחנו רוצים לעשות את השינויים ב-Branch שלנו. ככה לכל חברי הקבוצה יש גרסת master עובדת, וכל אחד יכול לעשות שינוי רק אצלו. בסוף, כשחבר קבוצה סיים את העבודה, הוא שולח את השינויים שלו לענן (GitHub) ומבקש מחברי הקבוצה Pull Request. חברי הקבוצה יכולים לעבור על הקוד, לראות שהשינוי הוא הגיוני ולצרף את השינוי ל-master branch, ממנו בהמשך כל האחד ימשוך את השינוי ל-branch שלו. כבר עכשיו אני יכול לומר שכנראה מתישהו תסתבכו ותשמרו את הדברים לא טוב או לא נכון, אבל אין מה לדאוג - בשביל זה בדיוק יש לנו את הכלי הזה וכמעט תמיד הכול ניתן לשחזור אחרי כמה חיפושים (שלכם) בגוגל.

חזרה לשלבים:

14. נקליק על Compare & pull request

Pull request הוא סוג של merge שדורש אישור, וכמו כל פעולת commit בגיט גם כאן אפשר להוסיף הערה. למטה בדף אפשר לראות בדיוק איזה קבצים שונים ואיזה שינויים ערכנו. במינוס וצבע אדום מסומנות שורות ששינוי או מחקנו. בירוק עם פלוס שורות חדשות או שינויים שנכנסו.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base: master ← compare: Shir-new ✓ Able to merge. These branches can be automatically merged.

Added my name

Write

Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviews

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Use [closing keywords](#) in the description to automatically close issues

Helpful resources

[GitHub Community Guidelines](#)

1 commit 2 files changed 1 contributor

שימו לב לחלק שמעל הכותרת – כאן ניתן להחליט מאיזה branch לפתוח את הבקשה (במקרה הזה: liel) ולאיזה branch אנחנו מעוניינים להכניס את השינויים (בדרך כלל main או master).

15. נלחץ על Create pull request

16. בדף החדש נקבל תפריט עם פרטים נוספים לגבי ה-Pull Request ונוכל להסכים על השינויים ע"י לחיצה על הכפתור Merge pull request.

Added my name #1

shirsneh wants to merge 1 commit into `master` from `shir-new`

Conversation 0 Commits 1 Checks 0 Files changed 2

shirsneh commented now

No description provided.

Added my name 2c58417

Add more commits by pushing to the `shir-new` branch on `SWEng-Tutorial/hellojit-swwinter2022-shir-s-demo-group-1`.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Write Preview H B I ≡ < > 🔗 ≡ ≡ ≡ @ 🗑 ↶

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

שימו לב כי הכפתור Merge pull request ירוק. זה אומר שהשינוי שערכנו לא מתנגש עם שינויים אחרים שחברי צוות אחרים יכלו לעשות בזמן שעבדנו. לעומת זאת, למשל, בעבודה על פרויקטים גדולים, שני מפתחים יכולים להכניס שינוי לאותה פונקציה, ואז צריך להחליט איזה שינוי נשאר או איך משלבים ביניהם. אנחנו בטוחים שהתנגשות כזאת תקרה גם לכם ומשארים לכם איך למצוא את הפתרון.

17. נקליק על Merge Pull Request ואז Confirm merge.

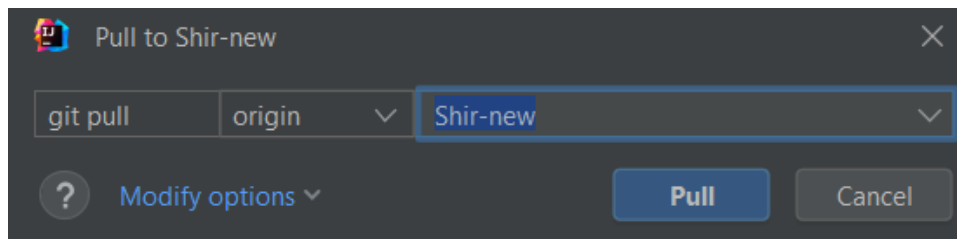
כעת השינוי שעשינו נכנס גם ל-branch master.

18. עכשיו נעבור למחשב של השותף\פה השני\ה ונמשיך משם. (גם השותף הראשון צריך לעשות pull ל-master ולעדכן את הקוד).

כרגע יש לנו פרויקט ב-GitHub ובו שני branches אחד של השותף\ה הראשון והשני master.

19. נרצה שהשותף/ה השני גם יהיה מעודכן עם גרסת הקוד האחרונה.

אם אין עדיין את הפרויקט ב-IntelliJ של השותף השני אז פשוט צריך לחזור על כל השלבים עד כה. אם משכנו את הפרויקט גם במחשב השני, צריך למשוך (pull – בעצם להוריד) את השינויים. ניתן לעשות זאת בעזרת לחיצה בתפריט על Git->Pull. יפתח החלון הבא:



במקום origin נוכל לבחור מאיזה remote (כתובת של מאגר) למשוך (שימושי במיוחד כאשר עושים fork – כלומר, משכפלים מאגר קיים, אבל רוצים בשלב כלשהו למשוך את השינויים מהמאגר המקורי, שאת הכתובת שלו צריך להוסיף) למשוך, ונוכל גם לבחור את ה-branch המתאים (במקרה הזה נצטרך למשוך מה-master). נלחץ על Pull ונמתין עד שנקבל התראה האומרת כי הפרויקט עודכן.

20. אנחנו כבר יודעים איך ליצור branch חדש ולתת לו שם, לכן אם עדיין אין לשותף branch נעשה את זה. ה-branch שיווצר יהיה שכפול של המצב של ה-branch הקודם באותו מצב, לכן מומלץ לבצע את זה ישירות לאחר משיכת השינויים מה-master.

נזכיר כי לאחר שלב 19 במחשב השני יש לנו פרויקט שהוא העתק של הפרויקט במחשב הראשון, שהוא זהה לפרויקט ב-GitHub.

21. כרגע ניצור branch גם לשותף השני, במחשב שלו (נחזור אחרי השלבים הרלוונטיים במסמך)

22. נוודא שאנחנו עובדים על ה-branch הנכון – שם ה-branch שעובדים עליו מופיע בשורת הסטטוס בצד ימין.

23. נוסף את השם של השותף השני.
24. נחזור על פעולת ה-commit (כולל הוספה ל-staging) ונקליק על Commit and Push.
25. כעת נחזור על השלבים 14-17 ובכך נשלים את ה-pull request.
- כרגע אם הכול הלך כסדרו המצב שלנו אמור להיות כזה:
- ל-master ו-branch של השותף השני יש את הקוד הכי מעודכן. הקוד הזה נמצא גם במחשב של השותף השני וגם ב-GitHub. עכשיו אנחנו רוצים לחזור לשותף הראשון ולעדכן גם אותו לגרסה הכי מעודכנת. ואת זה נעשה בעזרת pull.
- נראה שסיימנו.
- אז מה עשינו?
- משכנו פרויקט מוכן מהאינטרנט, הסטודנט הראשון עשה שינוי בקוד שלו, העלה את השינוי בחזרה לאינטרנט ולבסוף ביצע Pull request על מנת לעשות merge ל-master בצורה **מבוקרת**.
- הסטודנט השני משך את השינויים, הוסיף שינויים משלו וחזר על הפעולה של pull request.
- הסטודנט הראשון משך את השינויים ועדכן אותם ב-branch שלו.
- זהו סדר עבודה פשוטני יחסית של עבודה עם Version Control כמו Git למשל. אתם לא חייבים להשתמש ב-GUI של IntelliJ המתואר במעבדה. אתם רשאים להשתמש בכל כלי אחר שנחמ לכם או שאתם מכירים מלפני. יש לא מעט אופציות. כמובן שהכי הרבה כוח יש לעבודה דרך הטרימינל או command line, אך ליום-יום של מפתח תוכנה מה שראינו אמור להספיק.

חזרה קצרה על הפקודות:

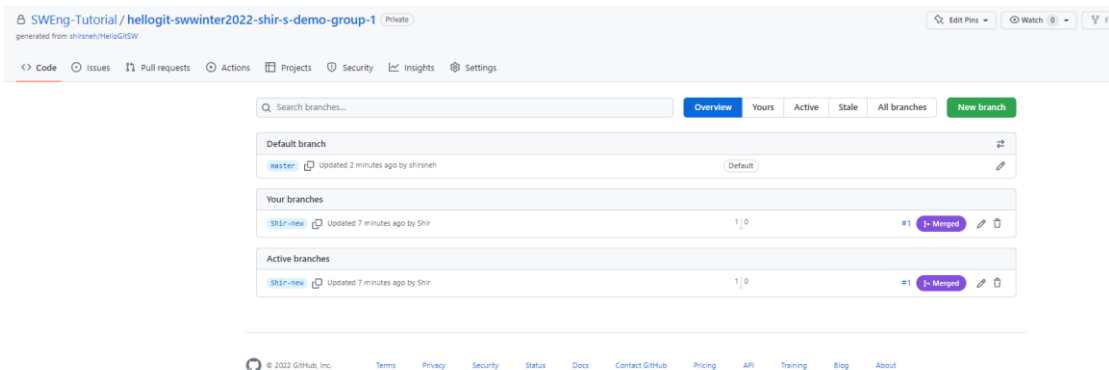
- בסימון הקבצים ולחיצה על הכפתור Commit and push הרצנו שלוש פקודות שהן:
 1. Add – מוסיף קבצים ל-Staging Area
 2. Commit – מוסיף הערה
 3. Push – מעלה את הקבצים ל-remote repository במקרה שלנו זה GitHub אך יש לא מעט אחרים: אפשר גם להקים שרת גיט ביתי, או פרטי בענן.
- השתמשנו ב-Pull בשביל למשוך את השינויים מ-remote repository ל-local. פעולה זו גם מבצעת (לרוב) merge, כלומר מיזוג השינויים ל-branch שאנחנו נמצאים בו כרגע (אם שונה מה-branch שממנו מושכים את הקוד).
- וכמובן Pull request בשביל להכניס שינויים מה-branch שלנו ל-master. בגדול אין מניע לעשות push לתוך master, אבל אנחנו עושים הכנה לקראת העבודה שלכם על הפרויקט ואתם לפחות 4 אנשים בקבוצה, לכן עדיף לשמור על סדר מסוים. זה עניין שלכם האם לאמץ את ההצעה.

לגיט יש עוד לא מעט אפשרויות מעבר לכך, אך מה שהצגנו זו התחלה טובה מאוד. צוות הקורס לא לוקח על עצמו אחריות על הניהול של הפרויקטים שלכם. בבקשה **תמיד** תשמרו גיבוי נוסף. כמו כן אנו לא נתמוך בעבודה עם Git מעבר למעבדה הזאת. אין כיום חברה שלא משתמשת

בגרסה כזו או אחרת של Version Control ולכן יש כמות אינסופית של מידע באינטרנט; זה כלי שימש אתכם שנים רבות בעבודה.

הוראות הגשה:

- להוסיף קובץ בשם report.md ל repository עם השמות שלכם ות.ז ותמונה שמראה את מה שהוספתם ב about.
- נא לא להוסיף/לערוך שום דבר באופן ידני (כמו להשתמש ב add file או דברים מסוג זה).
- היכנסו לדף של הפרויקט ב-GitHub, לחצו על branches ותעשו צילום מסך שדומה לזה:



- עליכם לדאוג שצילום המסך יהיה ברור וכולל את שם המאגר – הבדיקה כוללת כניסה לפרויקט ומעבר עליו.
- את צילום המסך יש להגיש במודל כמו במעבדות הקודמות – שם קובץ ההגשה (צילום המסך) הוא מספרי תעודות הזהות של המגישים מופרדים ב-"_". למשל: 13216546_154646.jpg
-

חלק ב': תרגול בדיקות קבלה

נניח כי עלינו לתכנן מערכת מידע עבור חנות למזון לחתולים. נתון לנו המפרט הטקסטואלי של ה- Use Case הבא:

UC-1	ביצוע הזמנה
שחקנים ויעדים	לקוח: לשלוח לחנות את ההזמנה הנוכחית
בי"ע ואינטרסים	בעלי החנות: ביצוע הזמנה קל ומהיר ככל האפשר
Pre-conditions	<ul style="list-style-type: none">· ללקוח קיים חשבון במערכת· הלקוח מחובר למערכת· קיים לפחות פריט אחד בהזמנה
Post-conditions	<ul style="list-style-type: none">· הלקוח שלח את הזמנתו לחנות· סל הקניות ריק· המערכת במסך הקטלוג
Trigger	הלקוח לוחץ על כפתור "בצע הזמנה" בסל הקניות
MSS	<ol style="list-style-type: none">1. הלקוח מזין את הכתובת למשלוח ואת פרטי התשלום.2. המערכת מאמתת את הפרטים שנתן הלקוח.3. המערכת מציגה ללקוח את פרטי ההזמנה הסופיים.4. הלקוח בוחר "אישור הזמנה".5. המערכת שומרת את פרטי ההזמנה.6. המערכת מנקה את סל הקניות של המשתמש.7. המערכת מעבירה את המשתמש בחזרה למסך הקטלוג.
הסתעפות א'	<p>חלופה מצעד 4 של MSS: הלקוח מעוניין לשנות את פרטי ההזמנה לאחר שהוצגו.</p> <ol style="list-style-type: none">1א4. הלקוח בוחר "חזרה לסל הקניות".2א4. המערכת חוזרת למסך סל הקניות.3א4. הלקוח משנה את סל הקניות כרצונו.4א4. הלקוח לוחץ על "עדכן הזמנה".5א4. התרחיש נמשך.
הסתעפות ב'	<p>חלופה מצעדים 1,3 או 4 של MSS: הלקוח מעוניין להפסיק את תהליך ההזמנה.</p> <ol style="list-style-type: none">1א4/2/1. הלקוח בוחר "ביטול".2א4/2/1. המערכת חוזרת למסך הקטלוג.
הסתעפות ג'	<p>חריגה בצעד 2 של MSS: הפרטים שהזין המשתמש אינם תקינים.</p> <ol style="list-style-type: none">1ב2. המערכת מודיעה למשתמש על אי תקינות הפרטים וכן על השדות הספציפיים שיש לתקן.2ב2. הלקוח לוחץ על "תיקון פרטים".3ב2. חזרה לשלב 1, עם הפרטים שהוזנו מקודם בתוך השדות והדגשת השדות הבעייתיים.

עליכם להכין טבלת מקרי בדיקה עבור ה-UC המתואר לעיל.

שימו לב: זהו תרגיל לכיתה ולא להגשה.

ע ב ו ד ה נ ע י מ ה !