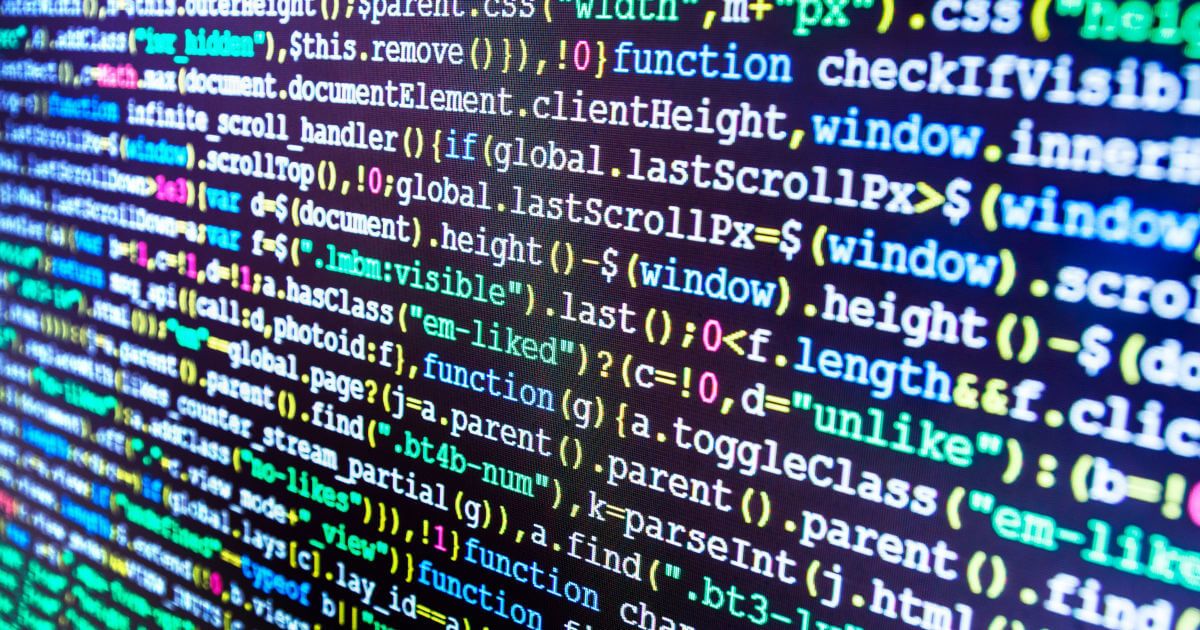
**מבוא לתכנות**

כפי שקראתם במסמכים הקודמים, פיתוח מערכת טכנולוגית מהווה אתגר אמיתי המצריך שיתוף פעולה של מספר גורמים.

במסמך זה אנחנו נדון בתפקיד **התוכניתן** בפיתוח תוכנה**.**

**קוד** – אוסף הוראות ביצוע הניתנות למחשב.

אוסף פקודות מסוימות הניתנות למחשב המשרתות מטרה אחת נקרא **אלגוריתם.**

**תוכנה/מערכת/פלטפורמה** כלשהי מורכבת מאוסף של אלגוריתמים שונים ומממשק למשתמש (יכול להיות ויזואלי ויכול להיות לא) שנועדו לפתירת בעיה מסוימת.

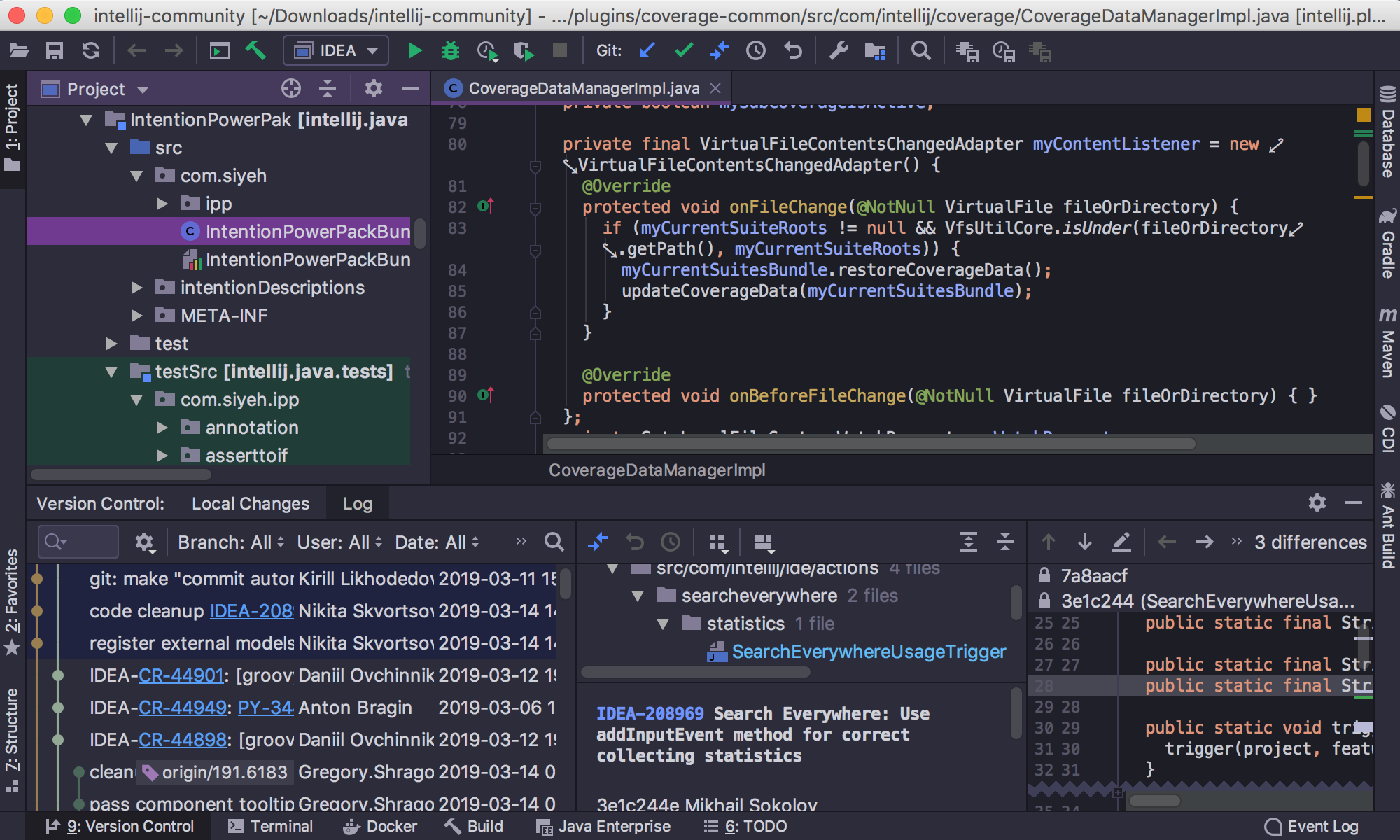
הקוד שאנחנו המפתחים כותבים, יכול להיכתב בשלל שפות תכנות שונות – כל שפה בנויה בצורה שונה שלה חסרונות ויתרונות שונים.

איך כותבים קוד?

קוד בדר"כ כותבים באמצעות סביבת עבודה – תוכנה המספקת פלטפורמה לכתיבת קוד.

למשל בשפת C ו C++ הסביבת עבודה הנפוצה היא Visual Studio, ואילו בשפת Java יש סביבות אחרות נפוצות כגון Intellij ו-Eclipse.

דוגמא לנראות סביבת העבודה intellij שאיתה נעבוד בJAVA:



ע"י שימוש חכם בשפות תכנות וכתיבת קוד נכון, ניתן ליצור אלגוריתמים המבצעים פעולות מורכבות . נעשה זאת באמצעות שימוש במשתנים, תאי זיכרון, קלט/פלט של המערכת, ופעולות מתמטיות ולוגיות – דברים שאותם נלמד בהמשך!

עקרונות חשובים בכתיבת קוד:

כתיבת קוד ש"עובד" לא מספיק.  
במערכות של היום הדרישות גבוהות מאוד, מבחינת אבטחה, ביצועים, חווית משתמש וכו'.

נדרש מהמפתחים לכתוב קוד בצורה "טובה" על פי הקווים המנחים הבאים:

* קריאות: היות ומספר מפתחים שונים עובדים על אותו הפרויקט, כלומר על על קוד משותף, יש לכל אחד צורך בלהכיר ולהבין את הקוד של האחר, לכן יש לשמור על סטנדרטי פיתוח ועל קוד קריא - שלשאר הגורמים יהיה נוח להבין אותו ולפתח אותו.
* מימוש: יש צורך לכתוב קוד בצורה נכונה לפתירת בעיה מסוימת. כלומר, על הקוד לבצע בדיוק את המוטל עליו לעשות. למשל, יש להימנע מכתיבה כפולה של קוד המבצע את אותה הפעולה, על מנת לחסוך במשאבים – בשפות תכנות רבות יש פתרונות רבים שהשפה מציעה על מנת לפתור בעיה זו.
* Best Practice: יש להשתמש בשפת התכנות לפי ההגדרות המומלצות – על מנת לשמור על סטנדרט אחיד ושימוש אופטימלי בכלים של השפה.
* תחזוקה: כל קוד שנכנס למערכת יכול להישאר בה לנצח, כל עוד לא משנים אותו – ומנגד, יכול להשתנות כל הזמן. לכן, חשוב להשקיע בתחזוקה של הקוד, לפתח אותו בצורה המאפשרת לשנות אותו בצורה מהירה ולשפר אותו לבניית יכולות נוספות – גם אם אותו קטע הקוד יישאר זמן רב באותה הצורה. דרישה זו כוללת בתוכה המון נושאים (בדיקות, תיעוד וכו') עליהם נלמד במהלך הקורס.

על כן, אחריותו של המפתח היא לא רק לכתוב אלגוריתם ופתרון לבעיה, עליו לכתוב את האלגוריתם בצורה נכונה וחכמה לעתיד.

**יש עוד הרבה עקרונות וטכניקות, לא נציין פה את כולם.**

**ריצה יבשה / Debug**

אחד הדברים החשובים ביותר בתור מפתחים הוא מעקב יסודי על הקוד שלנו. כלומר, בדיקה האם האלגוריתם פועל כפי שתכננו, או חיפוש אחר מקור של בעיה בקוד.

נוכל לעשות זאת במספר דרכים:

**שימוש בdebug** – דיבאג זו יכולת הקיימת בסביבת העבודה המאפשרת לנו לבצע מעקב אחר תמונת המצב של האלגוריתם בכל קטע נתון - נעקוב אחר סטטוס וערכים של משתנים, לאחר כל פעולת קוד.

1. **טבלת מעקב** – טבלת מעקב היא טבלה בה ניתן לעקוב אחר מצב נתון באלגוריתם. לעומת דיבאג, טבלת המעקב תתבצע בצורה ידנית.

דוגמה לטבלת מעקב : בטבלת מעקב, נדאג לתעד את ערך המשתנים לאחר כל פקודה. לדוגמה, לאחר פקודה 1 המשתנה sum יכיל את הערך 0, ולאחר פקודה 2 הוא יכיל את הערך 3 וכן הלאה. כך נוכל לדעת מה הפלט של התוכנית שלנו (כלומר מהי התוצאה של הקוד שלנו) בכל פקודה.

|  |  |  |  |
| --- | --- | --- | --- |
| **שלב** | **Sum** | **name** | **count** |
| פקודה 1 | 0 | “Yossi” | 0 |
| פקודה 2 | 3 | “Yossi” | 1 |
| פקודה 3 | 3 | “Yossi” | 1 |
| פקודה 4 | 7 | “Yossi” | 2 |
| פקודה 5 | 7 | “Yossi” | 2 |

במסמך זה למדתם מעט על הכלים החשובים ביותר לפיתוח תוכנה. בהמשך נרחיב יותר על כל נושא ונתחיל לכתוב תוכניות של ממש.

בהצלחה!