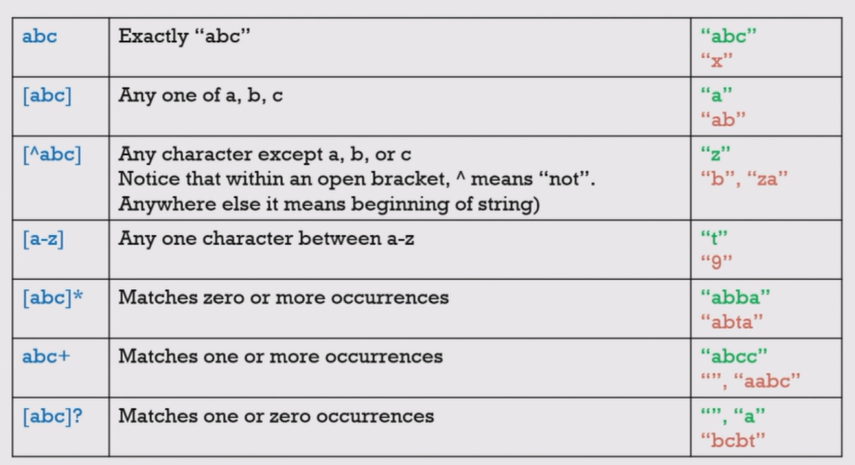
**תרגול 9 OOP**

**Regex – Regular Expressions:**

* ביטויים רגולריים / רג'קס הוא כלי שנועד לבדוק התאמה של מחרוזות לתבניות מסוימות, כלומר האם מחרוזת בנויה מתבנית מסוימת.
* הם עוזרים לחלץ מידע ממחרוזות שבנויות לפי תבניות מסוימות
* אבני בניין בסיסיות:



* 1. בדיוק המחרוזת הכחולה
  2. תו אחד בלבד ממה שיש בסוגריים
  3. הוספת ה^ אומר כל תו בודד חוץ מהתווים שיש בתוך הסוגריים

משמעות נוספת היא התאמה לחלקים מהמחרוזת אז ^ דורש שמשהו יהיה ממש שהתחלה של המחרוזת (כאשר לא מופיע בין סוגריים)

* 1. טווח של תווים, לפי ASCII
  2. עם נוסיף \* זה אומר 0 או יותר מופעים של מה שמופיע לפני (אם יש סוגריים אז כל מה שבסוגריים, אם אין סוגריים אז על התו האחרון שלפני)
  3. + מופע אחד או יותר של מה שלפני (אם יש סוגריים אז כל מה שבסוגריים, אם אין סוגריים אז על התו האחרון שלפני)
  4. ? מופע יחיד או 0 מופעים של מה שלפני (אם יש סוגריים אז כל מה שבסוגריים, אם אין סוגריים אז על התו האחרון שלפני)

שרשור:

* שרשור של רג'קסים – נכתוב אותם אחד אחרי השני בלי דברים בין לבין
* תבניות שנשרשר אחת אחרי השנייה צריכות להופיע אחת אחרי השנייה במחרוזת עצמה
* הגדרת תבנית שחוזרת על עצמה N פעמים נעשה ע"י הוספת {N} אחרי הרג'קס
  + - אם נרצה שיופיע בין M לבין N פעמים נוסיף {M,N}
    - אם נרצה שיופיע לפחות N פעמים נוסיף {,N}
* הוספת | היא עבור או – מאפשר לבחור בין כמה אפשרויות. האופציות יהיו כל מה שנמצא לפניו או כל מה שאחריו (או בין || אם יש כמה)

קיצורים:

* . - נקודה תופסת כל תו למעט תווים של ירידת שורה
* \d – ספרה בין 0 ל-9 - [0-9]
* \D – כל תו שאינו בין 0 ל-9, כלומר [9-0^]
* \s – כל תו רווח, כולל טאב ירידת שורה, כלומר [ \t\n\v\f\r]
* \S – כל תו שאינו רווח – [^\s]
* \w – תווי מילה, כלומר תווים שיכולים להיות חלק ממילה – אותיות קטנות, אותיות גדולות, קו תחתון וספרות – [a-zA-Z\_0-9]
* \W – כל מה שאינו תו שיכול להיות במילה

רג'קסים בJAVA:

* יש לנו ספרייה java.util.regex.
* יש לנו בספרייה Pattern ו-Matcher
* המחלקה Pattern – מקושר לרג'קס עצמו, לתבנית אליה נרצה להתאים

ניצור ע"י Pattern.compile – יקמפל את תבנית הרגקס עם אוטומט מצבים, לוקח זמן

* המחלקה Matcher – משהו שנוצר מתוך התבנית בצירוף מחרוזת שנרצה לבדוק האם מתאימה לתבנית (כלומר התבנית והמחרוזת לבדיקה)

יוצר matcher לחלק ספציפי של הטקסט

* את הmatcher יוצרים מתוך הpattern – תוצר של התבנית
* ניצור אובייקט pattern ואובייקט של matcher ניצור ע"י pattern.matcher
* פונקציות של Matcher:
  1. Matches() – מחזירה אמת אם התבנית מתאימה לכל המחרוזת בשלמותה
  2. lookingAt() – בודק האם ההתחלה של המחרוזת מתאימה לרגקס (האם מצליח למצוא משהו שמתחיל לפי התבנית)
  3. find() – מחזיר אמת אם יש חלק כלשהו במחרוזת שעונה על התבנית
     + שומר מצב פנימי וכל פעם שנקרא לו שוב הוא ימשיך לבדוק מאיפה שעצר
     + ברגע שעבר על כל המחרוזת יחזור להתחלה
  4. Start(), end() – מחזירים את האינדקס הראשון של ההתאמה ואת האינדקס אחד אחרי האחרון של ההתאמה
     + נשתמש בהם רק אחרי שבדקנו אם יש בכלל התאמה
     + הם מתעדכנים לפי ה-find()
* Backslashes:
  1. \b – תופס את ההפרדה בין משהו שהוא word character למשהו שהוא לא, ולהיפך
     + ב-JAVA התו שהמחשב מקבל כאשר אנו לוחצים על backspace
  2. קודם כל תתקבל המשמעות של JAVA (הסימן )
  3. אם נרצה שיתפוס את המשמעות של הרגקס נוסיף [\\b](file:///\\b) כלומר עוד \
* כדי לתפוס \ אחד נצטרך לכתוב \\\\ (4 פעמים)

קבוצות – capturing groups:

* יש לקבוצות שני שימושים עיקריים:
  1. אם נרצה להפעיל כמות על יותר מתו אחד, נעטוף בסוגריים את מה שנרצה
  2. גישה לפי אינדקסציה להתאמות ספציפיות
* כמות הסוגריים שיש תגדיר את כמות הקבוצות
* נוכל לגשת ל-group(n) לפי מספר הקבוצות שיש
* אם יש לנו סוגריים עם + ישר אחריהם אז ההתייחסות לפי קבוצה תהיה לקבוצה האחרונה בלבד (יכולות להיות כמה)
* אפשר לגשת לחלקים שונים במחרוזת
* אם נלחץ matcher.group(1) נקבל אתה קבוצה 1 מתוך המחרוזת
* הבחירה של הקבוצות היא בצורה חמדנית – ימשיך למלא את הקבוצה הראשונה עד שלא יוכל יותר ורק אז יפתח את הקבוצה השנייה

האתר regex101 – עוזר מאוד עבור רגקסים

* גם שאנו על מצב JAVA הוא לא יוסיף את ה\ הנחוצים