: **ThreadPool**

היא שיטה לבצע הפשטה בין כמות התהליכים הנדרשים למימוש האלגוריתם לבין כמות התהליכים שרצים בזמן אמת על המחשב בגלל אילוצי משאבים (לרוב זיכרון, משאבי חישוב ליבות פיסיות וכו').

וכך המתכנת יכול להגדיר כמה תהליכים שצריך בעוד שהמערכת (המחלקה ThreadPool) דואגת שכמות התהליכים שרצים באמת בצורה מקבילית תואמת את הגדרות "משאבי המערכת".

דוגמא פשוטה לכך ניתן לראות בשאלה 2 – כאשר ניתן להגדיר בצורה פשטנית את כמות התהליכים להיות ככמות הליבות הפיסיות שזמינות לביצוע המיון.

–**Synchronized** המילה השמורה synchronized שמופיעה בראש בלוק פקודות מבטיחה שבכל עת רק thread יחיד יוכל להפעיל את הבלוק על אובייקט ספיציפי, במילים פשוטות, כל עוד הבלוק לא מסתיים כל ההפעלות האחרות של אותו בלוק ע"י thread-ים אחרים ימתינו לסיום עבודתו של הבלוק על אובייקט ספיציפי.

**Singleton :**

היא שיטה להגדרה של מחלקה שממנה ניתן "לייצר" אך ורק אובייקט יחיד – שימוש בשיטה זאת נעשה כאשר רוצים להבטיח אובייקט יחיד – כגון אובייקט פרמטרים נוכחי של משתני מערכת. או כל משאב שיש שמייצג אובייקט שממנו יש עותק יחיד ולא נכון לאפשר לייצר ממנו יותר מאובייקט אחד.

מימוש שיטה זאת נעשה בד"כ ע"י כך שהבנאי של המחלקה הוא פרטי, ויש פונקציה סטטית שמחזירה עותק יחיד מהאובייקט – אם קיים מחזירה פשוט את המצביע אליו ואם לא מייצרת עותק סטטי יחיד ומחזירה אותו.

**Iterator :**

מאפשרת מעבר על איברי אובייקט נתון באופן שאינו תלוי במימוש הפרטני של מבנה הנתונים בו משתמש האובייקט. הממשק מעבר סידרתי (כולל בדיקת סיום ומחיקה) על מבנה נתונים ללא תלות במימוש הקונקרטי שלו. ובכך יוצרת אי תלות בין המימוש של מבנה הנתונים והאלגוריתמים שמשתמשים בו.

הפעולות המרכזיות ב git הן הפעולות של כל מערכת בקרת תצורה לרבות:

1. יצירת repository
2. הוספה של קבצים \ תיקיות ל repository וכן מחיקה של קבצים( 3
3. ביצוע עדכון commit
4. יצירת "ענף" branch וכן איחוד בעזרת merge
5. צפייה בכל הcommitים שנעשו באמצעות log
6. חזרה לקומיט ישן באמצעות cheackout

ב github יש יכולות נוספות מעבר ל git בפרט נושאים שקשורים לניהול פרויקט: חלוקת משימות, collaborators , ניהול issues

**JDBC :**

ממשק jdbc הוא חלק מהספריות של הjava ומייצג גישה גנרית לבסיסי נתונים. הAPI שלו מאפשר מספר פעולות מרכזיות בניהן:

1. טעינת driver (שנדרש להיות תואם לבסיס הנתונים שנרצה להשתמש בו).
2. יצירת Connection שלמעשה יוצר ערוץ תקשורת עם בסיס הנתונים דרך הממשק.
3. יצירת פקודה Statement למעשה מהווה פקודת SQL, אותה ניתן להריץ ולקבל תוצאה בדמות תת טבלה – ResultSet אוסף זה ניתן למעבר בצורה סדרתית באמצעות iterator.

היתרון בממשק הוא היכולת של מעבר בין בסיסי נתונים שונים כמעט ללא צורך לעשות שינויים בקוד.

במטלה 4 נעשה שימוש פשטני בבסיס נתונים: לאחר כל משחק תוצאת המשחק מבחינת נקודות וmoves נשלחו לטבלה גלובלית בשרת כלשהו – עדכון זה נעשה למעשה בצורה אוטומטית ולא דרש את התערבות הסטודנטים. בהמשך נדרשנו להוסיף תאפשרות הבאה: לגשת לטבלה הגלובלית דרך ממשק JDBC ולקבל את התוצאות שלנו ולהשוות בין התוצאות שלנו לבין שאר הקבוצה ביחס לתרחישים השונים שנבדקו (כ 11 במספר).

**Serializable :**

הינו ממשק ריק (ללא שיטות) בjava אשר מממשים כאשר רוצים לסמן שאובייקטים ממחלקה נתונה יכולים לעבור תהליך סריאליזציה. סריאליזציה הוא תהליך של תרגום מבנה נתונים או מצב של אובייקטים, לפורמט שניתן לאחסן אותו (לדוגמא בקובץ או להעביר אותו על גבי רשת מחשבים). משמע מיפוי של נתוני האובייקט לאוסף רציף של bytes.

**Json :**

פורמט קל להעברת נתונים משרת לדף אינטרנט. קל להבנה, לא תלוי לשפת תכנות ספציפית ולכן ניתן לשימוש לכל שפות התכנות, ופורמט טקסט בלבד. הוא תומך בעיקר ב 6 סוג משתנים: string , number , Boolean , array , object , null