JAVA – SE (Standard Edition)

מאפיינים עיקריים ויתרונות של Java SE כוללים:

1. **API עשיר** מספקת מגוון רחב של ממשקי תכנות יישומים (APIs) המאפשרים גישה לפונקציות שונות כמו ניהול קבצים, רשת, GUI, ועוד.
2. **תמיכה במולטי-תהליכים**: Java SE תומכת בפיתוח יישומים שיכולים לבצע מספר תהליכים במקביל.
3. **Java Development Kit (JDK)** שמספק את הכלים הנדרשים לפיתוח והרצת תוכנות Java, כולל קומפיילר, כלים לניהול גרסאות, ועוד.

* **Java Compiler (javac)** - כלי לקומפילציה של קוד Java לקוד בינארי (bytecode).
* **Java Runtime Environment (JRE)** - מספק את הסביבה הנדרשת להרצת יישומי Java, כולל את ה-JVM (Java Virtual Machine).
* **ספריות API** - אוסף רחב של ספריות המאפשרות למפתחים להשתמש בפונקציות שונות, כמו עבודה עם קלט/פלט, רשתות, אוספים ועוד.
* **כלים נוספים** - כלים לניהול קוד, כמו javadoc ליצירת תיעוד, jar לאריזת קבצים ועוד.

1. **פלטפורמה עצמאית-: JVM** יישומים שנכתבים ב-Java SE יכולים לפעול על מערכות הפעלה שונות ללא צורך בשינויים בקוד.

העיקרון המרכזי הוא "כתוב פעם, רוץ בכל מקום" (Write Once, Run Anywhere - WORA), כלומר, קוד Java שנכתב במערכת אחת יכול לרוץ על מערכות אחרות כל עוד יש להן JVM מותקן.

Thread

המחלקה Thread ב-Java מייצגת יחידת ביצוע עצמאית, כלומר, היא מאפשרת להריץ קוד במקביל ליחידות ביצוע אחרות. באמצעות המחלקה הזו, ניתן ליצור ולנהל Threads, מה שמאפשר לבצע מספר פעולות במקביל, דבר שיכול לשפר את הביצועים של יישומים, במיוחד כאשר יש משימות שאינן תלויות אחת בשנייה.

**תכונות עיקריות של המחלקה Thread:**

* **ביצוע מקבילי**: מאפשרת להריץ קוד במקביל, מה שמאפשר ליישומים לנצל טוב יותר את המשאבים של המערכת.
* **ניהול משאבים**: מאפשרת להגדיר עדיפויות ל-Threads, כך שניתן לקבוע איזה Thread יקבל יותר משאבים במצבים של תחרות.
* **המתנה וסנכרון**: כוללת מתודות כמו join() ו-sleep(), המאפשרות לשלוט על זמני הביצוע וההמתנה של Threads.

**שימושים נפוצים:**

* ביצוע משימות ארוכות טווח במקביל לממשק משתמש, כך שהממשק נשאר תגובתי.
* טיפול בבקשות בו זמנית בשרתים (כמו שרתים של אפליקציות אינטרנט).
* ביצוע חישובים מקבילים כדי לייעל ביצועי אלגוריתמים.

ה-API של Java עבור המחלקה Thread כולל את המתודות והמאפיינים הבאים:

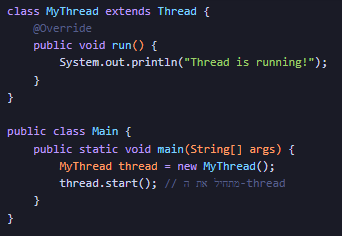
**מאפיינים:**

* **MAX\_PRIORITY**: קבוע המייצג את העדיפות הגבוהה ביותר ל-Thread (ערך 10).
* **MIN\_PRIORITY**: קבוע המייצג את העדיפות הנמוכה ביותר ל-Thread (ערך 1).
* **NORM\_PRIORITY**: קבוע המייצג את העדיפות הנורמלית ל-Thread (ערך 5).

**מתודות עיקריות:**

1. **start() - מתחילה את התהליך ומביאה להרצת המתודה run().**
2. **run() - מכילה את הקוד שיתבצע כאשר התהליך רץ.**
3. **sleep(long millis) - עוצרת את התהליך הנוכחי למשך זמן מסוים (במילישניות).**
4. **join() - מחכה שהתהליך הנוכחי יסיים את הריצה שלו.**
5. **join(long millis)-מחכה שהתהליך הנוכחי יסיים את הריצה שלו או עד שהזמן שצוין יחלוף.**
6. **isAlive() - מחזירה ערך בוליאני שמצביע אם התהליך עדיין רץ.**
7. **setPriority(int priority) - קובעת את העדיפות של התהליך.**
8. **getPriority() - מחזירה את העדיפות של התהליך.**
9. **getName() - מחזירה את השם של התהליך.**
10. **setName(String name) - קובעת את השם של התהליך.**
11. **yield() - מבקש מהמערכת להפסיק את התהליך הנוכחי ולתת לתהליכים אחרים לרוץ.**
12. **currentThread() - מחזירה את התהליך הנוכחי שמבצע את הקוד.**
13. **activeCount() - מחזירה את מספר התהליכים הפעילים באותו זמן.**

**דוגמת קטע קוד של Thread:**



Collections

ב-Java, קיימים מספר סוגי אוספים (Collections) שניתן להשתמש בהם, והם חלק מהחבילה java.util. הנה סקירה של הסוגים העיקריים:

**1. List**

* **ArrayList**: רשימה דינמית, גישה מהירה לאלמנטים לפי אינדקס.
* **LinkedList**: רשימה מקושרת, טובה להוספה והסרה של אלמנטים.

**2. Set**

* **HashSet**: אוסף שאינו שומר על סדר אלמנטים, מבוסס על טבלאות גישה.
* **LinkedHashSet**: אוסף שאינו שומר על סדר אלמנטים, אך שומר על סדר ההוספה.
* **TreeSet**: אוסף שמקיים סדר טבעי של האלמנטים או לפי Comparator שניתן.

**3. Map**

* **HashMap**: מפת מפתח-ערך שאינה שומרת על סדר.
* **LinkedHashMap**: מפת מפתח-ערך ששומרת על סדר ההוספה.
* **TreeMap**: מפת מפתח-ערך ששומרת על סדר טבעי של המפתחות או לפי Comparator שניתן.

**4. Queue**

* **PriorityQueue**: תור שבו האלמנטים ממוינים לפי סדר עדיפויות.
* **LinkedList**: יכול לשמש גם כתור, עם תמיכה בפעולות תור.

**5. Deque**

* **ArrayDeque**: תור דינמי המאפשר גישה לשני הקצוות.