**Multi Programming ב-C#**

**יישום שהוא שילוב של תוכניות עצמאיות.**

**ככל שמערכת היא יותר ויותר גדולה, יש לנו עוד סיבות לעבוד בשיטה כזו.**

**מוטיבציות עיקריות:**

* **צורך לעבור בפתאומיות מנקודה מסוימת בקוד לנקודה אחרת.**
* **מידור והגבלה של רכיבי התוכנית.**
* **מקל על התכנון של יישום.**

**ב-Java יש תמיכה רק ב-Thread-ים בגלל האופי הפורטבילי של השפה.**

**ב-C# ההנחה היא שעובדים ב-Windows לכן יש גם תמיכה בשימוש בתהליכים.**

**Thread-ים**

* **מבחינת המערכת קבוצת Thread-ים הם תהליך אחד**
* **מרחב הכתובות של ה-Thread-ים משותף**
* **העברת מידע אפשרית באמצעות שדות סטטיים (גלובליים) ופוינטרים**
* **כל הקוד של קבטצת thread-ים היא תוכנית אחת, קוד של Thread הוא מתודה (ולא קובץ exe).**

**בעוד שהעברת מידע בין Thread-ים הוא עניין פשוט, לסנכרון ומניעה הדדית בין Thread-ים עדיין צריך לדאוג.**

**מניעה הדדית – Mutual Exclusion – מצב שבו תוכניות עצמאיות ניגשות לשטחי זיכרון משותפים רק כאשר השטחים המשותפים הללו מכילים ערך תקין (ערכים של לחלוטין לפני או לחלוטין אחרי שינוי).**

**פעולות על Thread-ים**

* **כאשר Thread עושה פעולה על Thread אחר הוא יעשה זאת ע"י מתודה רגילה דרך מצביע למופע של ה- Thread האחר. למשל Start Join,.**
* **כאשר Thread רוצה לעשות פעולה על עצמו הוא יקרא למתודה סטטית של Thread.**

**נעילה לקריאה בלבד:**

 **אם מאפשרים נעילה לקריאה בלבד, אין צורך לעצור תוכניות המנסות לנעול לקריאה משאב הנעול לקריאה בלבד.**

 **אם משהו מנסה לנעול לכתיבה, הוא חייב להמתין עד שכל הקוראים יסיימו.**

**כל מי שרוצה לנעול לקריאה, ייעצר עד שהכותב יסיים.**

**תכנות ג'נרי**

**תכנות שמשאיר את סוג הנתון שפועלים עליו פתוח.**

**בכמעט כל שפה, אם רוצים לממש אלגוריתם נניח על float ו-double צריך לשכפל את הקוד.**

**במספר שפות (C++, Ada) אפשר לממש קוד ג'נרי בלתי מוגבל באמצעות תמיכה בסוג מיוחד של מקרו.**

**מקרו – מעין קוד שעובר "עיבוד טקסטאלי לפני קומפילציה".**

**ב-Java וב-C# משום מה החליטו לא לממש תכנות ג'נרי מבוסס מקרו.**

**בגרסאות הראשונות לא היה באופן רשמי תכנות ג'נרי בכלל.**

**מאוחר יותר בשני השפות הכלילו תכנות ג'נרי מוגבל.**

**Casting על פרימיטיבים**

**בהקשר הזה casting הוא בעצם קוד ביצועי.**

**Casting בהקשרים אחרים הוא בעיקר התעלמות מסוג הפוינטרים.**

**Boxing ו-Unboxing**

**ב-C# משתנה object יכול להצביע גם על פרמיטיביים (ב-Java אני אזדקק ל-Wrapper).**

**שימוש:**

**Cast\_stack היא מחסנית במימוש רציף של כל סוגי הנתונים האפשריים.**

**הנתונים אפילו לא חייבים להיות הומוגניים.**

**מה אנחנו רוצים שבאמת יהיה אפשר לאכסן בתוצאה של הקימפול רק איברים מסוג מסוים?**

**על מנת ליצור מחסנית שסוג האיברים שלה נשאר פתוח בקוד אבל נקבע ברגע הקומפילציה צריכים את התמיכה בקוד ג'נרי (החל מ-C# 2.0)**