**בניגוד ל-C שינוי הגדרת משתנה לא מחייבת שינוי של פקודות ה-printf המדפיסות אותו.**

**System.out.printf( "\nadding %.2f to account2 balance\n\n",**

**account1.getBalance() );**

**המרות:**

**שפת Java מתירה המרות בין מספרים כאשר אין אובדן דיוק:**

* **אפשר תמיד להציב סוג משתנה במשתנה גדול יותר**
* **אפשר להציב שלם לתוך ממשי.**

**המרות הכרוכות באובדן מידע או דיוק מחייבות קוד מפורש (casting) המתכנת חייב להיות מודע / לקחת אחריות.**

**ב-Java יש תקן אחיד לגדלים של משתנים פרימיטיביים והן ממומשת באופן אחיד בכל הפלפורמות.**

**בשפת Java כל קוד ביצועי נקרא מתודה.**

**מבדילים בין שתי סוגי מתודות:**

* **מתודות מופע**
* **מתודות סטטיות.**

**מתודות מופע מזוהות עם הקצאה ספציפית (אובייקט) של מחלקה.**

**בדרך כלל פעולה מסוג כלשהוא על המופע.**

**class Aclass**

**{**

**public void m()**

**{ ……**

**}**

**}**

**Aclass x = new Aclass();**

**x.m();**

**מתודת מופע אפשר לקרוא רק אחרי הגדרת משתנה והקצאה, דרך משתנה שמצביע על ההקצאה.**

**מתודה סטטית, מוגדרת ע"י מילת מפתח static נקראת דרך שם המחלקה:**

**class Aclass**

**{**

**public static void sm()**

**{ ……**

**}**

**}**

**הקריאה**

**Aclass.sm();**

**לא צריך להקצות מופע כלשהוא לקרוא למתודה סטטית.**

**מתודות סטטיות ב-Java הן למעשה המקביל של "פונקציות גלובליות" או "פונקציות עזר" בשפות הרגילות.**

**לעומתן, מתודות מופע אמורות להיות תמיד חלק מהמימוש של המחלקה.**

**שדות מידע סטטיים הם המקביל למשתנים גלובליים בשפת Java ושפות דומות.**

**בדרך כלל לא נאפשר גישה ישירה לשדות סטטיים לקוד לקוח. הגישה יהיה דרך מתודות סטטיות. יוצא מן הכלל הינם קבועים (שממילא אי אפשר לשנות).**

**מתודות סטטיות אינן מזוהות עם שום מופע, ולכן אינן יכולות להתייחס ישירות לשדות מופע.**

**class Aclass**

**{**

**public int v;**

**private int v1;**

**static int sv;**

**void m1()**

**{**

**}**

**void m()**

**{**

**v = 10;**

**sv = 30;**

**sm();**

**m1();**

**}**

**public static void sm()**

**{ ……**

**Aclass a1 = new Aclass();**

**a1.v1 = 10;**

**sv = 10;**

**m1();**

**v = 10;**

**}**

**}**

**main()**

**{**

**Aclass.sv =10;**

**Aclass a = new Aclass();**

**a.v = 20;**

**a.m();**

**Aclass.sm();**

**}**

**מחסנית המערכת ומנהל הזיכרון האוטומטי:**

* **ב-Java מוקצים על המחסנית הפרמטרים והמשתנים הלוקליים.**
* **פרימיטיביים מוקצים כמות שהם על המחסנית.**
* **משתנים מורכבים (מחלקות מערכים) יותר מוקצים רק מצביעים.**
* **אם מתודה משתמשת במבנה נתונים מורכב היא חייבת להקצות אותו דינמית במפורש ע"י new.**
* **ה-JVM מכיל מנהל זיכרון העוקב כמה מצביעים פעילים מצביעים על כל שטח שהוקצא דינמית.**
* **כאשר מתודה חוזרת משתחררים המשתנים הלוקליים במחסנית, כולל המצבעים לשטחים דינמיים.**
* **אם מנהל הזיכרון מגלה ששטח דינמי לא מוצבע יותר בעקבות כך ע"י אף מצביע פעיל, הוא משחרר את השטח אוטומטית.**
* **התוכניתן לא חייב לשחרר שטחים דינמיים בשפה הזו. אין מקביל ל-free או delete של שפות אחרות.**
* **לפעמים משחררים שטח דינמי ע"י השמה נוסך**

**ptr = null;**

* **המחיר הוא זמן CPU.**