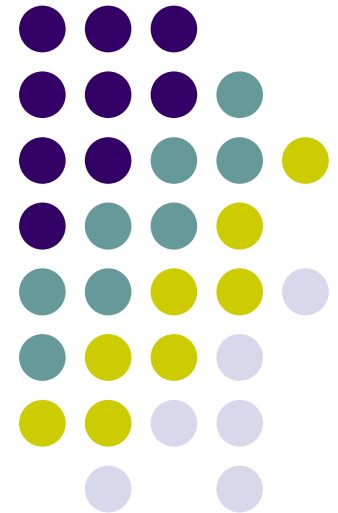


# תכנות מונחה עצמים

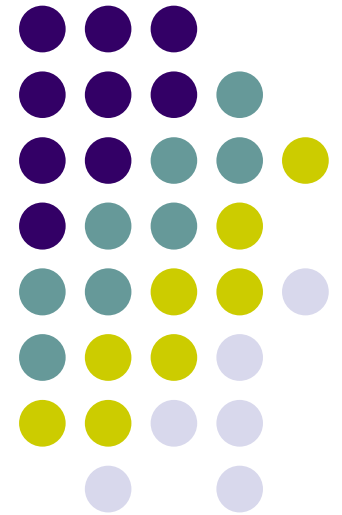
משתני מחלקה, עצמים מוכלים  
ועצמים מוצבעים

אוהד ברזילי  
אוניברסיטת תל אביב



# משתני מחלקה

Static Members





# משתני מחלקה (static members)

- משתנים סטטיים מוגדרים בתוך מחלקה ואולם הם אינם מוקצים בשטח המוקצה לכל עצם של אותה מחלקה
- הם מוקצים פעם אחת בלבד (ממש כמו משתנים גלובליים) ואולם הגישה אליהם היא לפי הרשאת הגישה שניתנה להם במחלקה
- משתנים סטטיים מתפקדים ממש כאילו הוגדרו ב namespace ששמו כשם המחלקה
- על משתנים סטטיים יש להכריז בקטע קוד גלובלי, אז גם ניתן לאתחל אותם (אחרת יאותחלו ל-0)



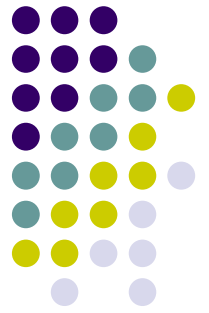
# CStudent

```
class CStudent
{
    const int m_grade;        // The student's grade
    static int nofStudents;
    static int maxGrade;

public:
    CStudent(int grade = maxGrade) : m_grade(grade)
    {
        if (grade > maxGrade)
            maxGrade = grade;
        nofStudents++;
    }

    // Errors
    // CStudent(int grade = m_grade) {}
    // CStudent(int grade) { m_grade = grade; }
    // CStudent(int grade): maxGrade(grade) ...
}
```

# CStudent



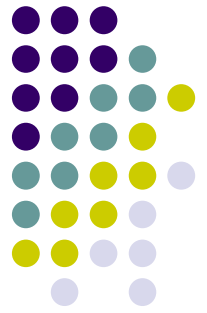
```
CStudent(const CStudent& std): m_grade(std.m_grade)
{
    nofStudents++;
}

~CStudent()
{
    maxGrade = 0;
    nofStudents--;
}

void Print() const
{
    cout << "Grade = " << m_grade << endl;
}

};
```

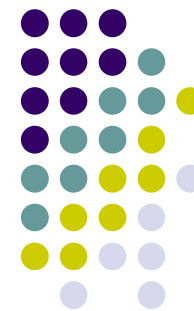
# CStudent



```
int CStudent::nofStudents = 0;
int CStudent::maxGrade = 0;

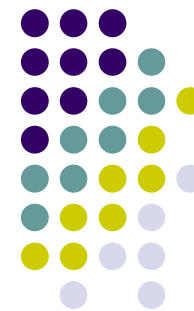
void main()
{
    CStudent Rafi;
    CStudent Moshe(97);
    CStudent Avi(89);
    CStudent::maxGrade++; // ERROR. Private variable
    CStudent CopiedFromMoshe = Moshe;

    CStudent* theNewStudent;
    theNewStudent = new CStudent (12);
}
```



## מתודות סטטיות

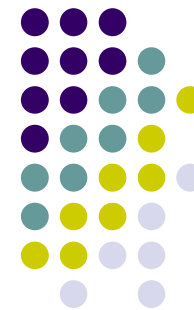
- מתודות השייכות למחלקה מסוימת ואולם לא פועלות על עצם מסוים של המחלקה
- כאשר מתודות פונות לחברי מחלקה שאינם סטטיים (משתנים או מתודות) עליהן לציין את שם העצם המפורש (כי אין להן `this`)
- המתודות גלובליות ואולם הגישה אליהם היא לפי הרשאת הגישה שניתנה להן במחלקה
- מתודות סטטיות מתפקדות ממש כאילו הוגדרו ב `namespace` ששמו כשם המחלקה



# מתודות סטטיות

- השימוש בחברים סטטיים מייתר למעשה את השימוש ב namespace כמודול עבור טיפוס נתונים בודד
- ב'עולם האמיתי' משמש namespace לתאור מודול המורכב מעשרות מחלקות ויותר (לדוגמא: namespace std או namespace microsoft::word)
- בשפת Java השימוש בפונקציות גלובליות אסור מבחינה תחבירית. כל מחלקה המעוניינת לשמש נקודת פתיחה לביצוע יישום כלשהו מגדירה פונקציה סטטית בשם main() ובה היא מבצעת את כל פעולות האתחול הנדרשות

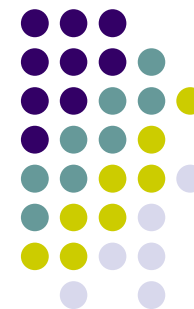




# CStudent

```
class CStudent
{
    public:
        const int m_grade; // The student's grade
        static int nofStudents;
    private:
        static int maxGrade;
        static CStudent* bestStudent;

    public:
        CStudent(int grade = maxGrade):m_grade(grade)
        {
            if(grade > maxGrade)
            {
                maxGrade = grade;
                bestStudent = this;
            }
            nofStudents++;
        }
}
```



# CStudent

```
CStudent(const CStudent& std) // as before...
```

```
~CStudent()  
{  
    if (bestStudent == this)  
    {  
        maxGrade = 0;  
        bestStudent = NULL;  
    }  
    nofStudents--;  
}
```

```
void Print() const // as before...
```

```
static int GetMaxGrade() { return maxGrade; }  
};
```



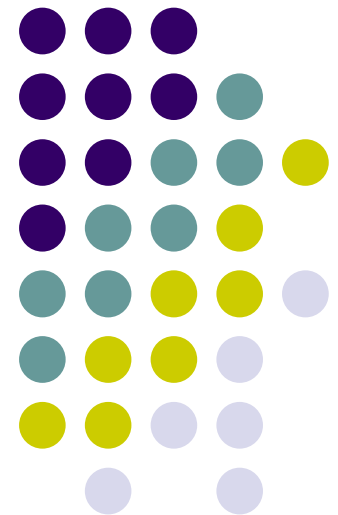
# CStudent

```
int CStudent::nofStudents = 0;
int CStudent::maxGrade = 0;
CStudent* CStudent::bestStudent = NULL;

void main()
{
    cout<<"We now have"<<CStudent::nofStudents<<endl;
    CStudent Rafi;
    CStudent Moshe(97);
    CStudent Avi(89);

    cout<<Rafi.nofStudents<<endl;
    cout<<CStudent::nofStudents<<endl;
    // cout<<"The max grade is:"<<Rafi.maxGrade<<endl;
    cout<<"Max grade"<<CStudent::GetMaxGrade()<<endl;
}
```

# עצמים מוכללים



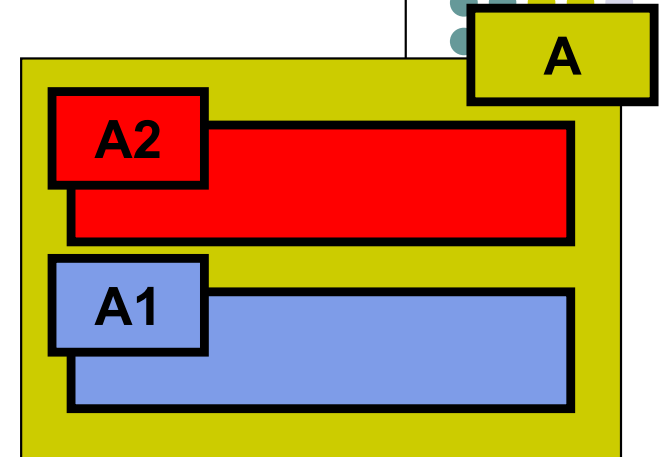
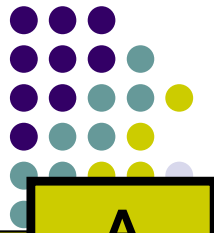


# עצמים מוכלים

```
class A1
{
public:
    A1(){ cout << "A1 Constructor runnig..." << endl; }
    A1(A1& a1){ cout<<"A1 Copy Constructor runnig..."<< endl;}
    ~A1(){ cout << "A1 Destructor runnig..." << endl; }
};
```

```
class A2
{
public:
    A2(){ cout << "A2 Constructor runnig..." << endl; }
    A2(A2& a2){ cout<<"A2 Copy Constructor runnig..."<< endl;}
    ~A2(){ cout << "A2 Destructor runnig..." << endl; }
};
```

# עצמים מוכלים

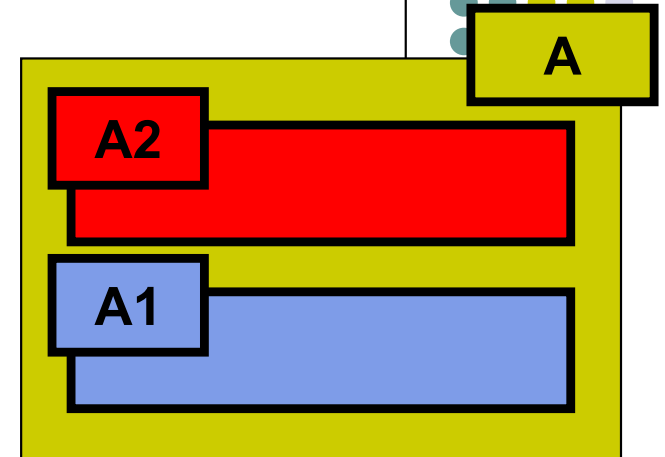
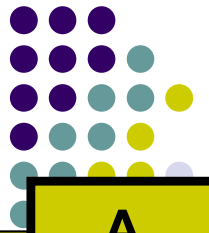


```
class A
{
    A1 a1;
    A2 a2;
public:
    A(){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
};

int main()
{
    A a;
    return 0;
}
```

מה מדפיס הקוד הבא ?

# עצמים מוכלים

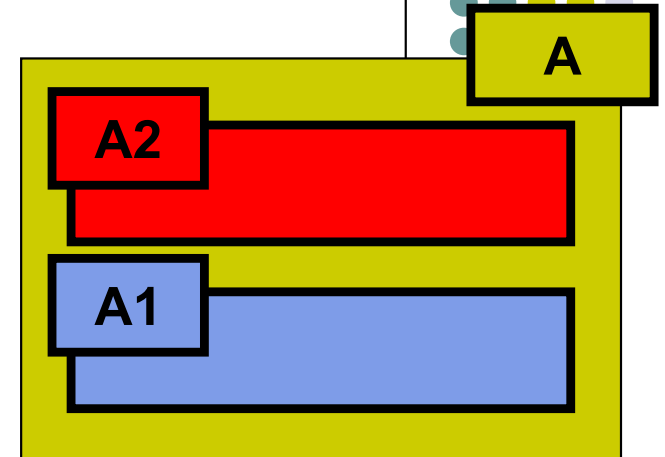
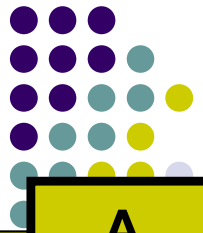


```
class A2
{
    A1 a1;
    A2 a2;
public:
    A(){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
    void f(A& a) { cout << "f() is running..." << endl; }
};

int main()
{
    A a;
    a.f(a);
    return 0;
}
```

ומה מדפיס הקוד הבא ?

# עצמים מוכלים



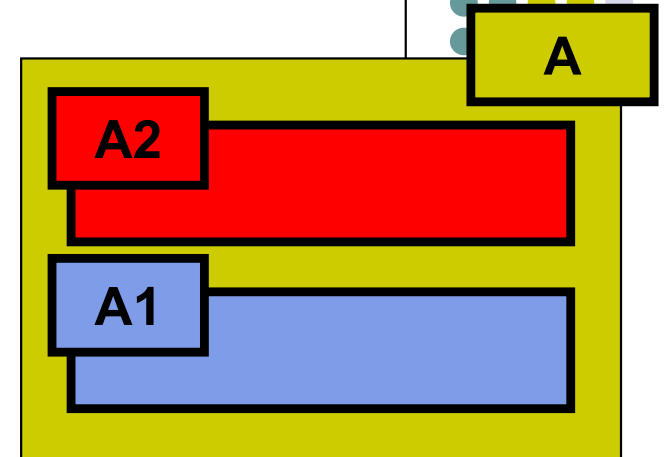
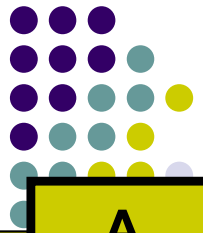
```
class A2
{
    A1 a1;
    A2 a2;
public:
    A(){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
    void f(A a) { cout << "f() is running..." << endl; }
};

int main()
{
    A a;
    a.f(a);
    return 0;
}
```

ועכשיו?



# עצמים מוכללים



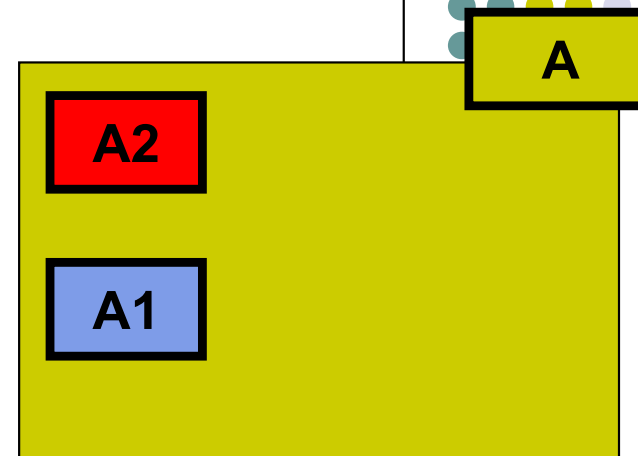
```
class A2
{
    A1 a1;
    A2 a2;
public:
    A(){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
    A f(A a) { cout << "f() is running..." << endl; return a; }
};

int main()
{
    A a;
    a.f(a);
    return 0;
}
```

ועכשיו מה?



# עצמים מוצבעים



```
class A2
{
    A1 *a1;
    A2 *a2;
public:
    A(){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
    A f(A a) { cout << "f() is running..." << endl; return a; }
};

int main()
{
    A a;
    a.f(a);
    return 0;
}
```

מה מדפיס הקוד הבא ?



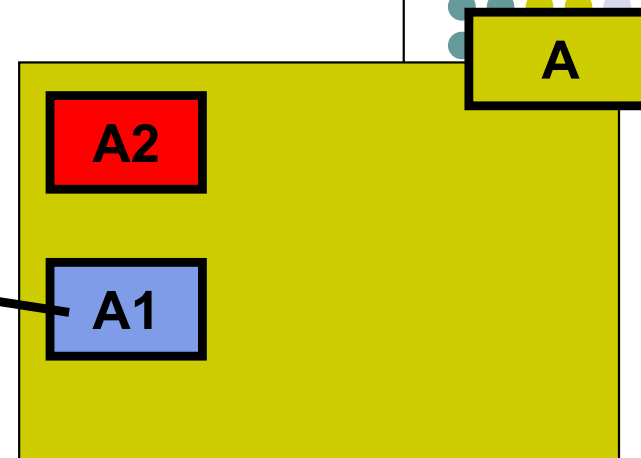
# עצמים מוצבעים

```

class A2
{
    A1 *a1;
    A2 *a2;
public:
    A() : a1(new A1){ cout << "A Constructor runnig..." << endl; }
    A(A& a){ cout<<"A Copy Constructor runnig..."<< endl;}
    ~A(){ cout << "A Destructor runnig..." << endl; }
    A f(A a) { cout << "f() is running..." << endl; return a; }
};

int main()
{
    A a;
    a.f(a);
    return 0;
}

```



ומה מדפיס הקוד הבא ?

---

**ויש עוד בתרגיל הבית...**

