יחידות 7-8 ירושה ופולימורפיזם

```
public class Box
{
  protected double _length;
  protected double _height;
  protected double _width;

  public Box(double len,double wid, double hei)
  {
     _length=len;
     _width=wid;
     _height=hei;
   }
   public double calculateVolume()
   {
      return _length*_width*_height;
   }
}

public class WoodenBox extends Box
{

  public WoodenBox(double len,double wid, double hei)
  {
      super(len,wid,hei);//קריאה לבנאי של האבא

   }
   public double calculateVolume()//overriding
   {
       return (_width-2)*(_length-2)*(_height-2);
    }

}
```

```
public class CerealBox extends Box
{
  private String _cerealName;

  public CerealBox(double len,double wid, double hei,String name)
  {
      super(len,wid,hei);//קריאה לבנאי של האבא
      _cerealName=name;
    }

    public double calculateCerealVolume()
    {
        final double PERCENT_FOR_CEREAL=0.9;
        return calculateVolume()*PERCENT_FOR_CEREAL;// שימוש
בשיטה של האבא

       }
     public int calculateFontSize()
     {
        return (int)_width/_cerealName.length();//שימוש בתכונה של האבא
    }
}
```

```
public class Tester
{
    public static void main (String[]args)
    {
        Box b=new Box(20,30,40);
        System.out.println("Volume is: "+b.calculateVolume());
        CerealBox cb=new CerealBox(30,40,50,"Cornflakes");
        System.out.println("Volume is: "+cb.calculateVolume());
        System.out.println("Cereal volume is: "+cb.calculateCerealVolume());
        System.out.println("font size:"+cb.calculateFontSize());
        WoodenBox wb=new WoodenBox(20,30,40);
        System.out.println("Volume is: "+wb.calculateVolume());
        System.out.println(wb instanceof WoodenBox);
         System.out.println(wb instanceof Box);
          System.out.println(b instanceof CerealBox);
          //polymorphism
          Box b2=new CerealBox(30,40,50,"Rice Crispies");
          System.out.println("Cereal volume is: "+((CerealBox)b2).calculateCerealVolume());
// חייבים כאן המרה כי מבחינת הקומפיילר מדובר ב-
//box
//  והוא לא מוצא את השיטה
//calculateCerealBox
          Box b1=new CerealBox(20,30,40,"Cornflakes");
          CerealBox b3=new CerealBox(30,40,50,"Rice Crispies");
          b1=b3;
          b3=(CerealBox)b1;
          System.out.println("Cereal volume is: "+b3.calculateCerealVolume());
          //array polymorphism
          Box[]boxArray=new Box[3];
          boxArray[0]=new Box(20,30,40);
          boxArray[1]=new CerealBox(30,40,50,"Cornflakes");
          boxArray[2]=new WoodenBox(20,30,40);
          for(int index=0;index<boxArray.length;index++)
            System.out.println(boxArray[index].calculateVolume());
// בזמן ריצה תמיד ילך לשיטה הספציפית ביותר
        }
    }
```

הפלט:

```
Volume is: 24000.0
Volume is: 60000.0
Cereal volume is: 54000.0
font size:4
Volume is: 19152.0
true
true
false
Cereal volume is: 54000.0
Cereal volume is: 54000.0
24000.0
60000.0
19152.0
```

```
public class Box
{
  protected double _length;
  protected double _height;
 protected double _width;
 protected static int _numBoxes=0;
  public Box(double len,double wid, double hei)
  {
      _length=len;
      _width=wid;
      _height=hei;
      _numBoxes++;
    }
    public static void printNumBoxes()
    {
        System.out.println("Num of boxes is:"+_numBoxes);
    }
    public static void resetCount(){
        _numBoxes=0;
    }
   public double calculateVolume()
   {
       return _length*_width*_height;
    }
}

public class Tester
{
     public static void main (String[]args)
     {
         Box b=new Box(20,30,40);

         CerealBox cb=new CerealBox(30,40,50,"Cornflakes");

         WoodenBox wb=new WoodenBox(20,30,40);

         Box.printNumBoxes();
         Box.resetCount();
         Box.printNumBoxes();
        }
    }
Num of boxes is:3
Num of boxes is:0
```

```
class ParentClass {

 private static int svar = 24;
 protected int _a=0;
   ParentClass() {
    //don't need this constructor
  }
  static int smethod() {
      _a=10;
    return 42;
  }
}
public class InheritStatics extends ParentClass {

  public void run() {

    svar++;
    System.out.println("svar is " + svar);
    System.out.println("smethod returns " + smethod());
  }
}


public class Test
{
    public static void main(String[] c) {
    InheritStatics i=new InheritStatics();
    i.run();
  }
}
```

מצא את השגיאות בתוכנית.
לאחר תיקון השגיאות, מה פלט התוכנית?

דוגמה יותר מורכבת. בחברה שלושה סוגי עובדים. עובד רגיל , עובד מכירות ועובד לפי שעות.

```
public abstract class Employee//מחלקה מופשטת, לא נוכל ליצור ממנה אובייקטים
{
  protected String _firstName;
  protected String _surName;
  protected String _id;
  protected String _telNumber;
  public Employee (String firstName,String surName,String id, String telNumber)
  {
    _firstName=firstName;
    _surName=surName;
    _id=id;
    _telNumber=telNumber;
   }
  public Employee(Employee other)//בנאי העתקה
 {
    _firstName=other._firstName;
    _surName=other._surName;
    _id=other._id;
    _telNumber=other._telNumber;
   }
   public String getFirstName()
   {
       return _firstName;
      }
      public String getSurName()
   {
       return _surName;
      }
      public String getId()
   {
       return _id;
      }
      public String getTelNumber()
   {
       return _telNumber;
      }
     public void  setFirstName(String firstName)
     {
        _firstName=firstName;
      }
      public void setSurName(String surName)
     {
        _surName=surName;
      }
      public void setId(String id)
     {
        _id=id;
      }
      public void  setTelNumber(String telNumber)
     {
        _telNumber=telNumber;
      }
```

```
public String toString()
{
   return _firstName+" "+_surName+" "+_id+" "+_telNumber+"\n";
}
 public abstract void printWageSlip();// שיטה מופשטת , נממש במחלקות היורשות
}
```

```
public class RegularEmployee extends Employee//עובד רגיל
{
  private double _salary;
  private double _pension;
  private int _holidayDays;
    public RegularEmployee(String firstName,String surName, String id,String telNumber, double
salary, double pension,int holidayDays)
    {
        super(firstName,surName,id,telNumber);
        _salary=salary;
        _pension=pension;
        _holidayDays=holidayDays;
}
   public RegularEmployee(RegularEmployee other)
   {
       super(other._firstName,other._surName,other._id,other._telNumber); //super(other);
       _salary=other._salary;
        _pension=other._pension;
        _holidayDays=other._holidayDays;
    }
     public double getSalary()
    {
         return _salary;
        }
        public double  getPension()
    {
         return _pension;
        }
        public int  getHolidayDays()
    {
         return _holidayDays;
        }
       public void  setHolidayDays(int holidayDays)
       {
          _holidayDays=holidayDays;
        }
       public void  setSalary(double salary)
       {
          _salary=salary;
        }
        public void  setPension(double pension)
       {
          _pension=pension;
        }
        public String toString()
        {
           String s= super.toString();//חייבים לציין כאן שמדובר בשיטה של האבא, כי יש גם שיטה בשם זה אצל הבן
           s=s+"Salary: "+_salary+" Pension (percent):  " +_pension+"\n";
           return s;
    }
    public void printWageSlip()
```

```
    {

System.out.println("*****************************************************************
*************************");
        System.out.print(toString()+" Wage is: "+( _salary-_pension*_salary)+"\n"+ " Amount set aside
for pension fund : "+(_salary*_pension)+" Holiday Days: "+_holidayDays+"\n");

System.out.println("*****************************************************************
*************************");
    }
    public void addBonus (double bonus)
    {
        _salary+=bonus;
    }
    public void useHolidayDays(int days)
    {
        _holidayDays-=days;
    }
    }
```

```
public class TempEmployee extends Employee//עובד זמני, עובד לפי שעות
{
  private double _rate;
  private  int _hours;
  public TempEmployee(String firstName,String surName, String id,String telNumber, double rate, int
hours)
    {
       super(firstName,surName,id,telNumber);
       _rate=rate;
       _hours=hours;
}
   public TempEmployee(TempEmployee other)
   {
      super(other._firstName,other._surName,other._id,other._telNumber); //super(other);
      _rate=other._rate;
       _hours=other._hours;
    }
     public double getRate()
    {
        return _rate;
       }
       public int  getHours()
    {
        return _hours;
       }
      public void  setRate(double rate)
      {
         _rate=rate;
       }
        public void  setHours(int hours)
      {
         _hours=hours;
       }
       public String toString()
       {
          String s= super.toString();
          s=s+"Hourly rate: "+_rate+" Hours:  " +_hours+"\n";
          return s;
    }
    public void printWageSlip()
    {

System.out.println("*****************************************************************
***************************");
         System.out.print(toString()+" Wage is: "+ _rate*_hours+"\n");

System.out.println("*****************************************************************
***************************");
       }
    }
```

```
public class SalesEmployee extends Employee//עובד לפי מכירות
{
  private double _commission;
  private double _sales;
  public SalesEmployee (String firstName,String surName, String id,String telNumber, double
commission, double sales)
    {
       super(firstName,surName,id,telNumber);
       _commission=commission;
       _sales=sales;
}
   public SalesEmployee(SalesEmployee other)
   {
      super(other._firstName,other._surName,other._id,other._telNumber); //super(other);
      _commission=other._commission;
       _sales=other._sales;
    }
     public double getCommission()
    {
        return _commission;
       }
       public double  getSales()
    {
        return _sales;
       }
      public void  setCommission(double commission)
      {
         _commission=commission;
       }
        public void  setSales(double sales)
      {
         _sales=sales;
       }
       public String toString()
       {
          String s= super.toString();
          s=s+"Commission (percent): "+_commission+" Sales made:  " +_sales+"\n";
          return s;
    }
    public void printWageSlip()
    {

System.out.println("*****************************************************************
***************************");
         System.out.print(toString()+" Wage is: "+ _commission*_sales+"\n");

System.out.println("*****************************************************************
***************************");
       }
    }
```

```
public class Company
{
  private Employee[] _companyEmployees;
  private int _actualNumEmployees;
  final int MAX_EMPLOYEES=100;

  public Company ()
  {
        _companyEmployees=new Employee[MAX_EMPLOYEES];
        _actualNumEmployees=0;
    }
   public boolean addEmployee(Employee employee)
   {
      if(_actualNumEmployees==MAX_EMPLOYEES)
            return false;
       else
         {
                    // כאן נרצה לקרוא לבנאי ההעתקה כדי לשכפל את האובייקט לפני הכנסתו למערך. כדי לדעת לאיזה בנאי צריך לקרוא יש
                    //לברר תחילה לאיזה בנאי העתקה צריך לקרוא
            if (employee instanceof TempEmployee)
                  _companyEmployees[_actualNumEmployees]=new TempEmployee((TempEmployee)employee);

           if (employee instanceof SalesEmployee)
                  _companyEmployees[_actualNumEmployees]=new SalesEmployee((SalesEmployee)employee);

           if (employee instanceof RegularEmployee)
                  _companyEmployees[_actualNumEmployees]=new RegularEmployee((RegularEmployee)employee);

            _actualNumEmployees++;
            return true;
          }
      }
       public void printAllWageSlips()
       {
           for(int i=0;i<_actualNumEmployees;i++)
                  _companyEmployees[i].printWageSlip();//תמיד ילך לשיטה לספציפית ביותר
              }

         public void division()//ספירת חלוקת החברה לפי סוג העובד
         {
             int tempCount=0,salesCount=0,regularCount=0;
             for(int i=0;i<_actualNumEmployees;i++)
                {
                    if (_companyEmployees[i] instanceof TempEmployee)
                      tempCount++;
             if (_companyEmployees[i] instanceof SalesEmployee)
                      salesCount++;
             if (_companyEmployees[i] instanceof RegularEmployee)
                       regularCount++;

                  }
            System.out.println("Regular employees: "+regularCount);
            System.out.println("Sales employees: "+salesCount);
            System.out.println("Temp employees: "+tempCount);
               }
           }
```

```
 public class Tester
{
    public static void main (String[]args)
    {
     TempEmployee dana=new TempEmployee("Dana","Cohen","123456789","03-7865432",35,45);
      SalesEmployee yael=new SalesEmployee("Yael","levy","987654321","03-
7867865",0.15,10000.0);
      RegularEmployee david=new RegularEmployee("David","Yisrael","888777799","09-
7845636",7500,0.18,20);
      Company aci=new Company();
      aci.addEmployee(dana);
      aci.addEmployee(yael);
      aci.addEmployee(david);
      aci.printAllWageSlips();
      aci.division();
    }
}
```

הפלט:

```
*****************************************************************************************
Dana Cohen 123456789 03-7865432
Hourly rate: 35.0 Hours:  45
 Wage is: 1575.0
*****************************************************************************************
*****************************************************************************************
Yael levy 987654321 03-7867865
Commission (percent): 0.15 Sales made:  10000.0
 Wage is: 1500.0
*****************************************************************************************
*****************************************************************************************
David Yisrael 888777799 09-7845636
Salary: 7500.0 Pension (percent):  0.18
 Wage is: 6150.0
 Amount set aside for pension fund : 1350.0 Holiday Days: 20
*****************************************************************************************
Regular employees: 1
Sales employees: 1
Temp employees: 1
```

שיטה יותר אלגנטית וגנרית יותר מהשיטה הקודמת לשכפול האובייקט לפני הכנסתו למערך.
אם נחליט ליצור סוג עובד נוסף, למשל סטודנט, אין צורך לגעת במחלקות הקיימות (חוץ השיטת החלוקה של החברה במקרה זה), נוכל רק לכתוב את המחלקה החדשה, עם מימוש שיטת השכפול. ראו חידושים בכחול.

נוסיף למחלקה
Employee

```
public abstract Employee duplicate();
```

נוסיף למחלקה
RegularEmployee

```
public  Employee duplicate()//קריאה לבנאי העתקה של עובד רגיל
{
   return new RegularEmployee(this);
  }
```

נוסיף למחלקה
SalesEmployee

```
public  Employee duplicate()
{
   return new SalesEmployee(this);
  }
```

נוסיף למחלקה
TempEmployee

```
public  Employee duplicate()
{
   return new TempEmployee(this);
  }
```

ועכשיו השיטה
addEmployee
בתוך
Company
נראת כך:

```
public boolean addEmployee(Employee employee)
{
   if(_actualNumEmployees==MAX_EMPLOYEES)
        return false;
    else
      {
     _companyEmployees[_actualNumEmployees]=employee.duplicate();//הוא יודע לאיזה שיטה ללכת, לספציפית ביותר
        _actualNumEmployees++;
        return true;
      }
   }
```

שיטה שלישית, לא פותרת את הצורך לעדכן את המחלקה Company אם נוסיף סוג עובד נוסף בעתיד.
כשנקרא לשיטה שמוסיפה עובד למערך, נגיע לשיטה המתאימה לפי סוג העובד.
שם מתבצע קריאה לבנאי ההעתקה, ואז נוסיף את האובייקט למערך בעזרת שיטה addAnyEmployee.

```
public class Company
{
  private Employee[] _companyEmployees;
  private int _actualNumEmployees;
  final int MAX_EMPLOYEES=100;

  public Company ()
  {

        _companyEmployees=new Employee[MAX_EMPLOYEES];
        _actualNumEmployees=0;

    }
    private boolean addAnyEmployee(Employee employee)
    {
       if(_actualNumEmployees==MAX_EMPLOYEES)
             return false;
        else
          {
             _companyEmployees[_actualNumEmployees]=employee;

             _actualNumEmployees++;
             return true;
           }
       }


public boolean addEmployee(SalesEmployee e)
 {
     return addAnyEmployee(new SalesEmployee(e));
 }

public boolean addEmployee(TempEmployee e)
 {
   return addAnyEmployee(new TempEmployee(e));
 }
public boolean addEmployee(RegularEmployee e)
  {
   return addAnyEmployee(new RegularEmployee(e));
 }
```

המשך המחלקה …

```
public abstract class Employee
{
  protected String _firstName;
  protected String _surName;
  protected String _id;
  protected String _telNumber;

 public Employee (String firstName,String surName,String id, String telNumber)
 {
    _firstName=firstName;
    _surName=surName;
    _id=id;
    _telNumber=telNumber;
   }

   public Employee(Employee other)
 {
    _firstName=other._firstName;
    _surName=other._surName;
    _id=other._id;
    _telNumber=other._telNumber;
   }
   public String getFirstName()
   {
       return _firstName;
      }
      public String getSurName()
   {
       return _surName;
      }
      public String getId()
   {
       return _id;
      }
      public String getTelNumber()
   {
       return _telNumber;
      }
     public void  setFirstName(String firstName)
     {
        _firstName=firstName;
      }
      public void setSurName(String surName)
     {
        _surName=surName;
      }
      public void setId(String id)
     {
        _id=id;
      }
      public void  setTelNumber(String telNumber)
     {
```

```
   _telNumber=telNumber;
}
public String toString()
{
    return _firstName+" "+_surName+" "+_id+" "+_telNumber+"\n";
}
 public abstract void printWageSlip();
}
```

```
public class TempEmployee extends Employee
{
  protected double _rate;//for oustudent, must make protected
  protected  int _hours;
    public TempEmployee(String firstName,String surName, String id,String telNumber, double rate,
int hours)
    {
        super(firstName,surName,id,telNumber);
        _rate=rate;
        _hours=hours;
}
   public TempEmployee(TempEmployee other)
   {
       super(other._firstName,other._surName,other._id,other._telNumber);
       _rate=other._rate;
        _hours=other._hours;
    }
     public double getRate()
    {
         return _rate;
        }
        public int  getHours()
    {
         return _hours;
        }
       public void  setRate(double rate)
       {
           _rate=rate;
        }
         public void  setHours(int hours)
       {
           _hours=hours;
        }
        public String toString()
        {
            String s= super.toString();
            s=s+"Hourly rate: "+_rate+" Hours:  " +_hours+"\n";
            return s;
    }
    public void printWageSlip()
    {

System.out.println("*****************************************************************
***************************");
         System.out.print(toString()+" Wage is: "+ _rate*_hours+"\n");

System.out.println("*****************************************************************
***************************");
        }
    }
```

```
public interface Student
{
  int COURSE_FEE=2000;
  void printAllCourses();
  void printFees();
}
```

```
public class OUStudent extends TempEmployee implements Student
{
  private String _course1;
  private String _course2;
  private String _course3;
  private String _course4;
  public OUStudent(String firstName,String surName, String id,String telNumber, double rate, int
hours,String c1,String c2,String c3,String c4)
    {
     super( firstName, surName,  id, telNumber, rate,  hours);
     _course1=c1;
     _course2=c2;
     _course3=c3;
     _course4=c4;
    }
   public OUStudent(OUStudent other)
   {
      super( other._firstName,other._surName,  other._id, other._telNumber, other._rate,  other._hours);
    _course1=other._course1;
     _course2=other._course2;
     _course3=other._course3;
     _course4=other._course4;
    }
public   void printAllCourses()//appears in  interface
  {
      System.out.println("courses:    \n");
     System.out.println(_course1+"\n");
     if (_course2!=null)
          System.out.println(_course2+"\n");
    if (_course3!=null)
          System.out.println(_course3+"\n");
     if (_course4!=null)
          System.out.println(_course4+"\n");
       }
  public void printFees()//appears in  interface
  {
      int countCourses=1;
     if (_course2!=null)
          countCourses++;
    if (_course3!=null)
         countCourses++;
     if (_course4!=null)
          countCourses++;
          System.out.println("Course fees:" + (COURSE_FEE*countCourses));
       }
       }
```

```
public class Tester
{
    public static void main (String[]args)
    {
     OUStudent Rafi=new OUStudent("Rafi","Cohen","123456789","03-7865432",35,45,"Intro to
cs","Statistics1",null,null);
     Rafi.printAllCourses();
     Rafi.printFees();
     Rafi.printWageSlip();
     System.out.Println(Rafi.toString());
    }
}
```

הפלט:

```
courses    :

Intro to cs

Statistics1

Course fees:4000
*****************************************************************************************
Rafi Cohen 123456789 03-7865432
Hourly rate: 35.0 Hours:  45
 Wage is: 1575.0
*****************************************************************************************
Rafi Cohen 123456789 03-7865432
                                        Hourly rate: 35.0 Hours:  45
```

```
public interface Residence
{
  int  RENT=0, SELL=1;
  String toString();
  int getNumRooms();
  double getSize();
  int getPrice();
  int getSellOrRent();
  void setPrice(int price);

}
```

```
public class House implements Residence
{
  private String _address;
   private int _sellOrRent;
   private int _numRooms;
   private double _size;
   private int _price;
   private double _gardenSize;
  public House(String address,int numRooms,double size,int price,int sellOrRent, double gardenSize  )
   {
      _address=address;
      _numRooms=numRooms;
      _size=size;
      _price=price;
       _sellOrRent=sellOrRent;
     _gardenSize=gardenSize;
   }
   public String toString()
   {
        String s="\n";
        s+=_address+"\n";
        s+="Rooms: "+_numRooms+" Size: "+_size+" Garden size: "+_gardenSize+" Price: "+_price;
        if( _sellOrRent==SELL)
            s+= " For Sale";
         else
            s+=" For Rent";
          return s;
     }
  public int getNumRooms()
  {
  return _numRooms;
}
  public double getSize()
  {
     return _size;
   }
  public int getPrice()
  {
     return _price;
   }
  public int getSellOrRent()
  {
     return _sellOrRent;
   }
  public void setPrice(int price)
  {
     _price=price;
   }

   public double getGardenSize()
  {
     return _size;
   }
```

```
}


public class Apartment implements Residence
{
   private String _address;
   private int _sellOrRent;
   private int _numRooms;
   private double _size;
   private int _price;
   private double _balconySize;
   private int _floor;
   private boolean _elevator;

   public Apartment(String address,int numRooms,double size,int price,int sellOrRent, double
balconySize ,boolean elevator,int floor )
   { _address=address;
      _numRooms=numRooms;
      _size=size;
      _balconySize=balconySize;
      _price=price;
       _sellOrRent=sellOrRent;
      _floor=floor;
      _elevator=elevator;
   }
   public String toString()
   {

        String s="\n";
        s+=_address+"\n";
        s+="Rooms: "+_numRooms+" Size: "+_size+" Balcony size: "+_balconySize+" Price:
"+_price+ " Floor : "+ _floor;
        if( _sellOrRent==SELL)
            s+= " For Sale";
         else
            s+=" For Rent";
         if( _elevator==true)
            s+= "elevator";
         return s;


   }
  public int getNumRooms()
  {
  return _numRooms;
}
  public double getSize()
  {
     return _size;
   }
  public int getPrice()
  {
     return _price;
```

```
    }
   public int getSellOrRent()
   {
      return _sellOrRent;
    }

   public void setPrice(int price)
   {
      _price=price;
    }

   public  double getBalconySize()
   {
      return _size;
    }
}
```

```
public class Residences
{
  private Residence[] _companyResidences;
  private int _actualNumResidences;
  final int MAX=100;

  public Residences ()
  {

        _companyResidences=new Residence[MAX];
        _actualNumResidences=0;

    }
    public boolean addResidence(Residence residence)
    {
       if(_actualNumResidences==MAX)
             return false;
        else
        {
            _companyResidences[_actualNumResidences]=residence;//aliasing! Need to call relevant
copy constructor! See employee example above.
             _actualNumResidences++;
             return true;
          }
       }
       public String toString()
       {
          String s="";
          for(int i=0;i<_actualNumResidences;i++)
                s+=_companyResidences[i].toString();
                return s;
             }

         public void division()
         {
            int houseCount=0,apartmentCount=0;
            for(int i=0;i<_actualNumResidences;i++)
              {
                 if (_companyResidences[i] instanceof House)
                   houseCount++;
                 if (_companyResidences[i] instanceof Apartment)
                   apartmentCount++;


                }
                 System.out.println("Houses: "+houseCount);
                  System.out.println("Apartments: "+apartmentCount);

             }
             public double getLargestGarden()
             {
                double MaxGarden=0;
                for(int i=0;i<_actualNumResidences;i++)
```

```
        {
            if (_companyResidences[i] instanceof House)
                if(((House)_companyResidences[i]).getGardenSize()>MaxGarden)
                        MaxGarden=((House)_companyResidences[i]).getGardenSize();
                    }
              return MaxGarden;
    }
    public String printAllForSale(){
        String s="";
       for(int i=0;i<_actualNumResidences;i++)
        {
            if (_companyResidences[i] .getSellOrRent()==1)
                 s+=_companyResidences[i].toString();
                }
          return s;
       }

}
```

```
public class Tester
{
    public static void main (String[]args)
    {
     House h1=new House("13, Bar Ilan",6,165,130000,1,120);
      House h2=new House("23, Ben Gurion",7,185,180000,0,150);
      House h3=new House("33, Weizmann",8,145,190000,1,110);
       Apartment a1=new Apartment("13, Shir Sameach",3,125,99000,0,10,true,3);
       Apartment a2=new Apartment("45, Hermon",4,125,119000,1,20,false,2);
       Apartment a3=new Apartment("13, Jabotinsky",5,125,199000,0,15,true,3);
       Residences all=new Residences();
      all.addResidence(h1);
      all.addResidence(a1);
      all.addResidence(h2);
      all.addResidence(a2);
      all.addResidence(h3);
      all.addResidence(a3);

      System.out.println(all.toString());
      all.division();
      System.out.println(all.getLargestGarden());
      System.out.println(all.printAllForSale());
   }
}
```

הפלט:

```
13, Bar Ilan
Rooms: 6 Size: 165.0 Garden size: 120.0 Price: 130000 For Sale
13, Shir Sameach
Rooms: 3 Size: 125.0 Balcony size: 10.0 Price: 99000 Floor : 3 For Rentelevator
23, Ben Gurion
Rooms: 7 Size: 185.0 Garden size: 150.0 Price: 180000 For Rent
45, Hermon
Rooms: 4 Size: 125.0 Balcony size: 20.0 Price: 119000 Floor : 2 For Sale
33, Weizmann
Rooms: 8 Size: 145.0 Garden size: 110.0 Price: 190000 For Sale
13, Jabotinsky
Rooms: 5 Size: 125.0 Balcony size: 15.0 Price: 199000 Floor : 3 For Rentelevator
Houses: 3
Apartments: 3
185.0

13, Bar Ilan
Rooms: 6 Size: 165.0 Garden size: 120.0 Price: 130000 For Sale
45, Hermon
Rooms: 4 Size: 125.0 Balcony size: 20.0 Price: 119000 Floor : 2 For Sale
33, Weizmann
Rooms: 8 Size: 145.0 Garden size: 110.0 Price: 190000 For Sale
```