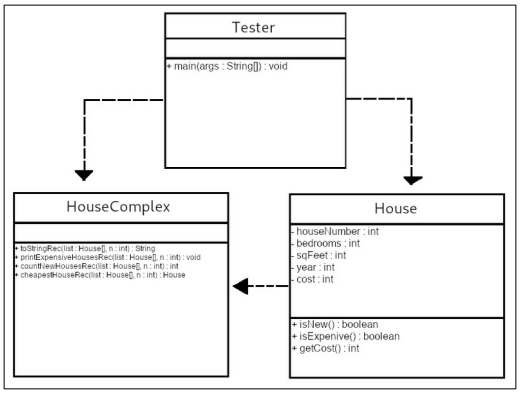
UML



Output

```
Options
All the houses in houseList, in order:
House #1        Bedrooms: 3     Square Feet: 1790       Year: 2017      Cost: 310
House #4        Bedrooms: 2     Square Feet: 1480       Year: 2004      Cost: 280
House #23       Bedrooms: 6     Square Feet: 2120       Year: 2005      Cost: 290
House #12       Bedrooms: 1     Square Feet: 2100       Year: 1976      Cost: 220
House #8        Bedrooms: 5     Square Feet: 7600       Year: 2009      Cost: 370
House #15       Bedrooms: 4     Square Feet: 2800       Year: 1998      Cost: 320
House #88       Bedrooms: 5     Square Feet: 4020       Year: 2019      Cost: 460

All the expensive houses:
House #1        Bedrooms: 3     Square Feet: 1790       Year: 2017      Cost: 310
House #8        Bedrooms: 5     Square Feet: 7600       Year: 2009      Cost: 370
House #15       Bedrooms: 4     Square Feet: 2800       Year: 1998      Cost: 320
House #88       Bedrooms: 5     Square Feet: 4020       Year: 2019      Cost: 460

The amount of new houses: 5

The cheapest house on the market:
House #12       Bedrooms: 1     Square Feet: 2100       Year: 1976      Cost: 220


 Can only enter input while your programming is running
BlueJ: Terminal Window - ObjectRecursion
```

Tester.java

```
import java.util.Scanner;
import java.io.File;

/**
 * @author (Kyle Guarco)
 * @version (July 12, 2020)
 */
public class Tester
{
    public static void main(String[] args)
    {
        House[] houseList = new House[70];
        int count = 0;

        try {
            Scanner scan = new Scanner(new File("inData.txt"));

            while (scan.hasNextLine())
            {
                // Format: houseNumber, bedrooms, sqFeet, year, cost
                houseList[count] = new House(scan.nextInt(), scan.nextInt(),
                        scan.nextInt(), scan.nextInt(), scan.nextInt());

                scan.nextLine();
                count++;
            }

            scan.close();
        } catch (Exception e) {
            // Either an IOException or a NumberFormatException. Print the message!
            System.err.println(e.getMessage());
            System.err.println("There was an error reading \"inData.txt\". Exiting...");
            System.exit(-1);
        }

        HouseComplex complex = new HouseComplex();

        // Using 'print' here to prevent an extra newline
        System.out.print("All the houses in houseList, in order:");
        System.out.println(complex.toStringRec(houseList, count));

        System.out.println("\nAll the expensive houses:");
        complex.printExpensiveHousesRec(houseList, count);

        System.out.println("\nThe amount of new houses: " +
```

```
complex.countNewHousesRec(houseList, count));

        System.out.println("\nThe cheapest house on the market: \n" +
complex.cheapestHouseRec(houseList, count));
    }
}
```

HouseComplex.java

```
import java.util.ArrayList;

/**
 * @author (Kyle Guarco)
 * @version (July 12, 2020)
 */
public class HouseComplex
{
    public String toStringRec(House[] list, int n)
    {
        if (n < 1)
            return "";

        return toStringRec(list, n - 1) + "\n" + list[n - 1];
    }

    public void printExpensiveHousesRec(House[] list, int n)
    {
        if (n < 1)
            return;

        printExpensiveHousesRec(list, n - 1);

        House house = list[n - 1];
        if (house.isExpensive())
            System.out.println(house);
    }

    public int countNewHousesRec(House[] list, int n)
    {
        if (n < 1)
            return 0;

        int add = list[n - 1].isNew() ? 1 : 0;

        return countNewHousesRec(list, n - 1) + add;
    }

    public House cheapestHouseRec(House[] list, int n)
    {
        if (n < 1)
            return null;

        House house = list[n - 1];
        House funcHouse = cheapestHouseRec(list, n - 1);
```

```
        if (funcHouse == null)
            return house;

        return funcHouse.getCost() > house.getCost() ? house : funcHouse;
    }
}
```

House.java

```
/**
 * @author (Kyle Guarco)
 * @version (July 12, 2020)
 */
public class House
{
    // 'cost' is given in thousands of dollars
    private int houseNumber, bedrooms, sqFeet, year, cost;

    public House(int houseNumber, int bedrooms, int sqFeet, int year, int cost)
    {
        this.houseNumber = houseNumber;
        this.bedrooms = bedrooms;
        this.sqFeet = sqFeet;
        this.year = year;
        this.cost = cost;
    }

    @Override
    public String toString()
    {
        return String.format("House #%d\t Bedrooms: %d\t Square Feet: %d\t Year: %d\t Cost:
%d\t",
                houseNumber, bedrooms, sqFeet, year, cost);
    }

    public boolean isNew()
    {
        return year >= 2000;
    }

    public boolean isExpensive()
    {
        return cost > 300;
    }

    public int getCost()
    {
        return cost;
    }
}
```