Bulding.java

Building
- mLength : int - mWidth : int - mLotLength : int - mLotWidth : int
+ Building(int length, int width, int lotLength, int lotWidth) // constructor + getLength() : int + getWidth() : int + getLotLength() : int + getLotWidth() : int + setLength(int) : void + setWidth(int) : void + setLotLength(int) : void + setLotWidth(int) : void + calcBuildingArea() : int + calcLotArea() : int + toString() : String

Notes on ***Building***:

- The setter and getter methods for Building are public since the auto-grader will call these methods to test that they work, so they must be public. Note that we will assume the user has provided valid parameters whenever a building is created or modified – we will not verify that the lot is actually large enough to hold the building.
- Create the *toString()* method to simply return a string representation of the Building object (you can simply return a string of the building dimensions; e.g., `"a 50x90 building"`).

```java
package mooc.vandy.java4android.buildings.logic;

/**
 * This is the Building class file.
 */
public class Building {

    // TODO - Put your code here.
    private int mLength;
    private int mWidth;
    private int mLotLength;
    private int mLotWidth;

    //constructor
    public Building(int length, int width, int lotLength, int lotWidth) {
        this.mLength = length;
        this.mWidth = width;
        this.mLotLength = lotLength;
        this.mLotWidth = lotWidth;
    }

    //getters

    public int getLength() {
        return mLength;
    }

    public int getWidth() {
        return mWidth;
    }

    public int getLotLength() {
        return mLotLength;
    }

    public int getLotWidth() {
        return mLotWidth;
    }

```

```
    //setters

    public void setLength(int length) {
        this.mLength = length;
    }

    public void setWidth(int width) {
        this.mWidth = width;
    }

    public  void setLotLength(int lotLength) {
        this.mLotLength = lotLength;
    }

    public void setLotWidth(int lotWidth) {
        this.mLotWidth = lotWidth;
    }

    //calculate the building area
    public int calcBuildingArea() {
        return mLength * mWidth;
    }

    //calculate the lot Area
    public int calcLotArea() {
        return  mLotLength * mLotWidth;
    }

    public String toString() {
        return "Owner:n/a";
    }

}

}
```

House.java

House extends Building
- mOwner : String - mPool : boolean

+ House(int length, int width, int lotLength, int lotWidth) // constructor
+ House(int length, int width, int lotLength, int lotWidth, String owner) // constructor
+ House(int length, int width, int lotLength, int lotWidth, String owner, boolean pool) // constructor
+ getOwner() : String
+ hasPool() : boolean
+ setOwner(String) : void
+ setPool(boolean) : void
+ toString() : String
+ equals(Object) : boolean

```java
 * This is the House class file that extends Building.
 */
public class House
        extends Building {

    // TODO - Put your code here.
    private String mOwner;
    private boolean mPool;

    public House(int length, int width, int lotLength, int lotWidth) {
        super(length, width, lotLength, lotWidth);
        this.mOwner = null;
        this.mPool = false;
    }

    //another constructor with owner name
    public House(int length, int width, int lotLength, int lotWidth, String owner) {
        super(length, width, lotLength, lotWidth);
        this.mOwner = owner;
        this.mPool = false;
    }

    //another constructor with owner name and pool status
    public House(int length, int width, int lotLength, int lotWidth, String owner, boolean pool) {
        super(length, width, lotLength, lotWidth);
        this.mOwner = owner;
        this.mPool = pool;
    }

    //getters
    public String getOwner() {
        return mOwner;
    }

    public boolean hasPool(){
        return mPool;
    }

```

```
42      //setters
43      public void setOwner(String owner) {
44          this.mOwner = owner;
45      }
46
47      public  void setPool(boolean pool) {
48          this.mPool = pool;
49      }
50
51      @Override
52      public String toString() {
53          return "Owner: " + (mOwner!=null ? mOwner : "n/a") +
54                  (hasPool() ? "; has a pool" : "") +
55                  (calcLotArea() > calcBuildingArea() ? "; has a big open space" : "");
56      }
57
58      @Override
59      public boolean equals(Object o) {
60
61          House house = (House) o;
62          return o instanceof House && (mPool == house.hasPool()) && (calcBuildingArea() == house.calcBuildingArea
                ());
63
64      }
65  }
```

Cottage.java

Cottage extends House
- mSecondFloor : boolean

+ Cottage(int dimension, int lotLength, int lotWidth) // constructor
+ Cottage(int dimension, int lotLength, int lotWidth, String owner, boolean secondFloor) // constructor
+ hasSecondFloor() : boolean
+ toString() : String

```java

/**
 * This is the cottage class file.  It is a subclass of House.
 */
public class Cottage
      extends House {

    // TODO - Put your code here.
    private boolean mSecondFloor;

    //constructors
    public Cottage(int dimension, int lotLength, int lotWidth) {
        super(dimension, dimension, lotLength, lotWidth);
        mSecondFloor = false;
    }

    //another constructor with second floor parameter

    public Cottage(int dimension, int lotLength, int lotWidth, String owner,boolean secondFloor) {
        super(dimension, dimension, lotLength, lotWidth, owner);
        this.mSecondFloor = secondFloor;
    }

    public boolean hasSecondFloor() {
        return mSecondFloor;
    }

    @Override
    public String toString() {
        return super.toString() + (hasSecondFloor() ? "; is a two story cottage" : "");
    }

}
```

Office.java

Office extends Building
- mBusinessName : String - mParkingSpaces : int - sTotalOffices : int //static variable

+ Office(int length, int width, int lotLength, int lotWidth) // constructor
+ Office(int length, int width, int lotLength, int lotWidth, String businessName) // constructor
+ Office(int length, int width, int lotLength, int lotWidth, String businessName, int parkingSpaces) // constructor
+ getBusinessName() : String
+ getParkingSpaces() : int

+ getTotalOffices() : int // static method + setBusinessName(String) : void + setParkingSpaces(int) : void + toString() : String + equals(Object) : boolean

```
3  /**
4   * This is the office class file, it is a subclass of Building.
5   */
6  public class Office
7        extends Building {
8
9      // TODO - Put your code here.
10     private String mBusinessName;
11     private int mParkingSpaces;
12     private static int sTotalOffices=0;
13
14     public Office(int length, int width, int lotLength, int lotWidth){
15         super(length, width, lotLength, lotWidth);
16         sTotalOffices++;
17     }
18
19     public Office(int length, int width, int lotLength, int lotWidth, String businessName){
20         this(length, width, lotLength, lotWidth);
21         mBusinessName = businessName;
22     }
23
24     public Office(int length, int width, int lotLength, int lotWidth, String businessName, int parkingSpaces){
25         this(length, width, lotLength, lotWidth, businessName);
26         mParkingSpaces = parkingSpaces;
27     }
28
29     public String getBusinessName() {
30         return mBusinessName;
31     }
32
33     public int getParkingSpaces() {
34         return mParkingSpaces;
35     }
```

```
36
37      public static int getTotalOffices() {
38          return sTotalOffices;
39      }
40
41      public void setBusinessName(String mBusinessName) {
42          this.mBusinessName = mBusinessName;
43      }
44
45      public void setParkingSpaces(int mParkingSpaces) {
46          this.mParkingSpaces = mParkingSpaces;
47      }
48
49      @Override
50      public String toString() {
51          String s = "Business: ";
52          if(mBusinessName == null)
53              s += "unoccupied ";
54          else
55              s += getBusinessName();
56
57          if(getParkingSpaces()!=0)   s += "; has " + getParkingSpaces() + " parking spaces ";
58
59          return  s + "(total offices: " + getTotalOffices() + ")";
60      }
61
62      @Override
63      public boolean equals(Object o) {
64          if (this == o) return true;
65          if (o == null || getClass() != o.getClass()) return false;
66          Office office = (Office) o;
67          return (this.calcBuildingArea() == office.calcBuildingArea()) && ( mParkingSpaces == office
                .getParkingSpaces());
68      }
69  }
```

Neighborhood.java

Neighborhood
// no instance variables

- Neighborhood() // private constructor
+ print(Building[] buildings, String header, OutputInterface out) : void
// static method
+ calcArea(Building[] buildings) : int // static method

```java
package mooc.vandy.java4android.buildings.logic;

import java.io.FileNotFoundException;

import mooc.vandy.java4android.buildings.ui.OutputInterface;

/**
 * This Neighborhood utility class provides static helper methods the
 * print a Building List and calculate the area of a Building list.
 * A utility class in Java should always be final and have a private
 * constructor.
 */
public final class Neighborhood {

    // TODO - Put your code here.
    public static void print(Building[] buildings, String header, OutputInterface out) {

    }

    public static int calcArea(Building[] buildings) {
        int totalLotArea = 0;
        for(int i = 0; i < buildings.length; i++) {
            totalLotArea += buildings[i].calcLotArea();
        }
        return totalLotArea;
    }
}
```