Note: No Brainstorm this week. This lab gives fairly detailed instructions on how to complete the assignment. The purpose is to get more practice with OOP.

Introduction

This lab introduces you to additional concepts of Object Oriented Programming (OOP), arguably the dominant programming paradigm in use today. In the paradigm, a program consists of component parts (objects) that are independent of each other and that interact in order to achieve a desired result. At a very basic level, an object-oriented program consists of the following types of things:

- **Classes**: The template or blueprint from which objects are created.
- **Objects**: The instantiation of a class. Once a class (the template) has been defined, many instances of it can be created.
- **Methods**: The named procedures or functions defined for an object. A method accepts input arguments and contains statements that are executed when the method is invoked. Some methods return values.
- **Instance Variables**: Variables that are initialized when an object is created. Each instance of a class can have distinct values for these variables.

In this lab, you will practice working with these fundamentals. A skeleton of code has been written for you defining the basic class and method structure that you are to use. You will then add code to the existing skeleton methods so that the classes function as intended. You are then to use a testing program illustrate how the component parts of what you have created interact.

Lab Objectives

By the end of the lab, you should be able to create classes utilizing:

- access modifiers;
- instance variables (also called fields);
- methods which return values and void methods (which do not return any value);
- methods calling other methods;
- method overloading;
- accessor and mutator methods;
- The `equals()` method;
- The `toString()` method.

What to Submit

Only `Person.java` and `House.java` should be submitted to eLC for grading.
Instructions

Three partially implemented files can be found on the course website. You should download these files and put them into a new project in your workspace in eclipse.

The file House.java contains a partial definition for a class representing a house that is bought and sold on a real estate marketplace. It will hold data such as whether or not is for sale, what its price is, and its area and volume.

The file Person.java contains a partial definition for a class representing a person in a real estate market. Functionally, the Person class exists to encapsulate data about users of our program, such as whether or not they own a home, how much money they have etc.

Save each to your computer and study them to see what methods they contain. Read and understand the method specifications of each. Then complete both the House class and Person class as described below. Note that you should test your methods in Main.java.

1. You must include a comment stating your name, the date, program purpose, and containing a statement of academic honesty. When writing, you must also abide by the Java formatting and naming conventions.

2. The House class is partially defined with an enum Color already defined for you. Fully define the House class by following the instructions below:

   a. Four private doubles instance variables price, width, length, height have been declared for you. Yes, we are assuming that the house is a 3-D rectangular box for simplicity. Next, we’ve also declared a private Color instance variable called color and a private boolean instance variable named forSale, which represent the color of the house and whether or not the house is for sale respectively.

   b. We will use two different constructors for House by utilizing method overloading, which is when a class contains two or more methods (or constructors) with the same name. The default constructor (no parameters) will generate an instance of House with random values for its instance variables. The bounds of what these should be are defined by the method specification (the multiline comment above the method header). The default constructor has already been written for you.

      Hint: Note how we accessed and assigned instance variables without using the this keyword in the default constructor. We can do this because there are no parameters or local variables with the same name.

   c. The second constructor is much simpler to implement than the first. You merely need to match the parameter and instance variables. There is a twist however: the second constructor’s parameters share the same name as the object’s instance variables. We must therefore use the this keyword to access our object’s instances variables. For example, to access the color instance variable of the House object, we write this.color. Assign the instance variables to the matching parameters.
Also note that when a house is created it is for sale. Therefore, the forSale instance variable should be set to true at initialization.

d. The toString() method has already been defined for you. Notice how this method calls the object’s other methods. If those are not complete, this method will not work!

e. Fill in the code for the equals() method. A house is equal to another if ALL of their instance variables are equal. For this method, check all instance variables other than forSale. Two houses can be the same even if one is for sale and the other isn’t.

f. Next we will define the area() method, which returns the area of our house. The area is simply the width instance variable multiplied by the length instance variable.

g. Define the volume() method. The volume is equal to area() multiplied by the height.

h. Next we will define our “setter” or “mutator” methods. These methods modify/mutate/set the instance variables of House objects. Each of their signatures begins with set, and work similarly to our second constructor. The instance variable being “mutated” will be assigned to the parameter of each method.

i. Finally we will create our accessor methods, also called “getters.” All these methods do is return the instance variable given away by their name. They are always prefixed by “get-” or “is-”. For example getPrice() returns the price instance variable of the House object.

3. The next step is to test the House class that you have just defined. Open Main.java. You will notice a large area of commented out code. This is code for a small real estate application that will use your classes as building blocks to run. We will uncomment this after we finish the Person class. At the beginning of the main method write:

```
House h = new House();
```

Now add a breakpoint right after, then run your program in debug mode and observe the values in your instance variables. Now try printing your house to the console:

```
System.out.println(h.toString());
```

Compare the output on the console to the instance variables you observed. If need be fix your getter methods to get the right output.

Continue this pattern of testing each of your class’s methods by using Eclipse’s debugging, and once you are confident they work properly, move on to step 4.

4. Start writing the Person class, following the instructions detailed below:

   a. There are four private instance variables: String name, int age, double cash and House house. The instance variable house represents the person’s home, assuming they have
one. If they do not, house will be a null reference. Note that you should not declare any
other instance variables other than these four.

b. Define the default constructor (no parameters) so that the person who is instantiate has the
name “John L.”, has an age of 21 years old, has a single penny of cash and no home. Note
that you must set the house instance variable to null.

c. A second constructor is overloaded with several different sets of parameters. Fill in the code
for the constructor that does not contain the house parameter. The constructor should
assign its instance variables to the matching parameters and set the house instance variable
to null.

d. Fill in the code for the third constructor, which does contain a house parameter. This
constructor must in addition to matching the correct instance variable to the correct
parameters, must mutate the house object passed in by the parameter so that it is no
longer for sale.

e. Fill in the code for method toString(), which should return a “string” containing values for
the name, age, cash, and house (if the user has one). The returned string value should be
formatted as below:

Name: name
Age: age
Cash: $cash
Home:

   house.toString() // do not print this line if
                     // house is null

f. Fill in the code for the accessor methods: getName(), getAge(), getCash(), and getHouse().

g. The last of the non-mutating methods we need to complete is ownsAHouse(). This will
return true if the user has a home. This method is not very complex, but it makes our code
more human-readable when it is used by other methods we will create later in this lab.

5. Stop here and test that your Person class is currently working by setting breakpoints in its methods
and testing it in the Main class. This will also help you observe the flow of an object-oriented
program. As is, the Person class does not have any methods that mutate its instance variables (and
thereby the Person object), so you should ensure everything is correct now before the class
becomes more complex.

6. Complete the Person class following the instructions detailed below:

   a. Complete the addCash() method. Simply add the amount parameter to the cash instance
      variable.

   b. Complete the sellHome() method. The Person instance will have the price of the house
      added to his cash, and the house will be for sale again and the program should print “name
      has sold their house!” However, if this user does not have a house, the program should print
      “name has no house to sell.” After the house has been sold back to the market, the Person’s
      house instance variable is set to null.
Lab 10 – Classes and Methods

c. Fill in the code for the `buyHouse()` method. Very specific requirements have to be met for the person to purchase a home successfully:

   i. They must not already own a home
   ii. They must be able to afford the home
   iii. The home must actually be for sale

   If the person meets these requirements, the person is subsequently charged the price of the house (their `cash` decreases by that price) and the `house` instance variable is assigned to the newly purchased house. Ensure you set `house`'s instance variable `forSale` to false.

d. Test your methods using the `Main` class once again.

7. Once you are confident each of these classes functions as expected, uncomment the code in the main method, and run the application!

One strength of Object-Oriented Programming is that it enables two programmers to cooperate easily. This application was written and is now able to run without the programmer knowing your specific implementation of each of these classes and methods. With Object Oriented programming, teams can work together on projects without every programmer worrying about implementation of each class. With the complexity today’s applications, this advantage is huge.

eLC Submission and Grading

After you have completed and thoroughly tested your programs, upload `House.java` and `Person.java` to eLC to receive credit. Always double check that your submission was successful on eLC!

Additional Requirements

These are things that make the graders lives easier, and ultimately, you will see in the real world as well. Remember that the teaching staff does not want to touch your code after they gave you requirements; they want to see the perfect results they asked for! Here is a checklist of things you can lose points for:

- (100 points) If the source file(s) are not submitted before the specified deadline’s late period ends (48 hours after the deadline) or if they do not compile.
- (25 points) Late penalty will be deducted as per the course syllabus.
- (10 points) If the source file(s)/class(es) are named incorrectly (case matters!)
- (10 points) If your source file(s) have a package declaration at the top
- (20 points) If the order of your output does not match the examples
- (15 points) If the output text does not match exactly (unless otherwise specified in the lab/project description)
- (100 points) If you use non-standard Java libraries, or other code specifically disallowed by the lab/project.
- (10 points) If you are missing your Statement of Academic Honesty
- If your (10 points) comments or (10 points) variables are “lacking”
  - Here, “lacking” means that you or a TA can find any lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like `b`, `bb`, `bbb`, etc. will almost never be acceptable)
- (10 points) Indentation is not consistent throughout your source code
Lab 10 – Classes and Methods

- Refresh your memory of indentation patterns in chapter 2 in the course textbook
- Be careful of a combination of tabs and spaces in your files (use one or the other)!

If any of the above do not make sense to you, talk to a TA, or ask on Piazza!

Examples

Your program should work correctly and follow the examples below. Each example is a separate run of a correctly working program. Please note that some long lines of output are wrapped around multiple lines in this document, and it is okay if your output displays them on a single line.

Enter your name:
John Smith
Enter your age (in years):
50
Enter the amount of money you have:
75000
You are:
Name: John Smith
Age: 50 years old
Cash: $ 75,000.00

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

4
Name: John Smith
Age: 50 years old
Cash: $ 75,000.00

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

1
House 1
Color:YELLOW
Dimensions: 184.490 x 105.220 x 6.150 meters
Price: $47,110.37
House 2
Color: YELLOW
Dimensions: 58.910 x 122.020 x 5.860 meters
Price: $18,528.03

House 3
Color: PINK
Dimensions: 48.990 x 76.580 x 4.430 meters
Price: $82,816.09

House 4
Color: PINK
Dimensions: 44.860 x 45.260 x 7.300 meters
Price: $12,422.44

House 5
Color: PINK
Dimensions: 191.550 x 172.760 x 6.020 meters
Price: $31,598.59

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

2
Enter the number of the house to buy:
5
John Smith is now a proud homeowner!
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

1
House 1
Color: YELLOW
Dimensions: 184.490 x 105.220 x 6.150 meters
Price: $47,110.37

House 2
Color: YELLOW
Dimensions: 58.910 x 122.020 x 5.860 meters
Price: $18,528.03

House 3
Color:PINK
Dimensions: 48.990 x 76.580 x 4.430 meters
Price: $82,816.09

House 4
Color:PINK
Dimensions: 44.860 x 45.260 x 7.300 meters
Price: $12,422.44

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

3
John Smith has sold their house to the market!
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

1
House 1
Color:YELLOW
Dimensions: 184.490 x 105.220 x 6.150 meters
Price: $47,110.37

House 2
Color:YELLOW
Dimensions: 58.910 x 122.020 x 5.860 meters
Price: $18,528.03

House 3
Color:PINK
Dimensions: 48.990 x 76.580 x 4.430 meters
Price: $82,816.09

House 4
Color:PINK
Dimensions: 44.860 x 45.260 x 7.300 meters
Price: $12,422.44

House 5
Color:PINK
Dimensions: 191.550 x 172.760 x 6.020 meters
Price: $31,598.59

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

5
Goodbye!

Enter your name:
Louise Oriondo
Enter your age (in years):
21
Enter the amount of money you have:
123.69
You are:
Name: Louise Oriondo
Age: 21 years old
Cash: $ 123.69

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

1
House 1
Color:RED
Dimensions: 99.660 x 193.180 x 5.020 meters
Price: $7,379.03

House 2
Color:YELLOW
Dimensions: 39.770 x 38.270 x 9.010 meters
Price: $30,116.95
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

2
Enter the number of the house to buy:
5
Louise Oriondo cannot afford this home.
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

3
Louise Oriondo has no home to sell.
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

5
Goodbye!

Enter your name:
Jaicob Stewart
Enter your age (in years):
20
Enter the amount of money you have:
250000
You are:
Name: Jaicob Stewart
Age: 20 years old
Cash: $ 250,000.00

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

2
Enter the number of the house to buy:
5
Jaicob Stewart is now a proud homeowner!
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

2
Enter the number of the house to buy:
12
Invalid house number
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

2
Enter the number of the house to buy:
1
Jaicob Stewart is already a homeowner!
Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
Enter your name:
Becca Burns
Enter your age (in years):
27
Enter the amount of money you have:
40500.12
You are:
Name: Becca Burns
Age: 27 years old
Cash: $ 40,500.12

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

Becca Burns has no home to sell.

Pick an option:
1. List houses for sale
2. Buy a house
3. Sell your house.
4. Show your profile.
5. Exit

Goodbye!