CS-152: Computer Programming Fundamentals in Java

Neal Holtschulte

June 25, 2013
Classes, Objects, and Instances

A blueprint for a house design is like a class. All the houses built from that blueprint are like objects of that class. A given house is like an instance.

Source: http://stackoverflow.com/questions/3323330/difference-between-object-and-instance
Objects

Objects hold data and perform actions. Objects in a program can represent real world objects or abstractions.

- Car
- Person
- Triangle
- Point
- List
- Message
Classes are definitions of objects.

Classes are like blueprints.
Classes

Classes are also like contracts that the object must satisfy in order to be a member of the class.
Classes describe the data an object holds and what the object can do (and what can be done to it).
Classes

Objects are for organization.

They also provide a concrete metaphor to make programs easier to understand.
Classes

Objects combine related data and actions in one place.
Another reason to (generally) keep instance variables private is because the variables should be related to each other and changing one might have an effect on another.
To Eclipse...

Go through Lab 5 in Eclipse.
Each Java Class goes in a separate .java file with very few exceptions that we won't address in this class.

Instance variables of class definitions should usually be private.

A class need not have instance variables. We have used many classes without instance variables so far.
Objects must be created with the “new” operator.

new is an operator that takes an object constructor as input, creates the object in memory, and returns the memory address of the object.

```java
Scanner in = new Scanner(System.in);
int [] xs = new int[10];
Vehicle car = new Car();
```
Inside methods you may see a variable name preceded by the keyword "this".

"this" is an optional keyword that refers to the object that the method is being called on.

So this.number = 7; means that the object whose method is being called will have its number instance variable set to 7.
public class Person {

    private String name;

    public Person() {
        name = "";
    }

    public Person(String name) {
        this.name = name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
Instance variables are variables defined by a class definition. Every instance of the object defined by the class has its own copy of the instance variables that are accessible to the methods of the instance.

```java
Person me = new Person("Neal");
Person myAdvisor = new Person("Melanie");
```
Local variables are variables declared within a method. They are not visible or accessible outside of the method. Different methods of the same object can declare local variables of the same name without affecting each other.
Instance variables should be private

Object instance variables (aka fields or attributes) should be private unless there is a good reason to make them public.

Getters and setters should be used to set and get instance variables.
Why?

- generally considered good practice
- protect your code from others
- save your future self work
- privacy settings are a form of documentation
- hide information
Consider the following setter

```java
public class Rectangle{
    private int width;
    private int height;
    private int area;

    public void setWidth(int width){
        this.width = width;
        area = this.width * height;
    }
}
```
If width was public

```java
int width = 5;
int height = 3;
Rectangle a = new Rectangle(width, height);
a.setWidth(10);
a.width = 8; //This would cause problems
```
In class assignment

- Write a method heading for each method
- Write pre and post conditions for each method
- Write some Java statements that test the class
- Implement the class
As a class

Consider a class MotorBoat with attributes:

- capacity of the fuel tank
- amount of fuel in the tank
- maximum speed of the boat
- current speed of the boat
- efficiency of the boat’s motor
- distance traveled

As a class

Consider a class MotorBoat with methods to:

- change the speed of the boat
- drive the boat at the current speed for a certain amount of time
- refuel the boat with a certain amount of fuel
- access the amount of fuel in the boat
- access the distance traveled so far

Source: Java an introduction to problem solving and programming by Savitch. Page 359, question 7.
As a class

MotorBoat

The boat has efficiency $e$. The amount of fuel used when traveling at speed $s$ for time $t$ is $e \times s^2 \times t$. The distance traveled in that time is $s \times t$.

In class assignment

- Write a method heading for each method
- Write pre and post conditions for each method
- Write some Java statements that test the class
- Implement the class
Homework

Consider a class PersonAddress that represents an entry in an address book. PersonAddress has attributes:

- first name
- last name
- email address
- telephone number

Source: Java an introduction to problem solving and programming by Savitch. Page 359, question 8.
Consider a class PersonAddress with methods to:

- Access each attribute
- Change the email address
- Change the telephone number
- Test whether two instances are equal based solely on name

Constructors

A constructor is a special method that is called when using the "new" operator.

A constructor is meant to assist in initializing an object.

Constructors have the same name as the class.

Source: *Java an introduction to problem solving and programming* by Savitch.
Constructors

You can have multiple constructors of the same name. These will be distinguished by different parameters. This can be useful, for example when you want the option to have a default constructor and a parameterized constructor. Source: Java an introduction to problem solving and programming by Savitch.
public class Person {
    private String name;

    // Default constructor
    public Person() {
        name = "No Name";
    }

    // Another constructor
    public Person(String name) {
        this.name = name;
    }
}
Constructors can call other constructors

```java
public class Person {
    private String name;

    // Default constructor
    public Person() {
        this("No Name"); // Call to the other constructor
    }

    // Another constructor
    public Person(String name) {
        this.name = name;
    }
}
```
There are lots of good tutorials.

- [http://fricke.co.uk/Teaching/CS152/OOP.pdf](http://fricke.co.uk/Teaching/CS152/OOP.pdf)
- [docs.oracle.com/javase/tutorial/java/javaOO/](http://docs.oracle.com/javase/tutorial/java/javaOO/)
  - [docs.oracle.com/javase/tutorial/java/javaOO/objects.html](http://docs.oracle.com/javase/tutorial/java/javaOO/objects.html)
  - [docs.oracle.com/javase/tutorial/java/javaOO/classes.html](http://docs.oracle.com/javase/tutorial/java/javaOO/classes.html)
- [www.tutorialspoint.com/java/java_object_classes.htm](http://www.tutorialspoint.com/java/java_object_classes.htm)