Object Oriented Programming

Week 1 Part 1
An introduction to Java, Objects and JUnit

Object Oriented Programming with Java

Syllabus

- This class teaches Object Oriented Programming using Java
- We will focus on the practical aspects of OO programming
  - We will use Scrum and Test Driven Development
  - We will use the Eclipse IDE augmented with
    - JUnit for unit testing
    - Git for version control.

Lectures

- Lectures will introduce new material
- You are required to attend lecture
  - Roll will be taken and you will be detained at the final if you miss too many classes. (Please do not let that happen.)
- Use the time in lecture to gain the knowledge you will need for the tutorials and labs.
  - Lectures give us the opportunity to efficiently disseminate the information you will need.

Tutorials

- Before the tutorial you will be given a list of questions pertaining to the lectures from the previous week.
  - These questions give you the opportunity to test your understanding.
- Objective questions on the test will be selected from the tutorial questions.
  - If you understand the tutorial questions, you will do well on the objective part of the tests

Labs

- Labs make the theoretical knowledge presented in lectures and tutorial practical.
- Labs are run as demos
  - You will show the teacher the programming you completed over the week.
  - You work on your programming for the following week while waiting your turn.
  - Most of the work you do will be done outside of class.

Lab Resources

- You will be given an account on an Amazon Web Services (AWS) server on which you should keep your work.
- You will work on your own computer.
- Your computer should be set up with Eclipse Mars running the Java Development Environment including JUnit and eGit.
  - If you do not have this set up see your teacher as soon as possible for help getting it set up.

Lab Resources (cont.)

- The Lab itself is yours: keep it running well.
  - There is a report sheet in the lab where you can note any problem with the computers.
  - Keep watch for anyone who might damage your lab
- You may use the lab whenever the University is open.
  - If there is no class, go to the server room and sign up to use it.
  - You may let others use the lab, but you will be asked if anything happens.
  - If others are using the lab and you want to leave, ask one of them to sign up to watch it.
  - Check with the teachers to see if you can use the lab when there is a class.

Learning to Program

- You learn to program only by programming.
  - Lectures and Tutorials can get your started and give you things to teach you tools and tricks.
  - But only through writing programs will you learn to do it.
- Computer Science is more than programming, but you cannot do it without programming.
  - Literature is more than letters, but if you do not know the letters, Literature is inaccessible.

Today's Lecture

- Java syntax is mostly like C
- Defining Object Oriented programming
- Example of a Java Class
- Java Formatting
- Java Packages
- Java main()

Java

Java is based on C

- You already know most of Java.
- The difference is that Java is based on Objects and Classes.
  - These differences have some impact on other aspects, but most of what you know from C will transfer.
- However, because Java is object oriented, it leads you to think differently
  - This course will help you learn to think in an object oriented manner.

Java Output differs from C

- C
  - #include <stdio.h>
  - printf(“...”, var1, var2 …)
- Java
  - import java.io.*
  - System.out.println(“...”);
  - System.out.printf(“...”, var1, var2, …);

You can add strings in Java

- int a = 5; “Here are ” + a + “ things.”
  - Produces “Here are 5 things.”
- Adding strings together is how you build up messages to print in Java.
- string name = “Nat”;

  System.out.println(“Hello ” + name);
  - Writes “Hello Nat” on the console.
- System.out.println always ends with a newline.

Input is different in Java

- C
  - #include <stdio.h>
  - int a; scanf(“%d”, &a);
- Java
  - import java.io.*
  - int a; a = StdIn.readInt();

Java has references

- Java references are like pointers except
  - All objects are stored in references.
  - References are created with the keyword new
  - References are automatically deleted when no longer needed.
  - You can only refer to elements of the object referred to only using the dot notation. (i.e., there is no pointer arithmetic)
  - Since objects are always stored in references, when you pass an object, you can access the elements of the object passed.

Minor differences

- C “NULL” is “null” in java
- In Java, you can define variables and methods as “private” meaning they can only be used in the object, or “public” meaning they can be used anywhere.
- In Java, you can use the comment character “//” to turn a line into a comment. You can still use “/* */” to surround arbitrary text.

Java Objects

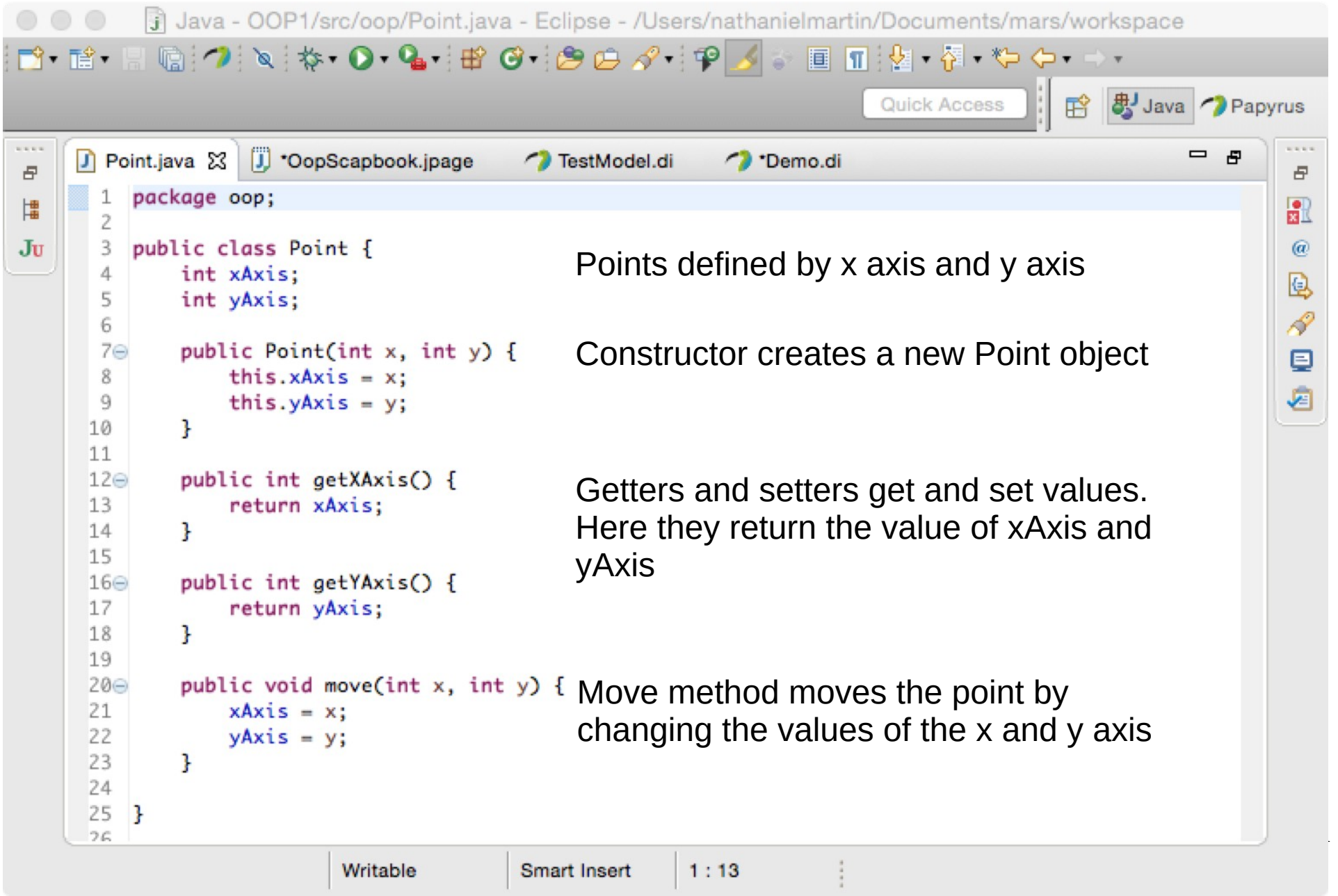
What is Object Oriented?

- Programs are build from *objects*
  - One can reason able programs as the interaction of objects
- Objects are defined by
  - Properties
    - Represented by *fields*.
      - Fields are like constants and variables in C.
  - Behaviors
    - Represented by *methods.*
      - Methods are like functions in C

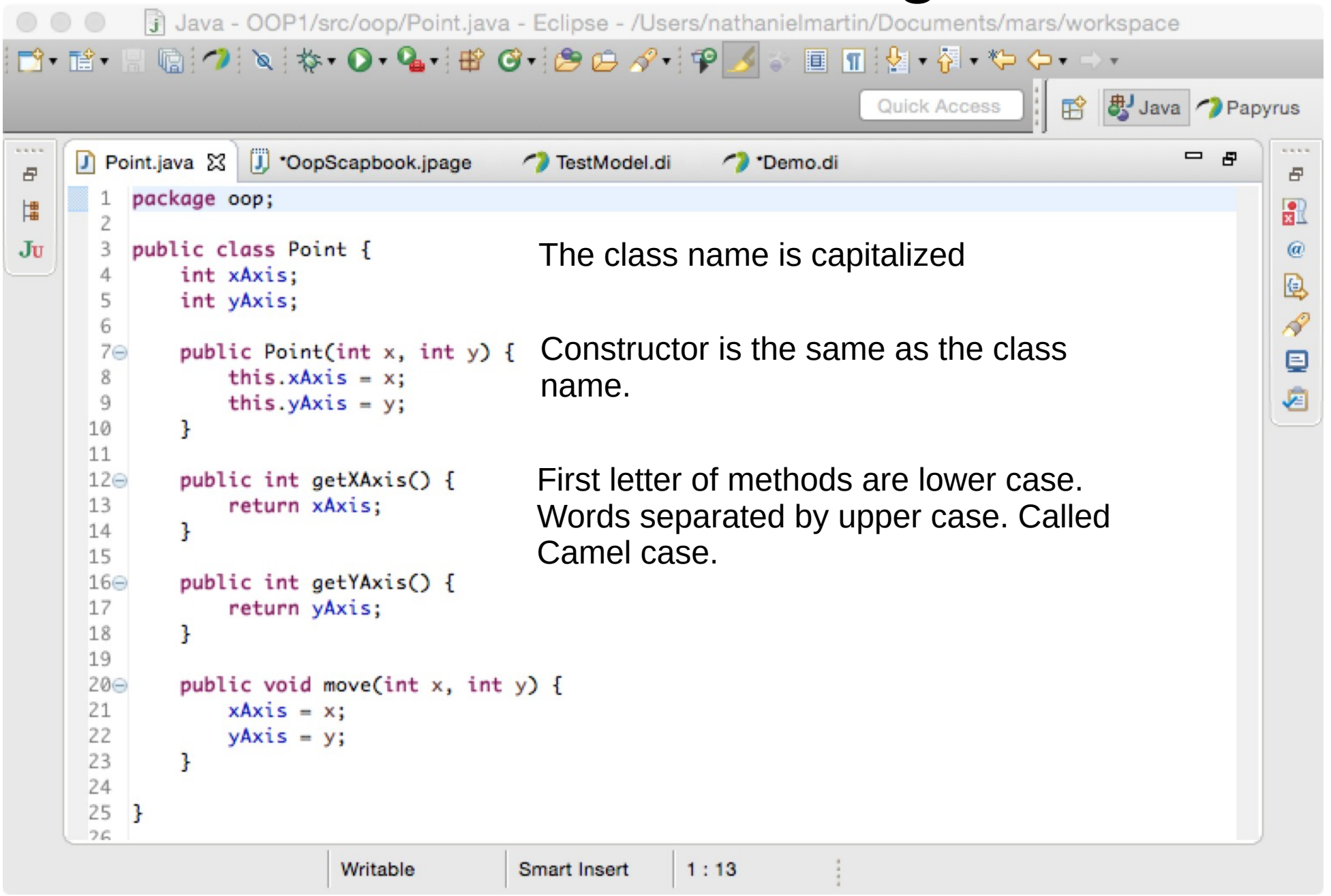
Objects are created from classes

- A *class* defines are kind of objects.
- An object is created from a class by defining its properties.
  - I.e., setting its *instance variables*
  - A *constructor* creates new objects from a class
- All objects of a particular class have the same behaviors.
  - i.e., they all have the same methods

Example Java Class (Point)



Java Formatting



The class name is capitalized

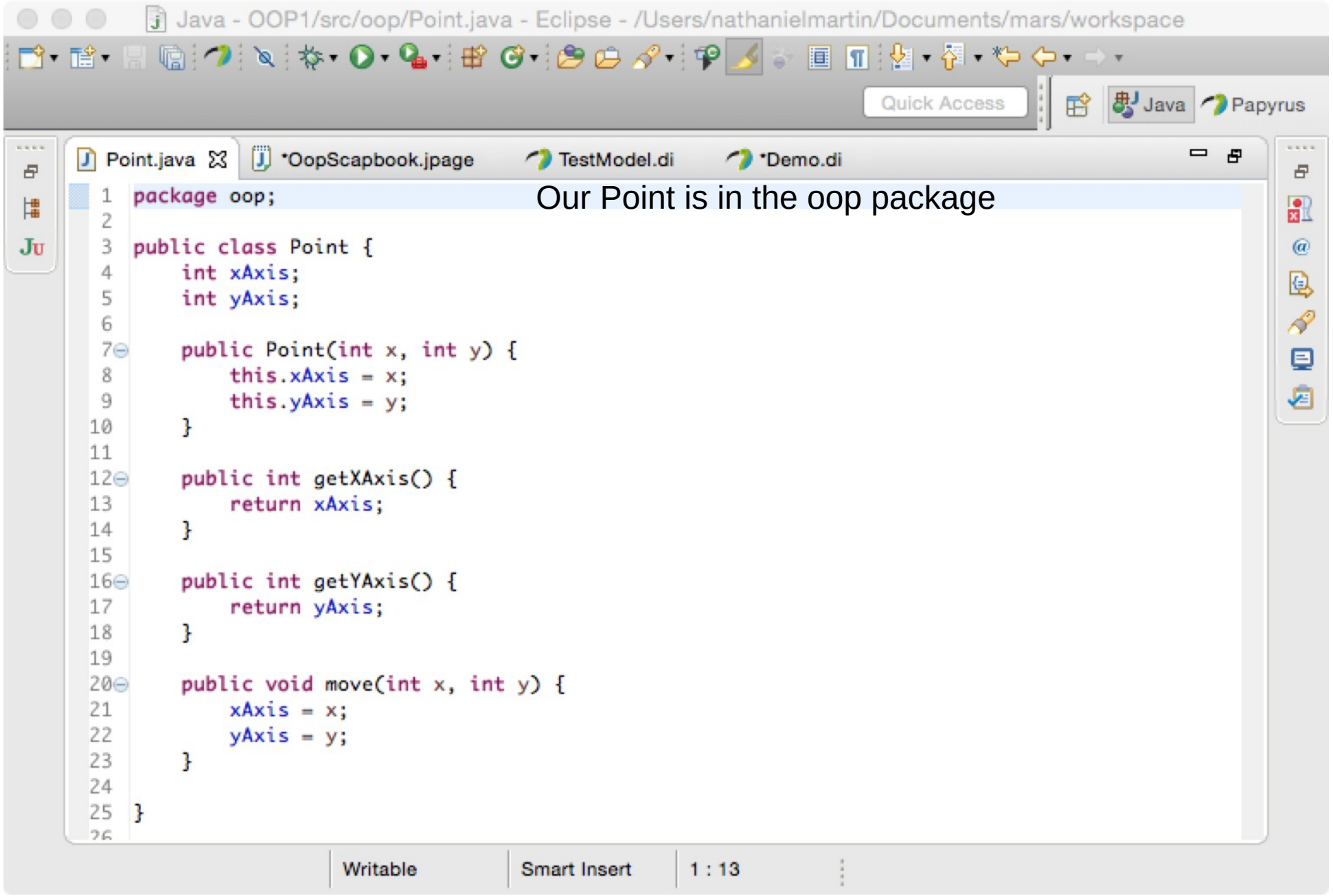
Constructor is the same as the class name.

First letter of methods are lower case. Words separated by upper case. Called Camel case.

Java Packages

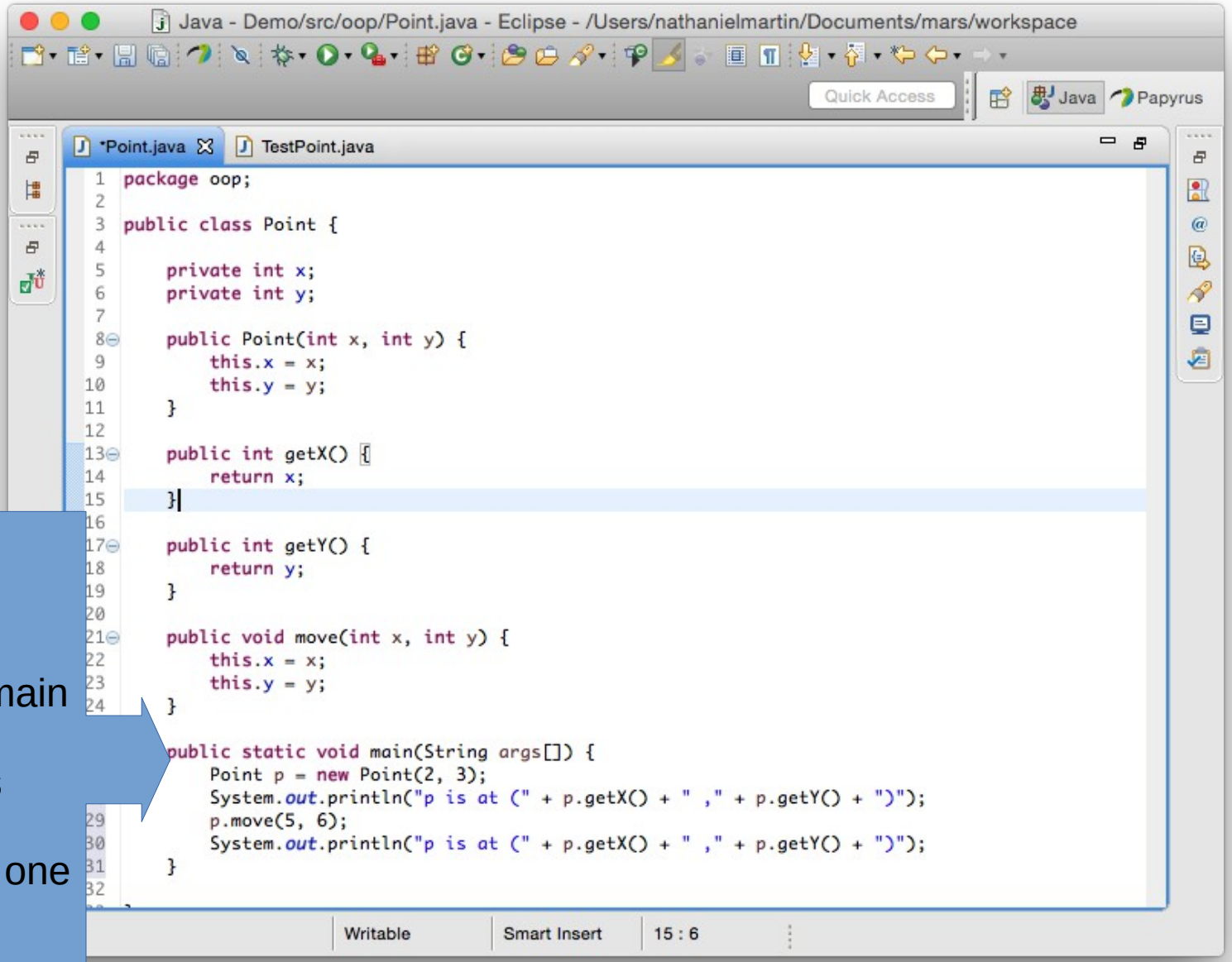
- Packages define a *name space*
  - Name spaces keep similar names separate.
- For Example
  - The AWT package has a Point class
    - Theirs is java.awt.Point
  - Our oop package has a Point class
    - Ours is oop.Point
- We can *import* names from one package into another.

Java Packages




```java
package oop;

public class Point {
    int xAxis;
    int yAxis;

    public Point(int x, int y) {
        this.xAxis = x;
        this.yAxis = y;
    }

    public int getXAxis() {
        return xAxis;
    }

    public int getYAxis() {
        return yAxis;
    }

    public void move(int x, int y) {
        xAxis = x;
        yAxis = y;
    }

}
```

Java main location



Like C
- Execution starts in main

Unlike C
- Each multiple mains
- One per class
- Must indicate which one

Java main details

```
public static void main(String args[]) {

    Point p = new Point(2, 3);

    System.out.println("p is at (" + p.getX() + " ," + p.getY() + ")");

    p.move(5, 6);

    System.out.println("p is at (" + p.getX() + " ," + p.getY() + ")");
}
```

public: visible outside class
static: one per class
String args[]: array of strings

Create a new Point
Stored in reference p
xAxis = 2; yAxis = 3

Output: p is at (2, 3)

Move to (5, 6)

Output: p is at (5, 6)

Executing the Class

Main function

Output

Point.java | TestPoint.java

```
19      }
20
21      public void move(int x, int y) {
22          this.x = x;
23          this.y = y;
24      }
25
        public static void main(String args[]) {
            Point p = new Point(2, 3);
28          System.out.println("p is at (" + p.getX() + " ," + p.getY() + ")");
29
30          p.move(5, 6);
31          System.out.println("p is at (" + p.getX() + " ," + p.getY() + ")");
32      }
33
34  }
35
```

Problems | Javadoc | Declaration | Search | Console | Tasks

<terminated> Point [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/bin/java (Jun 8

```
p is at (2 ,3)
p is at (5 ,6)
```