Unit 1A

Why study computer science if you don't want to be a programmer?

Computer science is the study of *process*. Process is important to nearly every field, from business to science to medicine to law. Knowing process formally is important to everyone.

> [1]Our nation's current trajectory points to a lasting digital era, and we'll need people who can think like software engineers and network architects, whether they are writing an app or solving resource distribution problems in a third-world setting — or doing both at the same time.
>
> Teaching CS is about giving them the thinking skills that will help them become proactive learners and citizens — as opposed to just consumers and denizens — in a world that's increasingly influenced by the manipulation of the digital bit.

Why Java?

Java is platform independent, free, and used in a variety of applications such as web pages, smart phone apps, animations, and business applications. It is truly **object oriented**, and the concepts you learn with Java can be applied to all other areas of computer science.

Also, that is the language of choice for the AP course! ☺

Hello World

Open Netbeans, and choose Java SE Applications under Demos and Tutorials. From there, choose the Java Quick Start Tutorial. This is the iconic "Hello World" app!

[1] https://www.iste.org/explore/articledetail?articleid=216

About Object Orientation

The following is a really good example of what objects are, and how they might interact. [2]

Java is an object-oriented programming language. This means that the focus is on objects (who) as well as procedures (what). Objects are persons, places, or things that do the action in a situation or are acted upon.

An example might help you to understand what focusing on the objects means. When customers enter a restaurant a greeter will welcome them to the restaurant and show them to their table. A waiter will take the order and bring the drinks and food. One or more chefs will cook the food. The waiter will create the bill and give it to the customers. The customers will pay the bill.

How many people does it take to get a customer fed in a restaurant? Well, you need at least a customer, greeter, waiter, and a chef. What other things are doing action or being acted upon? We mentioned order, table, drink, food, and bill. Each of these are objects. The objects in this situation are working together to feed the customer.

What types of objects are they? We have given names to each thing we mentioned: customer, waiter, food, and so on. The names we gave are how we classify these objects. You probably know what we mean by a customer or food. But the computer doesn't know what we mean by these things. The way that we get the computer to understand what we mean is by defining a *class*. A class in Java tells the computer what data we expect objects of that class to have and what they can do. We would expect that food would have a name, a price, and a way to prepare it. We would expect that a customer would know what they can afford to pay and how to pay a bill.

Every object of the same class will have the same skills or operations (things it can do) and data or variables (things it knows about). For example, each object of the order class should know which customer placed that order and what food is in the order. An object of the chef class should know how to prepare the food.

There can be many objects of the same class. A restaurant might have three chefs, ten waiters, two greeters, and 100 food objects on its menu. On a given day and time it might have 100 customers.

Why don't restaurants just have one type of employee? One person could greet the customers, take the orders, cook the food and deliver the food. That might be okay if there is only one customer but what about when there are many customers? You can imagine that one person wouldn't be able to handle so many tasks and food would get burnt, orders would take too long to fill, and customers wouldn't be happy. Restaurants break the tasks into different jobs so that they can be efficient and effective. Object-oriented programs also try to distribute the tasks to be done so that no one object does all the work. This makes it easier to maintain and extend the program. It can also make the program more efficient.

[2] Introduction to Computing & Programming with Java by Mark Guzdial and Barbara Ericson,2007

Unit 1A Questions

Objects and Procedures

Under the section 'About Object Orientation' there is an excerpt for you to read. After reading it, answer the following questions in reference to the excerpt.

1. The focus of OOP (Object Oriented Programming) is on objects and Methods.
2. Objects are person, place or thing.
3. From the example, a *waiter* is an object/procedure (choose one) and it can do
   a. object
   b. serve,give the bill,take money,bring food
4. How many staff (from the example) does it take to feed a customer? (4)3 Each of these can be considered an object.
5. The things that the staff can do are considered Methods.
6. To create objects in Java (or any other language) we define a Class that tells the computer what data the object has and what it is capable of doing.
7. Why would a restaurant break the tasks up for different staff to perform to be efficient and effective.

Note: Although this example refers to *procedures*, we will be calling them *methods* most of the time.

8. Try this…think of an object you are familiar with…a car, an instrument, whatever. Now, what characteristics does it have? (color, age, size, shape, model, etc.) Lastly, what can it do? List these below.

Object: car

Characteristics:
color, model,size-----PROPERTIES-ADJECTIVES

Abilities:
drive
brake
Horn
Lights-----METHODS-VERBS

Unit 1B

Software

In order to program in Java, we need the JDK installed on the machine and some sort of programming environment to code in. We are going to use the Netbeans IDE which can come packaged with Java, so installing is very easy. There is a copy here G:\CP\EVERYONE\Klimack\Java and AP\Java Software for you to take home.

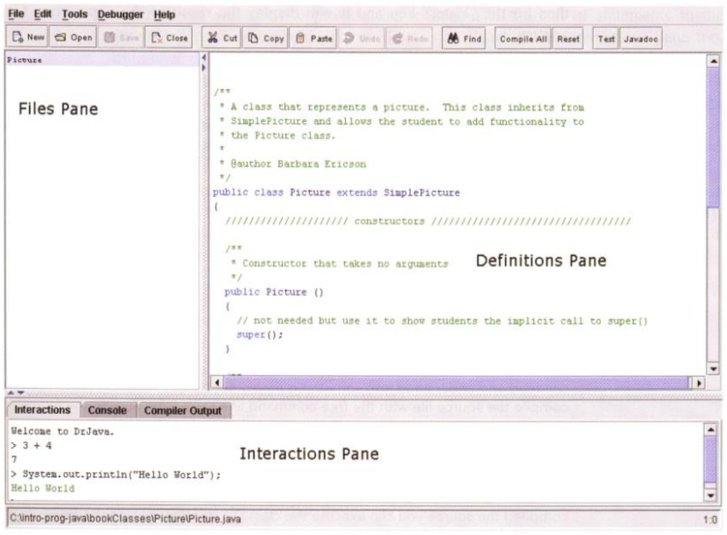
Java with Netbeans has been installed for you to use. Netbeans is an IDE that has some intellisense, and will help you with your transition from VB.

Dr. Java is a another free, easy to use development software. You simply run the exe and begin coding. You will likely have to set up some paths in order for it to work. Choose **Edit > Preferences> Add (**below **Extra Classpath)** and point to any classes you copy and paste to use. Also add a Classpath to the **grdraw.jar file.** I suggest it because it has a built-in java app runner that works well.

For video help installing and using Dr. Java:

http://www.cs.rice.edu/~javaplt/drjava/videos/SelfTeachingJavaVideo2.shtml

http://www.cs.rice.edu/~javaplt/drjava/videos/SelfTeachingJavaVideo3.shtml


File Edit Tools Debugger Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile All Reset Test Javadoc
Picture
Files Pane
Definitions Pane
Interactions Console Compiler Output
Interactions Pane
C:\intro-prog-java\bookClasses\Picture\Picture.java
1:0

```
/**
 * A class that represents a picture.  This class inherits from
 * SimplePicture and allows the student to add functionality to
 * the Picture class.
 *
 * @author Barbara Ericson
 */
public class Picture extends SimplePicture
{
  ///////////////////// constructors //////////////////////////////////

  /**
   * Constructor that takes no arguments
   */
  public Picture ()
  {
    // not needed but use it to show students the implicit call to super()
    super();
  }
```

```
Welcome to DrJava.
> 3 + 4
7
> System.out.println("Hello World");
Hello World
```

You also need to set the **Interactions Working Directory** to h:\java\projects or similar. The **h:** is the important part.

If you don't, some of your programs won't work.

Unit 1C

Java Basics

For starters, simply type some commands into the interactions pane. Write your results in the space provided.

>34 + 56	90
>26 – 3	23
>3 * 4	12
>4/2	2
>Math.pow(2,3)	8

What do you think the difference is in the next 3? int division...Decimal

>7 / 2	3
>7.0 / 2	3.5
>7 / 2.0	3.5

Remember your order of operations.

> 2 + 4 * 3	14
>(2 + 4) * 3	18

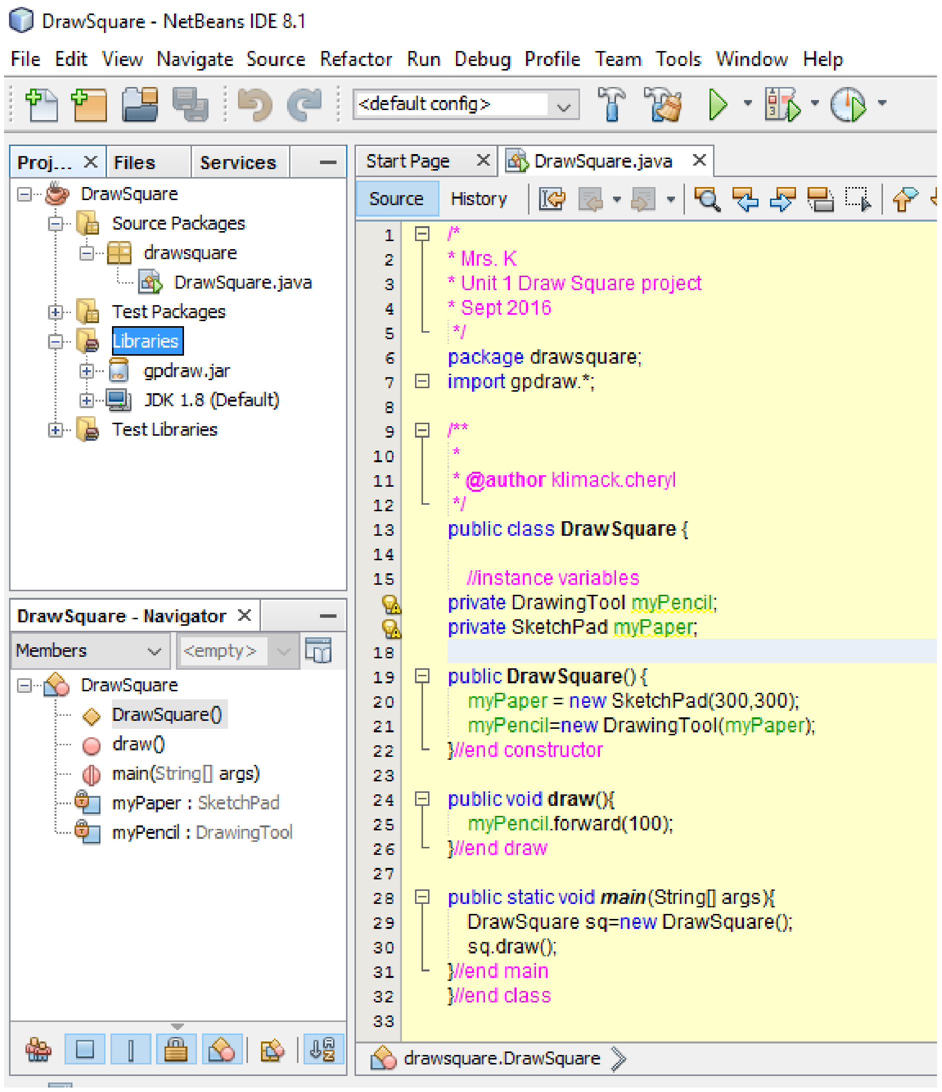
Can you guess what % means?

>5 % 2	1
>6 % 2	0
>7 % 2	1
>8 % 2	0
>9 % 3	0
> 10 % 3	1

Unit 1D

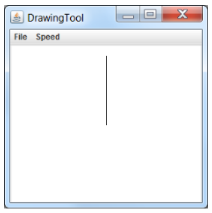
We are going to use the imported library **gpdraw.jar** to draw using Java code. Copy and paste this file to your account into your Netbeans folder. The file is found at G:\CP\EVERYONE\Klimack\Java and AP\Java Software

- NetBeans: After creating your DrawSquare project, right click the **Libraries** folder and Add Jar/Folder, navigating your way to the new jar file.
- Add the following code.



```java
/*
 * Mrs. K
 * Unit 1 Draw Square project
 * Sept 2016
 */
package drawsquare;
import gpdraw.*;

/**
 *
 * @author klimack.cheryl
 */
public class DrawSquare {

    //instance variables
private DrawingTool myPencil;
private SketchPad myPaper;

public DrawSquare() {
    myPaper = new SketchPad(300,300);
    myPencil=new DrawingTool(myPaper);
}//end constructor

public void draw(){
    myPencil.forward(100);
}//end draw

public static void main(String[] args){
    DrawSquare sq=new DrawSquare();
    sq.draw();
}//end main
}//end class
```

Go ahead and run it. You should see:

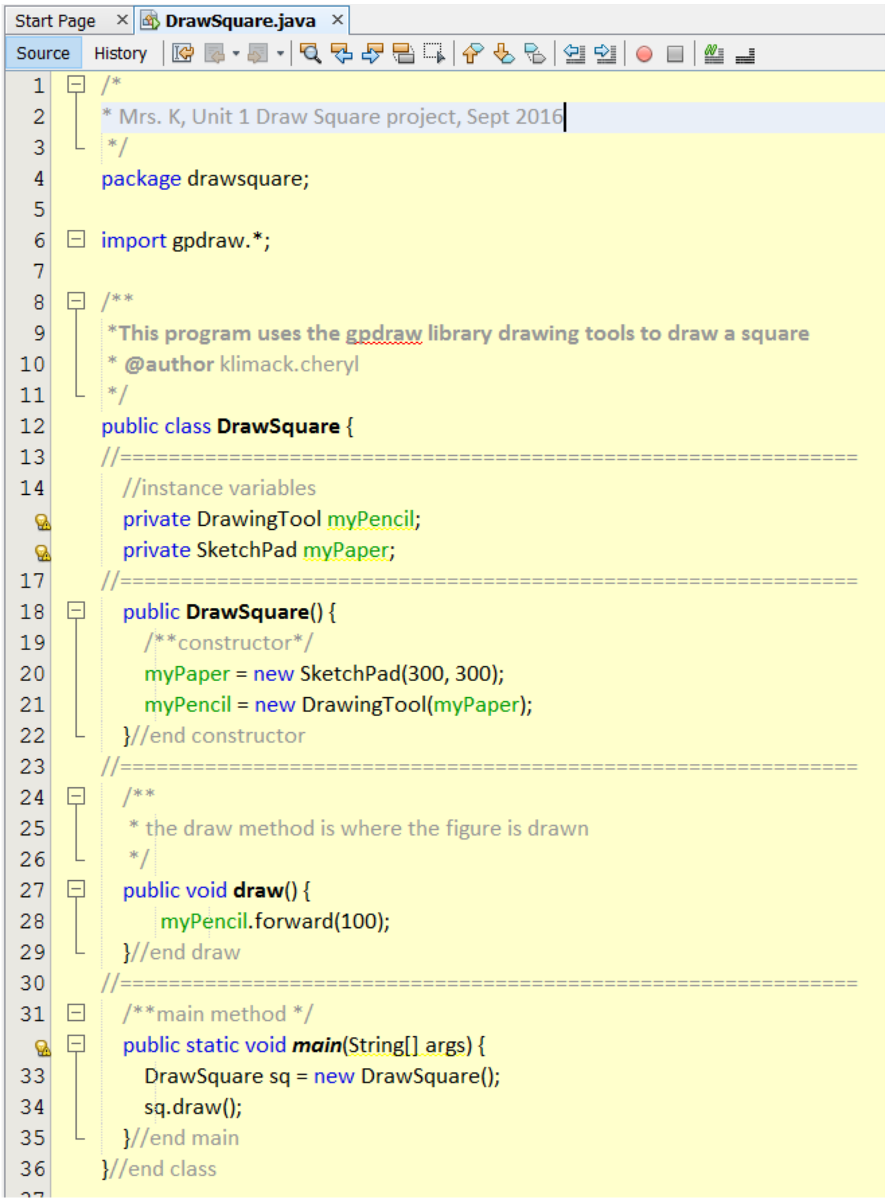


Adding Comments

Comments make life simpler for anyone reading your code, including yourself later on down the road when it isn't fresh in your mind. At the top of each project, include comments similar to:

```
/* YOUR NAME
 * PROJECT NAME
 * DATE
 */
```

Use separating lines between methods in your program. Single line comments begin with //. Each method should have at least one comment.
http://journals.ecs.soton.ac.uk/java/tutorial/getStarted/application/comments.html



```java
/*
 * Mrs. K, Unit 1 Draw Square project, Sept 2016
 */
package drawsquare;

import gpdraw.*;

/**
 *This program uses the gpdraw library drawing tools to draw a square
 * @author klimack.cheryl
 */
public class DrawSquare {
//==============================================================
   //instance variables
   private DrawingTool myPencil;
   private SketchPad myPaper;
//==============================================================
   public DrawSquare() {
      /**constructor*/
      myPaper = new SketchPad(300, 300);
      myPencil = new DrawingTool(myPaper);
   }//end constructor
//==============================================================
   /**
    * the draw method is where the figure is drawn
    */
   public void draw() {
        myPencil.forward(100);
   }//end draw
//==============================================================
   /**main method */
   public static void main(String[] args) {
      DrawSquare sq = new DrawSquare();
      sq.draw();
   }//end main
}//end class
```

Errors

Java is **case sensitive**. You must be perfect when typing. Dr Java does not autocorrect for you.

Indenting doesn't matter to the computer, but it definitely matters to whomever is trying to read or debug your code.

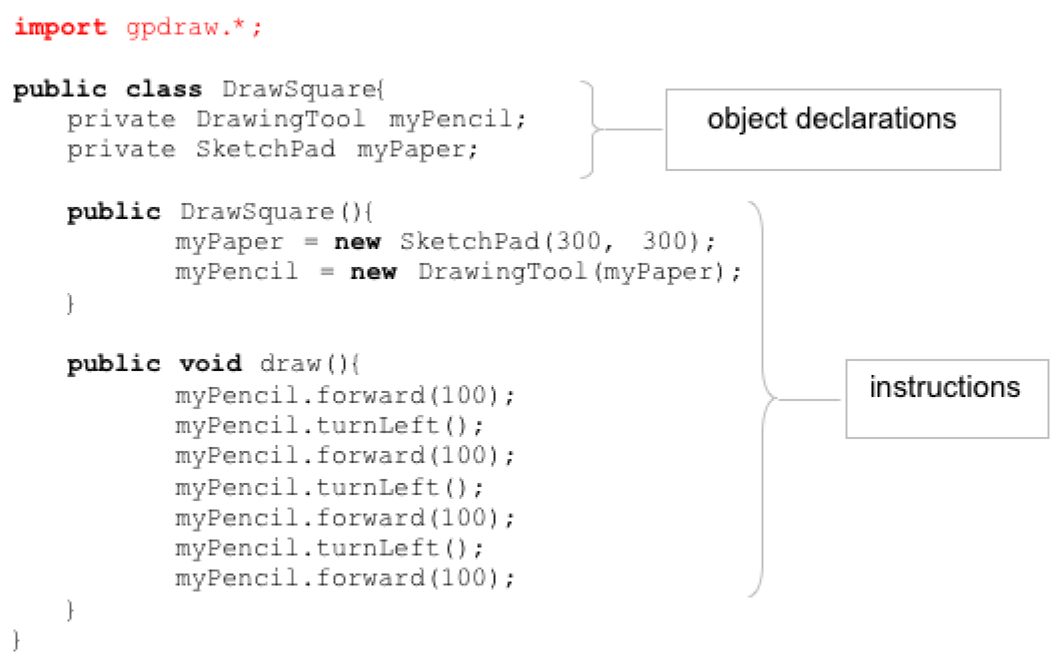
Syntax errors will happen with mistakes in grammar. These are found when the program is **compiled**.

Runtime errors will happen if you code something that doesn't work, such as dividing by 0 or forgetting an argument like the 100 in the code for **myPencil.forward**. These are found when the program is **run**.

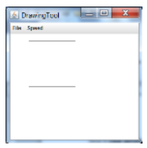
Logic errors are the hardest to find. This is a flaw in the program's logic…the planning that went into the program. These are found by **tracing** your program and using debugging techniques.

More code

Add code to the DrawSquare draw() method to make it look like the following. Run it.



```
import gpdraw.*;

public class DrawSquare{
    private DrawingTool  myPencil;
    private SketchPad  myPaper;

    public DrawSquare(){
            myPaper  = new SketchPad(300,  300);
            myPencil  = new  DrawingTool(myPaper);
    }

    public void draw(){
            myPencil.forward(100);
            myPencil.turnLeft();
            myPencil.forward(100);
            myPencil.turnLeft();
            myPencil.forward(100);
            myPencil.turnLeft();
            myPencil.forward(100);
    }
}
```

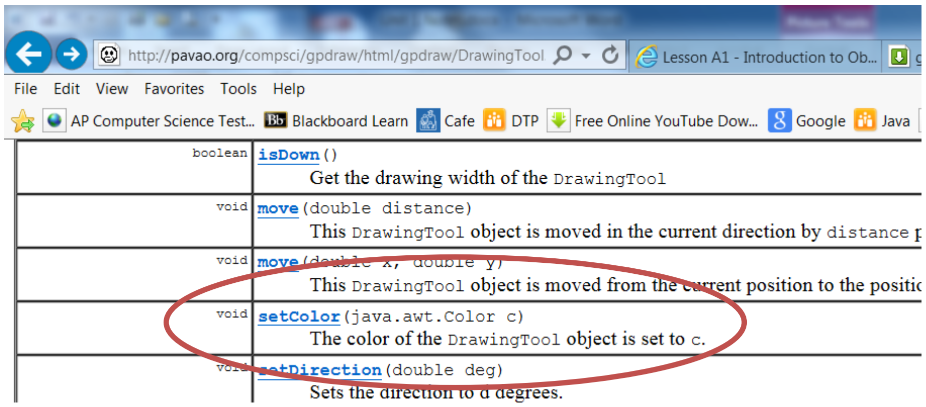
You should get: 

Getting Help

Search for *java gpdraw*. This will bring up the link to the java api for the gpdraw library.

http://pavao.org/compsci/gpdraw/html/gpdraw/DrawingTool.html

It will look confusing for now, but soon you will be used to it and this is your main source of java help. The API lists the properties (**field variables**) that you can use, such as the xPos and yPos of the pencil in our program. It also lists all the **methods** that come with the library. Look through the list for a method to change the color of our square.


http://pavao.org/compsci/gpdraw/html/gpdraw/DrawingTool
Lesson A1 - Introduction to Ob...
File Edit View Favorites Tools Help
AP Computer Science Test... Blackboard Learn Cafe DTP Free Online YouTube Dow... Google Java

```
boolean isDown()
            Get the drawing width of the DrawingTool
   void move(double distance)
            This DrawingTool object is moved in the current direction by distance p
   void move(double x, double y)
            This DrawingTool object is moved from the current position to the positic
   void setColor(java.awt.Color c)
            The color of the DrawingTool object is set to c.
   void setDirection(double deg)
            Sets the direction to d degrees.
```

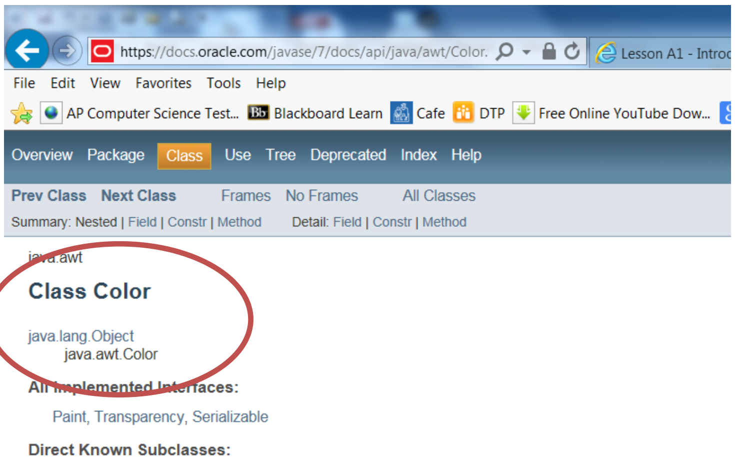
To use this, note that it takes a **color argument.** Here is an example:

```
myPencil.setColor(Color.RED);
```

To use the Color class, however, you must import the Color library with this line:

```
import java.awt.Color;
```

Search for *java color*, and look at the api to see how this is indicated.


https://docs.oracle.com/javase/7/docs/api/java/awt/Color.
Lesson A1 - Introc
File Edit View Favorites Tools Help
AP Computer Science Test... Blackboard Learn Cafe DTP Free Online YouTube Dow...
Overview Package Class Use Tree Deprecated Index Help
Prev Class Next Class Frames No Frames All Classes
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.awt

Class Color

java.lang.Object
    java.awt.Color

All Implemented Interfaces:

Paint, Transparency, Serializable

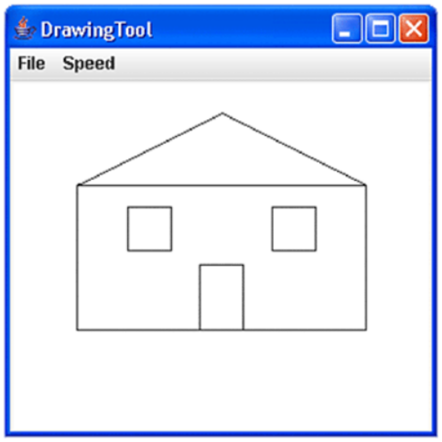
Direct Known Subclasses:

Unit 1D Labs

1. Modify your DrawSquare to have lines of 4 different colors.
2. Draw a circle the touches the 4 sides of the square.

Unit 1D-A Assignment for Marks

Write a program that creates a drawing area of appropriate size (try 500 x 500) and draws a house similar to the one shown below and with these specs:



1. The house should fill up most of the drawing area, ie draw it BIG.
2. The house should be centered horizonally on the screen.
3. The house must have a sloped roof. It can be of any slope or configuration, but not flat.
4. Include at least 1 window and a door.
5. Include something not shown in the figure above of your own design.

Unit 1D-B Assignment for Marks

Write a program that creates a drawing area of appropriate size (try 500 x 500). Your drawing will include as many different methods as you can. It has to be a recognizable image, not random shapes. Each item must have a useful purpose (flower, not an oval in the middle of nowhere). You have a time limit. At the end of the time allotted, you will hand in your drawing and will be marked according to how many different gpdraw methods you used purposefully.

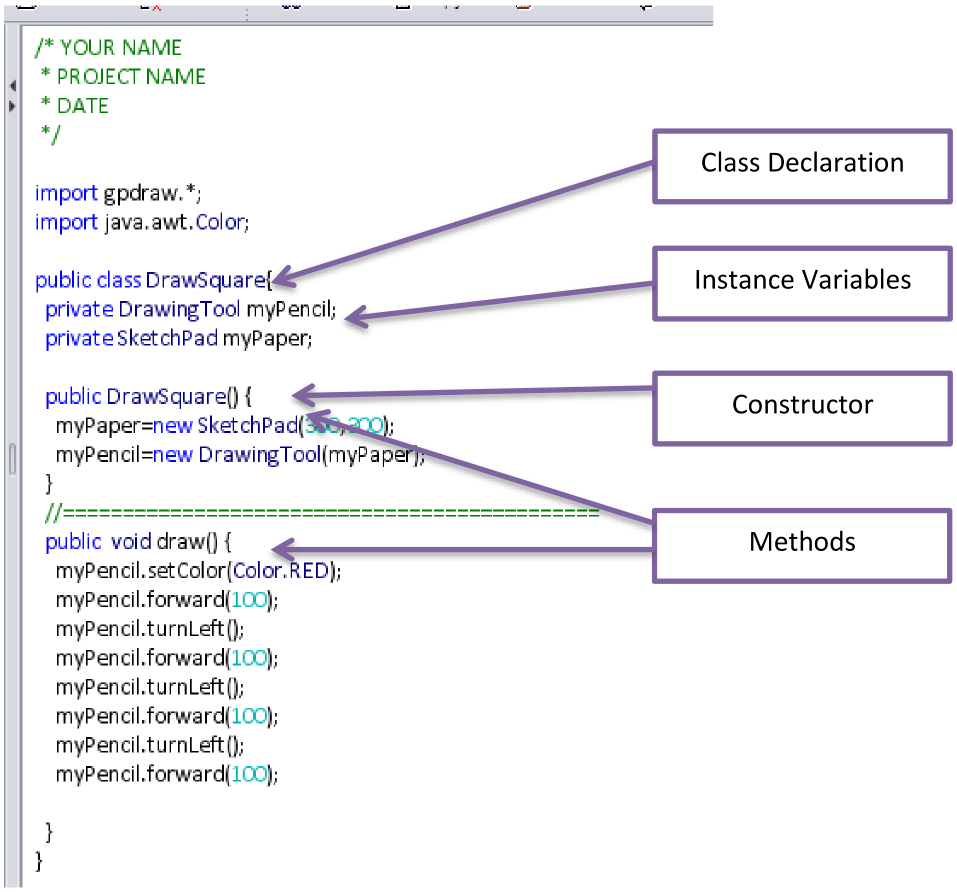
Handing in work
See the document *Assignment Checklist* for each assignment handed in.

- Include your name, project name, and date in the comments at the top of your code as indicated below.
- Copy and paste this project to Klimack Marking. All projects must be contained in their own folder. Stay organized!

Unit 1E OOP Vocab

Use separating lines between methods in your program. Single line comments begin with //.

```
/*  Multi-line comment. This style allows you to write
    longer documentation notes as the text can wrap around to
    succeeding lines.
 */

// Single-line comment.
// This starts at the slashes and stops at the end of the line.

/** Java-doc comment. This is a special type of comment used
    to create APIs and will be discussed more thoroughly
    in a later lesson.
*/
```



```
/* YOUR NAME
 * PROJECT NAME
 * DATE
 */

import gpdraw.*;
import java.awt.Color;

public class DrawSquare{
 private DrawingTool myPencil;
 private SketchPad myPaper;

 public DrawSquare() {
  myPaper=new SketchPad(300,300);
  myPencil=new DrawingTool(myPaper);
 }
 //============================================
 public void draw() {
  myPencil.setColor(Color.RED);
  myPencil.forward(100);
  myPencil.turnLeft();
  myPencil.forward(100);
  myPencil.turnLeft();
  myPencil.forward(100);
  myPencil.turnLeft();
  myPencil.forward(100);

 }
}
```

Class declarations are written in *ProperCase* and begin with ***public class***.

Instance variables are declared outside of any methods, and are declared with ***private** access.* This means that only the class they are declared in can use them. They are nouns. They are named in *titleCase*.

Class variables are similar to *instance variables*, except they are declared with the keywords **private static**.

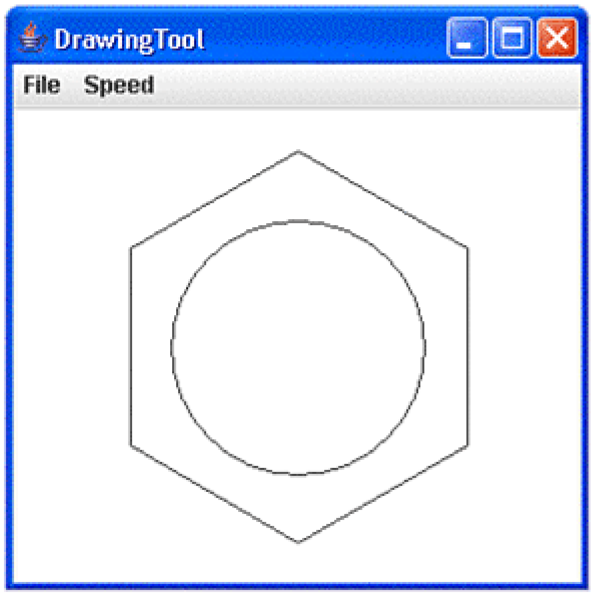
Methods are the actions that the object can perform. They are declared using verbs or action words. They are written in *titleCase* (except for the constructor).

Constructors are special methods with the exact same name as the class. This method is called when the object is 'constructed' or built.

Unit 1E Assignment

Benzene

Benzene is a very common and useful organic compound. Its chemical formula is C_6H_6 and is often drawn in abbreviated symbol form as a circle centered inside of a hexagon. Here's an example:



Assignment:

This program will use the `gpdraw` library. Refer to the `DrawingTool` Class Specifications as described in http://pavao.org/compsci/gpdraw/html/gpdraw/DrawingTool.html. Write a program that creates the benzene ring symbol consisting of a circle centered inside a hexagon. The circle does not touch the outer hexagon, as indicated in the example picture.

Instructions:

1. Include your name as a documentation statement and also a brief description of the program.
2. Ask for instructions on submitting your finished project for marking.

Unit 1F

System.out

Let's walk through creating a simple program with console output.

1. All Java programs start with a class. The { and } mean begin and end. The name of this class is *CompSci.*

```
public class CompSci {
        }
```

2. All console applications require a *main* method to **run**. You don't have to put a main in each object, but the program won't run without one.

```
public class CompSci  {
      public static void main(String[] args){
          System.out.println("Comp Sci!");
      } //end main
} //end class
```

Java is case sensitive.

frog is NOT the same as *Frog.*

Java developers follow a strict set of programming standards to make it easy to follow the code.

The end } should line up with the corresponding code section beginning. Also, it will be helpful to you to add a comment indicating what the ending } is ending, as shown above.

This program should compile and run. You should see output in the interactions pane unless you have syntax errors.

- watch for the semi colons that end a statement
- watch for the case sensitivity of your code
- watch for proper { and } placement
- note that the java file name and the class name must be exactly the same

3. Start a new java file and enter the program you see below. Before running it, try to guess what the output will be.

```
public class One
{
      public static void main(String args[])
      {
            System.out.print("compsci");
            System.out.println();

      }
}
```

4. Add the following lines to the main and guess again before running.

```
System.out.print("compsci");
System.out.print("compsci");
System.out.println();
```

Next, add

```
System.out.println("compsci");
System.out.println();
```

5. Lastly, add

```
System.out.println("compsci");
System.out.println("compsci");
System.out.println();
```

System.out	
Frequently used methods	
Name	**Use**
print(x)	print x and stay on current line
println(x)	Print x and move to the next line down
printf(s,x)	Print x according to s spec s

Unit 1F Lab A

1. Write a program to output your full name on one line, one word at a time.
2. Output a blank line.
3. Output your address, one line at a time.

Your output should look similar to:

```
Your name

123 Anystreet
Sometown, MB
R7A 0X0
```

Escape Sequences

\n, \t, \r, and \b are common escape sequences used with print, println, and printf .

Escape Sequences	
\t	is used to tab over five spaces.
\n	is used to move the cursor down to the next line.
\r	is used to move the cursor to the beginning of the current output line.
\b	is used to backspace one place on the current line.
\\	is used to print a single \.
\”	is used to print a single “.
\’	is used to print a single ‘.

Formatting with printf

printf is a method used to print values on the console window.
For example, printf can be used to set the number of decimal when printing a decimal number.
printf can be used to print any type of data.

Unit 1F Lab B

Enter this program into a new file. Compile and run.

```
public class Two{
     public static void main(String args[]){
          System.out.println("c\tompsci");
          System.out.println("com\tpsci");
          System.out.println("comp\nsci");
     }
}
```

Unit 1F Lab C

Enter this program into a new file. Compile and run.

```
public class Three{
     public static void main(String args[]){
          System.out.println("\\compsci\"/");
          System.out.println("\\'comp\'sci\'/");

          //printf will be used more later
          System.out.printf("%s","compsci\n");
     }
}
```

Unit 1F Worksheet

DIRECTIONS : Fill in each blank with the correct answer/output. Assume each statement happens in order and that one statement may affect the next statement.

```
System.out.print( "hello" );                    // LINE 1

System.out.print( "h\tllo" );                   // LINE 2

System.out.print( "hel\tlo" );                  // LINE 3

System.out.print( "hel\\lo" );                  // LINE 4

System.out.print( 1 + 2 + 3 );                  // LINE 5

System.out.print( 1 + " " + 2 );                // LINE 6

System.out.print("\" quotes  \"");              // LINE 7

System.out.print( "\\t\\a ");                   // LINE 8

System.out.print( "\\\\" );                     // LINE 9

System.out.print( 1 + " " + 2 + 3);             // LINE 10

System.out.print( 1 + 4 + " " + 2 + 3);         // LINE 11
```

```
1.   hello
2.   h    llo
3.   hel    lo
4.   hel\lo
5.   6
6.   1 2
7.   " quotes "
8.   \t\a
9.   \\
10.  1 23
11.  5 23
```

[3]

[3] outputworksheet.doc

DIRECTIONS : Fill in each blank with the correct answer/output. Assume each statement happens in order and that one statement may affect the next statement.

1. All java statements are terminated with a ;____________________.
2. An open brace__________________ must have a close brace___________________.
3. \n___________________ is the newline character.
4. \t____________________ is the tab character.
5. //_______________________ is used for a one line comment.
6. Never put a ;_______________________ before an open brace.

```
System.out.print( "one" );                    // LINE 1
System.out.print( "o\tne" );                  // LINE 2
System.out.print( "on\te" );                  // LINE 3
System.out.print( "on\\e" );                  // LINE 4
System.out.print( 4 + 3);                     // LINE 5
System.out.print( 3 + " " + 4 );              // LINE 6
System.out.print("\" quotes  \"");            // LINE 7
System.out.print( "\\\\t\\a\\");              // LINE 8
System.out.print( "\\a\\b\\c" );              // LINE 9
System.out.print( 3 + " " + 4 + 5);           // LINE 10
System.out.print( 3 + 4 + " " + 5 + 6);       // LINE 11
```

1.	one
2.	o ne
3.	on e
4.	on\e
5.	7
6.	3 4
7.	" quotes "
8.	\\t\a\
9.	\a\b\c
10.	3 45
11.	7 56

4

[4] syntaxoutput.doc

Unit 1F Assignment- ASCII ART

Lab Goal : This lab was designed to teach you how to use print and println to write text to the console window.

Lab Description : Draw a creature/animal using ASCII art.

Lab Shell :

```
public class Lab03a{
   public static void main(String args[])
   {

      System.out.println(" Your Name \t  Date \t Period" );
      System.out.println(" What type of ANIMAL YOU WILL DRAW" );

      System.out.println(" \n\n\n\n" );

      System.out.println("                /\\                " );
      System.out.println("               /  \\               " );
      System.out.println("              /    \\              " );
      System.out.println("             [------]             " );
      //add other output

      System.out.println(" \n\n\n\nHelpFul Hints" );
      System.out.println("\\ draws one backslash on the screen!\n" );
      System.out.println("\" draws one double quote on the screen!\n" );
      System.out.println("\' draws one single quote on the screen!\n" );
   }
}
```

Sample Output :

```
name      date


This animal is a whatever-u-want-it-to-be!

         \         /
           \      /
         #############
         ##  -- --  ##
         ##    {}   ##
         ##         ##
         ##    <>   ##
         ##   ___   ##
         ##         ##
           #########


HelpFul Hints
\ draws one backslash on the screen!

" draws one double quote on the screen!

' draws one single quote on the screen!
```

Files Needed ::
`Lab0a3.java`

Copy and paste the required file into your project folder and start with the code that is included in it.

HelpFul Hints

\\ draws one backslash on the screen!

\" draws one double quote on the screen!

\' draws one single quote on the screen!