CS415

INTRODUCTION TO COMPUTER SCIENCE

FALL 2017

4-BASIC JAVA SYNTAX

CHAPTER 1

LAST TIME

- Models gives us a way to deal with complex systems.
- A program is a model.
- Object Oriented Programming is a way of creating a model that organizes the system into interacting objects.
- Objects have state (properties) and behavior (capabilities.)

OBJECTS IN JAVA

- Objects have type that describe its state and behavior:
  - In Java we will define a "Class" to specify an objects type.
- Objects have state (properties)
  - In a Java Class we will define "instance variables" to implement the objects state.
- Objects have behavior (capabilities.)
  - In a Java class we will define "methods" to implement the objects behavior

PREVIEW

- Java is an OOP Language. This means it will allow us to build an OOP model.
- We will begin to learn the Java language: Syntax, keywords, identifiers.
- Java style conventions: A program not only has to work but it has to look good.
- Instance variables to implement properties.
- Methods to implement capabilities.

ANATOMY

```
import wheelsunh.users.*;
import java.awt.Color;

/**
 * Lab1.java:  Displays a red circle.
 * @author Mark Bochert
 */
public class Lab1
{
    private Ellipse _circle;

    /**
     *   Constructor for the Lab1 class.
     */
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
    }

    /** main program creates a Frame and
      *  invokes the class constructor.
      *
      * @param  a the command line.
      */
    public static void main( String[] a)
    {
       Frame f = new Frame();
       Lab1 app = new Lab1();
    }
} //End of Class Lab1
```

imports

class comment

class header

class body

- instance variable
- constructor comment
- constructor header
- constructor body
- main method comment
- main method header
- main method body

IMPORT

- Sometimes you will create your own objects by writing your own classes, other times you will use object created by other programmers in a collection called a "library".

- To make these easily available you must import the library.

- The first line imports ALL (*) the classes in the library "wheelsunh.users"

- The second line imports only the class "Color" from the library "java.awt"

```
import wheelsunh.users.*;
import java.awt.Color;
```

JAVA COMMENTS

- Good style requires that we use Java comments in our code.
  - Included to explain something to the human reader and ignored by the compiler.
  - The first step in the process of documenting a program.
  - Critical in good programming, it makes the program easier to read and understand.

JAVA COMMENTS

- There are three types of Java comments
  - A Javadoc comment starts with `/**` and ends with `*/`
  - An inline comment starts with `//` and ends at the end of the line.
  - A standard comment begins with `/*` and ends with `*/`

```
/**
*   GarbageCan.java.
*   This is a simple model of a garbage can.
*
*   @author  J. Alfred Prufrock
*/
public class GarbageCan
{
   // here we will model the properties and
   // capabilities of a garbage can
}
```

COMMENT CONVENTIONS

- Conventions are rules that are not required by Java but are required for good style (and a good grade).
- Every class should have a JavaDoc comment, use the "@author" tag before your name.
- Every method should have a JavaDoc comment.
- Inline comments should be used to clarify confusing code.

JAVA KEYWORDS

- The individual units of text in a program (i.e. "words") are referred to a "tokens"

- Some tokens have a predefined meaning in Java, these are called keywords.

```
abstract
continue
for
new
switch
assert
default
goto
package
synchronized
boolean
do
if
```

```
private
this
break
double
implements
protected
throw
byte
else
import
public
throws
case
enum
```

```
instanceof
return
transient
catch
extends
int
short
try
char
final
interface
static
```

```
void
class
finally
long
strictfp
volatile
const
float
native
super
while
true
false
null
```

JAVA IDENTIFIERS

- Sometimes you will want to create your own tokens, for example to name a variable.
- A Java identifier must start with a letter or underscore "_"
- After that it can have any number of letters, numbers or underscores.
- It cannot contain blanks, punctuation or other symbols.
- It cannot be a Java keyword.

IDENTIFIER CONVENTIONS

- Capitalize the first letter of all class names: **`MyClass`**
- Start all other identifiers with lower case letters: **`aMethod`**
- Use internal capital letters for each "word" in the name: **`MyClass`** and **`aMethod`**.
- Use mnemonic, meaningful names: **`NameTable, addName.`**
- Start instance variable names with an underscore: **`_aCircle`**

CLASS HEADER

```
import wheelsunh.users.*;
import java.awt.Color;

/**
 * Lab1.java:  Displays a red circle.
 * @author Mark Bochert
 */
public class Lab1
{
    private Ellipse _circle;

    /**
     *   Constructor for the Lab1 class.
     */
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
    }

    /** main program creates a Frame and
      *  invokes the class constructor.
      *
      * @param  a the command line.
      */
    public static void main( String[] a)
    {
       Frame f = new Frame();
       Lab1 app = new Lab1();
    }
} //End of Class Lab1
```

- The class header gives a name to the class, this class is named Lab1
- The file containing this code must match the class name, this file name must be: Lab1.java
- The token class is a keyword that means we are defining a class.
- The public key word means this class can be used outside this file.
- The open and closing braces delimit the class body.

INDENTATION CONVENTIONS

- Each block of code that is logically nested inside some other piece of code should be indented 4 columns

- Comments should be indented at the same level as the surrounding code.

- the left and right curly brackets of a block are always in the same column.

VARIABLES

- In a program we will manipulate values.
- Values are stored in main memory as variables.
- We can create a new variable with a variable declaration.
- When we declare a variable we need to specify a name and the type of value that it will contain.
- As we will see there are different types of variables for different purposes.

INSTANCE VARIABLES

```
import wheelsunh.users.*;
import java.awt.Color;

/**
 * Lab1.java:  Displays a red circle.
 * @author Mark Bochert
 */
public class Lab1
{
    private Ellipse _circle;

    /**
     *   Constructor for the Lab1 class.
     */
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
    }

    /** main program creates a Frame and
      *  invokes the class constructor.
      *
      * @param  a the command line.
      */
    public static void main( String[] a)
    {
       Frame f = new Frame();
       Lab1 app = new Lab1();
    }
} //End of Class Lab1
```

- A type of variable that is available in all parts of a class is called an instance variable.
- Notice that it is declared inside, at the beginning of the class body but outside any method.
- The private key word means it is not available outside the class.
- The type is: Ellipse
- The name is _circle
- The declaration ends with ;

INSTANCE VARIABLES

- We can describe the syntax of an instance variable declaration by giving its “general form”

  private <type> <identifier>;

```
private Ellipse _circle;   // type is Ellipse name is _circle
private Rectangle _rectangle;   // type is Rectangle name is _rectangle
private int total;           // type is int (integer) name is total
```

INSTANCE VARIABLES

- Multiple instance variables of the same type can be declared with the general form:

  private <type> <identifier>, ... , <identifier>;

```
private Ellipse _circle1, _circle2, _circle3;
```

- Note: At this point in a program you have created the three variable names but there are no circles yet!

MAIN METHOD

```
import wheelsunh.users.*;
import java.awt.Color;

/**
 * Lab1.java:  Displays a red circle.
 * @author Mark Bochert
 */
public class Lab1
{
    private Ellipse _circle;

    /**
     *   Constructor for the Lab1 class.
     */
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
    }

    /** main program creates a Frame and
      *  invokes the class constructor.
      *
      * @param  a the command line.
      */
    public static void main( String[] a)
    {
       Frame f = new Frame();
       Lab1 app = new Lab1();
    }
} //End of Class Lab1
```

- When you click the “run” button in DrJava the java virtual machine(JVM) starts running and it looks for a main method in the class.

- For now take the header as boilerplate

- The body is usually simple, here we just

    1. create a Frame

    2. call our Lab1 constructor

CONSTRUCTOR METHOD

```
import wheelsunh.users.*;
import java.awt.Color;

/**
 * Lab1.java:  Displays a red circle.
 * @author Mark Bochert
 */
public class Lab1
{
    private Ellipse _circle;

    /**
     *   Constructor for the Lab1 class.
     */
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
    }

    /** main program creates a Frame and
      *  invokes the class constructor.
      *
      * @param  a the command line.
      */
    public static void main( String[] a)
    {
       Frame f = new Frame();
       Lab1 app = new Lab1();
    }
} //End of Class Lab1
```

- In this example, the second line in the main method calls the constructor

- The purpose of the constructor is to initialize the instance variables.

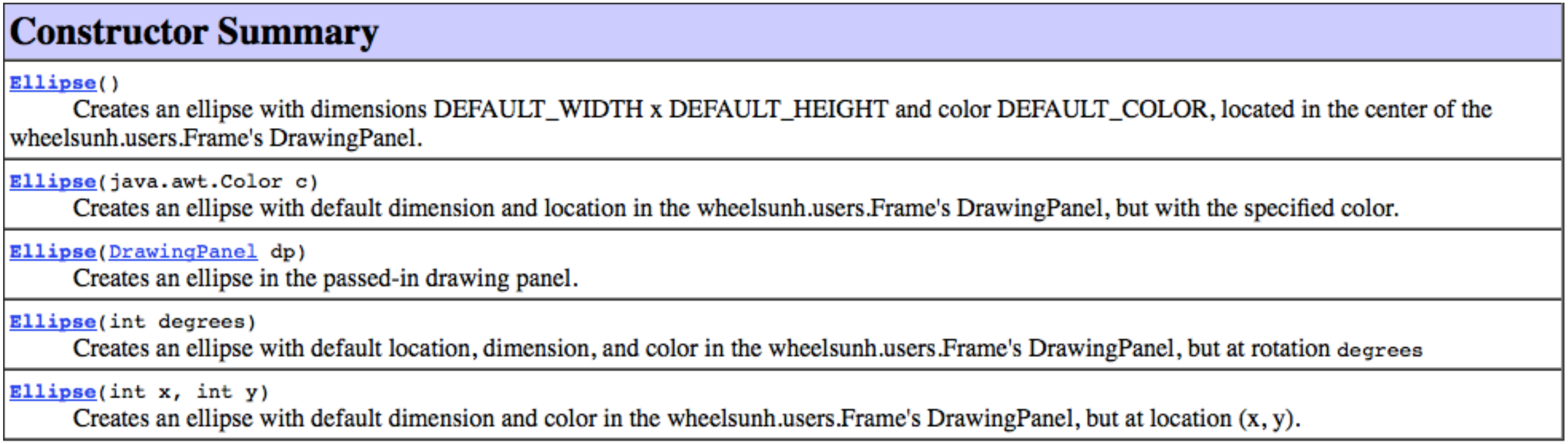
- The constructor then calls the Ellipse constructor to create a new Ellipse and assigns it to the instance variable.

CONSTRUCTING AN OBJECT

```
_circle = new Ellipse( Color.RED );
```

- The keyword "new" indicates that we are calling a constructor to create a new object.

- You need to look a the API to see what constructors are available.



Constructor Summary
`Ellipse()` Creates an ellipse with dimensions DEFAULT_WIDTH x DEFAULT_HEIGHT and color DEFAULT_COLOR, located in the center of the wheelsunh.users.Frame's DrawingPanel.
`Ellipse(java.awt.Color c)` Creates an ellipse with default dimension and location in the wheelsunh.users.Frame's DrawingPanel, but with the specified color.
`Ellipse(DrawingPanel dp)` Creates an ellipse in the passed-in drawing panel.
`Ellipse(int degrees)` Creates an ellipse with default location, dimension, and color in the wheelsunh.users.Frame's DrawingPanel, but at rotation `degrees`
`Ellipse(int x, int y)` Creates an ellipse with default dimension and color in the wheelsunh.users.Frame's DrawingPanel, but at location (x, y).

USING OBJECTS

- In Java each type of object is defined by a Class.
- At first we will use objects define by classes that someone else wrote.
- Remember, Objects have behavior, or capabilities that they can perform.
- In Java each capability is implemented with a method. ("The method of performing the capability")
- You need to look a the API to see what methods are available.

USING OBJECTS

- Once you have created an object you can make it perform its behavior.

- For example, one of the capabilities of an Ellipse is that it can change its size.

```
    public Lab1( )
    {
      _circle = new Ellipse( Color.RED );
      _circle.setSize( 10, 20 );
    }
```

- We say: "We have sent the setSize message to _circle"

- This invokes the setSize method of _circle.

- 10 and 20 are called actual parameters , different methods require different actual parameters.

- The required parameters are specified in the API.

java.awt.Color

- The Java AWT library provides a Color class for modeling colors.
- It provides a few initialized color objects that are "ready to use".
  `Color.yellow` or `Color.YELLOW`
- It provides constructors to create arbitrary colors:

```
Color _myColor = new Color( 255, 0, 127, 100 ) ;
```

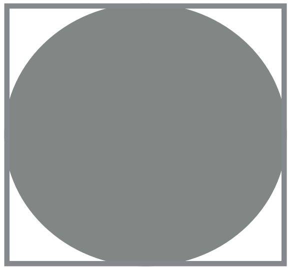
- Creates a color with the specified red, green, blue, and alpha (transparency) values in the range (0 - 255).

GRAPHICS COORDINATES

- The pixels on a Frame are specified with x and y integer coordinates.

- The upper left hand corner of the frame has the coordinates (x, y) =(0, 0)

- The x coordinate increases to the right



- The y coordinate increases down

- The location of an Shape is the upper left hand corner of its "bounding box"

RELATIVE VS. ABSOLUTE COORDINATES

- Absolute Coordinates:

```
_circle.setLocation( 20, 30 );
```

  - The circle is located with absolute coordinates
  - The actual location is (20, 30) on the frame

- Relative Coordinates:

```
int x = 100, y = 50;
_circle.setLocation( x + 20, y + 30 );
```

  - The circle is located at (20, 30) relative to (x, y)
  - In this case the actual location on the frame is (120, 80)