## CSE 142

Defining an Object's Parts



## Overview

- Quick Review
  - Creating, naming, and using objects
  - Creating (empty) classes and instances of them
- Today
  - Details of declaring and initializing names (identifiers)
  - Defining and initializing an object's parts
- Reading
  - Dugan notes: ch. 6, first part of ch. 8



## Naming Revisited – Declarations

- We've used the following pattern to give names to values

  <type> <name> = <expression> ;
- This is called a *declaration*. Purpose:
  - Introduce a new name (or identifier)
  - Specify the type of values it can name
  - Specify the (initial) value that the name is bound to (optional)
- A name can be declared without an initial value
  - A name declared like this must eventually be *initialized* (bound to a value) before it can be used



## Defining Parts of Objects

- "Instance variable" declarations specify the parts (or "fields") that all objects (*instances*) of a class have

```
import uwcse.graphics.*;
public class House {
    private Rectangle frame;
    private Triangle roof;
}
```

- Pattern for an instance variable declaration

  <type> <name of instance variable/part> ;
- Use import declaration to reference library classes

  **import** <compound name of library> **.** * **;**
  - or

  **import** <compound name of library> **.** <ClassName> **;**



## Initializing Instance Variables

- Need to set the values of the parts of a new House object.
- One way: use an *assignment* statement. Pattern:

  <object name> . <instance variable name> = <expression>
- Execution of an assignment statement
  - First, evaluate the expression
  - Second, bind the value to the name
- An assignment statement is not a declaration
  - Does not create a new name



## Initializing House Instance Variables

- We want to execute assignments like the following to initialize a House object when it is created

```
<house object name> . frame = new Rectangle(50, 75, 100, 50, Color.blue, true);
<house object name> .roof = new Triangle(40,75,100,50,160,75,Color.red,true);
```

- Questions
  - Where do these statements go?
  - What is the name of this house?

## Initializing Instance Variables

- We can define a *constructor* in the House class

```
import uwcse.graphics.*;
import java.awt.Color;
/** A House Object */
public class House {
    private Rectangle frame;
    private Triangle roof;
    /** Initialize this house with a frame and a room */
    public House( ) {
        this.frame = new Rectangle(50, 75, 100, 50, Color.blue, true);
        this.roof = new Triangle(40, 75, 100, 50, 160, 75, Color.red, true);
    }
}
```



## Constructors

- Pattern

```
public class <ClassName> {
    ...
    public <ClassName> ( ) {
        <statements to execute to initialize new object>
    }
    ...
}
```

- Inside of the constructor, "this" is a *scratch* name for the new object that's being initialized.
  - "this" binding is thrown away after the constructor finishes
- If one is provided, a constructor is guaranteed to be called when objects are created – gives programmer a chance to guarantee that object fields are initialized properly when object is created.



## Using Objects with Constructors

- Now can create nicely initialized House objects!

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

- In BlueJ: Right click on class icon to create new object



## Making Changes

- Imagine someone telling you that they love your house, they just wish
  - it were a little over to the left
  - it were a little taller
  - it were a little wider
- How hard is it going to be to make these modifications?



## A Better Constructor

- We can rewrite the House constructor to use *local variables*

```
public House( ) {
    int width = 100;
    int height = 50;
    int leftX = 50;
    int topY = 50;
    int roofHeight = height/2;
    int overhang = 10;
    int roofY = topY + roofHeight;
    this.frame = new Rectangle(leftX, roofY, width, height, Color.blue, true);
    this.roof = new Triangle(leftX – overhang, roofY, leftX + width/2, topY,
                             leftX + width + overhang, roofY, Color.red, true);
}
```

- Why is this better?



## What Next?

- We have defined a class representing House shapes
- Next we need to add *methods* to handle messages like addTo(aGWindow), moveBy(deltax, deltay), etc.
- Would also like to make the constructor a bit smarter so we can specify parameters when we create a House; for example

```
House home = new House(50, 50, 100, 50, Color.blue, Color.red);
home.addTo(aGWindow);
```

- Need some place to put the code to create a scene by drawing a House in a GWindow