CSE 142 Summer 2001

Creating a Class

Introduction

- Quick Review:
  - Creating, naming and using objects
  - Expressions and Types
  - Messages and parameters
  - Syntactic and Semantic errors
- Today:
  - Put it all together to build our first class!

Classes and Objects

- The Java World
  - Objects interact with each other by sending and reacting to messages
  - Objects are instances of *classes*
  - A class defines a new type and is a blueprint for individual instances (objects) of that class
- The fundamental programming task in Java is creating or modifying classes

Program Development

- Let's look at some statements we might use to draw a picture of a house:

```
Gwindow theWindow = new Gwindow( );

// Make the frame of the house
theWindow.add(new Rectangle(50, 200, 150, 100));
theWindow.add(new Triangle(30, 180, 230, 180, 125, 150, Color.red, true));

// Make a window by drawing four frames.
theWindow.add(new Rectangle(60, 220, 15, 30));
theWindow.add(new Rectangle(75, 220, 15, 30));
theWindow.add(new Rectangle(60, 250, 15, 30));
theWindow.add(new Rectangle(75, 250, 15, 30));
```


A House class

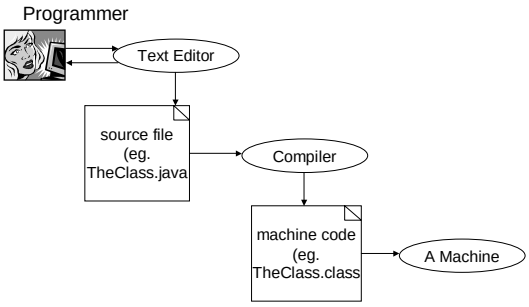
- We would like to package these statements so we can create “house” objects that carry out those actions

```
House home = new House( );
home.drawHouse( );
```


Tools: Editor & Compiler

- We'd like to be able to keep these statements somewhere, so we don't have to retype them all the time.
- We can use a tool (program) called a *text editor* to enter and modify our statements.
- The editor can save our statements into a file (*source file*).
- We can then use another tool called a *compiler* to translate our source code into another language: machine code.
- The compiler produces another file of machine code, which we can give to a machine to execute.

Tools in Pictures: Compilation



Developing Software with a Compiler

- It usually involves (at least) the following steps:
  - 1: Edit your source code
  - 2: Compile your source code
  - 3: Possibly repeat steps 1 and 2. (Why?)
  - 4: Run the program
  - 5: Possibly go back to step 1 (Why?)
- Notice the differences compared to the interpreter:
  - "Batch mode" (compiler) vs. "interactive mode" (interpreter)

A Java Source File

```
/** A House picture */
public class House {

 /** Create a new window and draw a picture of a house in it */
 public void drawHouse( ) {
  // Make a new Graphics window
  GWindow theWindow = new GWindow();

  // Make the basic frame of the house
  theWindow.add(new Rectangle(50, 200, 150, 100));
  theWindow.add(new Triangle(30, 180, 230, 180, 125, 150, . . .));

  // Draw a single window, by drawing 4 panes
  theWindow.add(new Rectangle(60, 220, 15, 30));
  theWindow.add(new Rectangle(75, 220, 15, 30));
  theWindow.add(new Rectangle(60, 250, 15, 30));
  theWindow.add(new Rectangle(75, 250, 15, 30));
 }
}
```

- What looks familiar about this? What looks different?

A Pattern

- For now, use the following pattern:

```
/** Description of the class */
public class < Some name for your class > {

 /** description of the method */
 public void <some name for your method>( ) {

  // put the method's code here

 }
}
```

- If you choose a class name like TheClass, put this in a file and call the file TheClass.java

A Bug

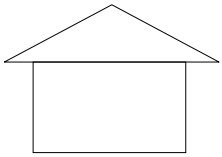
- Run the program.
- Is there something wrong with it?
- This is a kind of error that we haven't seen yet.
- It's as if we've given directions that are perfectly understandable (correct syntax and semantics), it's just that they don't get us to where we want to go…

Style Issues

- Ok, so we've fixed the bug. What else is wrong with the program?
- 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 Program

How would you define the house? What coordinates and dimensions matter?



Picking Dimensions

- Clearly, there are a variety of combinations of dimensions that we can pick:
  - Overall width, wall width, overall height, wall height, (x,y)
  - Wall width, roof width, wall height, roof height, (x,y)
  - Can you think of others?
- There may be no "best" combination, but the second (above) seems pretty natural, so let's go with it.

Rewriting the code:

```
int frameX = 50;        // lower left x of the walls
int frameY = 200;       // lower left y of the walls
int wallWidth = 150;
int wallHeight = 100;
int roofWidth = 200;
int roofHeight = 50;
int overhang = roofWidth - wallWidth;
int roofBaseY = frameY - wallHeight;

// Make the basic frame of the house
theWindow.add(new Rectangle(frameX, roofBaseY, frameWidth, frameHeight));

// Now make the roof
theWindow.add(new Triangle(frameX-overhang, roofBaseY,
                    frameX + wallWidth + overhang, roofBaseY,
                    frameX + (wallWidth/2), roofBaseY - roofHeight));
```


A Better Program

- Notice that both versions of the program have the same basic functionality -- they both draw a house.
- The second version is better, however: higher quality, more beautiful, and so on.
- Why? Because it is easier to modify, easier to understand, and so on.