Comments

- Comment syntax:

```
// Line-oriented.
```

or

```
/*
   block-oriented
   can span several lines.
*/
```

Comments (cont.)

- For block-oriented comments, I suggest:

```
/*
 * Why?  It makes it easier to identify
 * the comments, and distinguish them
 * from code.
 */
```

Class Definitions

- Class definition syntax: *[...] means optional.*

```
[public] [abstract]
          class class-name [inheritance-specification]
{
  [field-list;]

  [constructor-list;]

  [method-list;]
}
```

Examples

```
public class PizzaClient
{
  public static void main(String[] args)
  {
    // instantiation and assignment.

    Pizza cirPizza = new Pizza(4.69, new Circle(2.5));
    Pizza rectPizza = new Pizza(4.49, new Rectangle(5, 4));
```

Examples (cont.)

```
    // output to standard output stream.
    System.out.println(cirPizza);
    System.out.println(rectPizza);
    System.out.print("Round Pizza is a better deal " +
                     "than Rectangular Pizza: ");
    System.out.println(
            (cirPizza.dPrice() / cirPizza.dArea()) <
            (rectPizza.dPrice() / rectPizza.dArea()));
  }
}
```

- NOTE: infix notation for arithmetic expressions, and "dot" notation for method calls.

Examples (cont.)

```
public class Rectangle extends AShape
{
  private double _dHeight; // Note the underscore.
  private double _dWidth;

  public Rectangle(double dWidth, double dHeight)
  {
    _dHeight = dHeight;
    _dWidth = dWidth;
    // the underscore helps distinguish the field from the
    // parameter.
  }
```

Examples (cont.)

```
  public double dArea()
  {
    return _dHeight * _dWidth; // infix notation!
  }

  public String toString()
  {
    return "Rectangle (width = " + _dWidth +
           ", heigth = " + _dHeight + ")";
  }
}
```

Field Definitions

- Field list syntax: A field list consists of zero or more field declarations of the form:

```
[static] [public | private]
        field-type field-name [assignment];
```

Constructor Definitions

- Constructor list syntax: A constructor list consists of zero or more constructor definitions of the form:

```
[public | private] class-name([param-list])
{
  [statement-list;]
}
```

- NOTE: The constructor's name is the same as the class name. Constructors are used for initialization of the object during the object's instantiation only.

Method Definitions

- Method list syntax: A method list consists of zero or more method definitions of the form:

```
[static] [public | private] [abstract]
        return-type method-name([param-list])
{
  [statement-list;]
}
```

- A return type void means the method does not return any value.

Method Definitions (cont.)

- A param-list looks like:

```
type1 param1, type2 param2, ..., typeN paramN
```