COP-2210
Computer Programming I

Instructor: Dr. Antonio Hernandez

Text: Big Java: Early Objects, Interactive Edition, 6th Edition

Developing Better Programs

32. UML

System Development

Large enterprise applications - the ones that execute core business applications, and keep a company going - must be more than just a bunch of code modules. They must be structured in a way that enables scalability, security, and robust execution under stressful conditions, and their structure - frequently referred to as their *architecture* - must be defined clearly enough that maintenance programmers can (quickly!) find and fix a bug that shows up long after the original authors have moved on to other projects.

taken from the omg webpage (omg.org)

omg – object management group

System Development

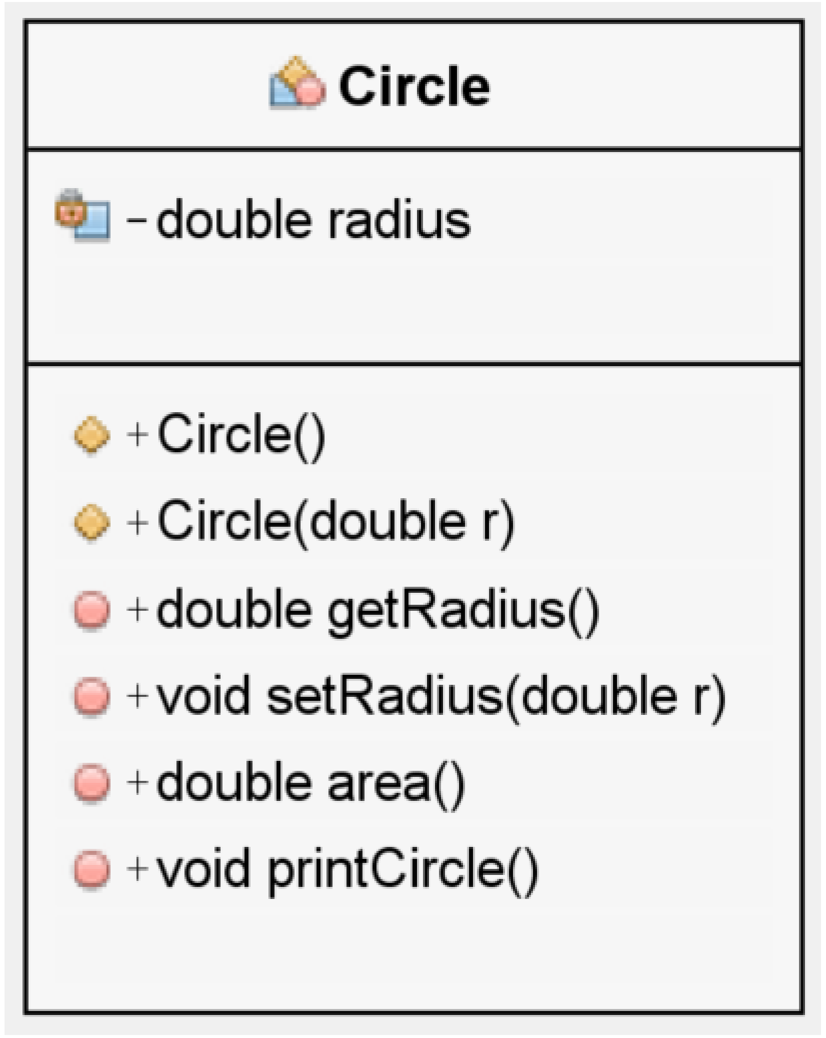
The Unified Modeling Language (UML)

- object oriented modeling language sponsored by the *Object Management Group* (OMG)

- published as a standard in 1997.

- *UML* is the result of an effort headed by the *OMG* to develop a common set of diagrams and notation for the analysis, design, and modeling of (object oriented) systems.

UML: Class Diagram

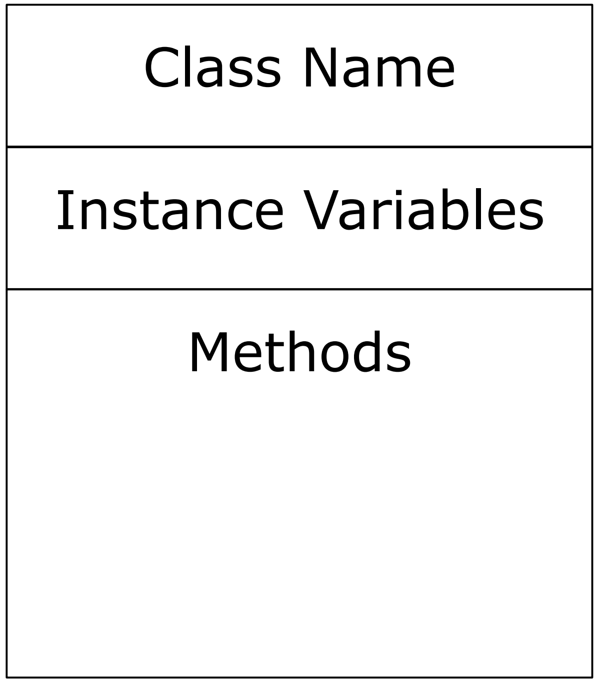


Circle

- double radius

+ Circle()
+ Circle(double r)
+ double getRadius()
+ void setRadius(double r)
+ double area()
+ void printCircle()

+ : public
- : private



Documentation

33. Javadoc

Javadoc

Javadoc:

Tool distributed with Java Development Kit (JDK) used to generate documentation in HTML.

Javadoc use a predefined format:

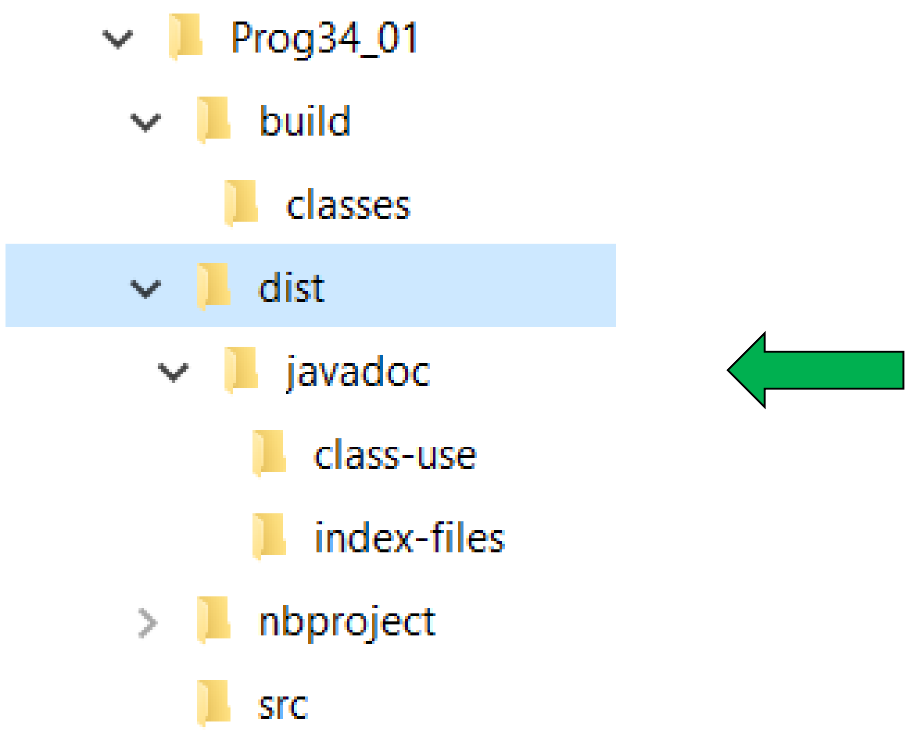
```
/**

    tags and other comments

*/
```

Javadoc: Tags

Tag	Syntax	Description
author	@author name	Adds class author.
version	@version version	Adds version.
since	@since release	States version in which class/method was added.
param	@param name description	Describes method parameter.
return	@return description	Describes return value.

Javadoc

NetBeans: for automatic generation, use "Run/Generate Javadoc"



Javadoc: *Try it yourself*

```
/**
 * Implements a Circle class. Based on Prog31_09.
 *
 * @author Antonio Hernandez
 * @version 1.0
 * @since 2018-09-01
 */
public class Circle
{
    private double radius;

    /**
     * Default constructor.
     */
    public Circle()
    {
        radius = 5;
    }
```

```
    /**
     * Parameterized constructor.
     *
     * @param r radius of circle
     */
    public Circle(double r)
    {
        radius = r;
    }

    /**
     * Returns the radius of the circle.
     *
     * @return the radius of the circle
     */
    public double getRadius()
    {
        return radius;
    }
```

Javadoc: *Try it yourself*

```
/**
 * Sets the radius of the circle.
 *
 * @param r the radius of the circle
 */
public void setRadius(double r)
{
    radius = r;
}

/**
 * Calculates the area of the circle.
 *
 * @return the area of the circle
 */
public double area()
{
    return Math.PI*Math.pow(radius, 2);
}
```

```
/**
    * Returns a description of this circle.
    *
    * @return string describing this circle
    */
   public String toString()
   {
      return "Radius = " + radius +
                  ", area = " + area();
   }
}
```

Javadoc: *Try it yourself*

```
/**
 * Tests the Circle class.
 *
 * @author Antonio Hernandez
 */
public class Prog33_01
{
    public static void main(String[] args)
    {
        new Prog33_01();
    }
    public Prog33_01()
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(6);

        System.out.println("Circle 1: " + c1);
        System.out.println("Circle 2: " + c2);
    }
}
```

PRACTICE

Program 33_02:

Write a program that defines and tests a class **Sphere**. Make the class variable private, include accessor/mutator methods, a default constructor, a parameterized constructor, and a toString method. **Add Javadoc documentation.**

$$V = \frac{4}{3}\pi r^3 \qquad A = 4\pi r^2$$

PRACTICE

Program 33_03:

Write a program that defines and tests a class **Pyramid**. Make the class variables private, include accessor/mutator methods, a default constructor, a parameterized constructor, and a toString method. **Add Javadoc documentation.**

V = (width * length * height) /3

More on Modifiers

34. Static Modifier

Modifiers: *static*

Static variable/method:

associated with the class, not with objects

Outside the class, they may be accessed by

<class name> . <variable/method name>

Example

```
class MyClass
{
        static int x;

        static double myMethod()
        {

                .
                .
                .
        }
}
```

```
//another class, another method


int w = MyClass . x  +  1;
double d = MyClass.myMethod();
```

Modifiers: *static*

Terminology:

class methods/variables = static methods/variables

instance methods/variables = non static methods/variables

Within a class:

- Instance methods can access instance variables and instance methods directly.
- Instance methods can access class variables and class methods directly.
- Class methods can access class variables and class methods directly.
- Class methods cannot access instance variables or instance methods directly—they must use an object reference.
- Class methods cannot use the *this* keyword as there is no instance for *this* to refer to.

Static Modifier: *Try it yourself*

```
public class Circle  //from 31_04
{  private double radius;
   private static int numberOfCircles = 0;

   public Circle()
   {
      radius = 1;
      numberOfCircles++;
   }

   public Circle(double r)
   {
      radius = r;
      numberOfCircles++;
   }

   public static int getNumberOfCircles()
   {
      return numberOfCircles;
   }
```

static member

accessor

```
   public double getRadius()
   {
      return radius;
   }

   public void setRadius(double r)
   {
      radius = r;
   }

   public double area()
   {
      return Math.PI*Math.pow(radius, 2);
   }

   public String toString()
   {
      return "Radius = " + radius +
            ", area = " + area();
   } }
```

Static Modifier: *Try it yourself*

```
public class Prog34_01
{
    public static void main(String[] args)
    {
        new Prog34_01();
    }

    public Prog34_01()
    {
        System.out.println("Number of circles: " + Circle.getNumberOfCircles());

        Circle c1 = new Circle();
        System.out.println("Number of circles: " + Circle.getNumberOfCircles());

        Circle c2 = new Circle(6);
        System.out.println("Number of circles: " + Circle.getNumberOfCircles());
    }
}
```

Creating our Own Packages

35. Packages

Packages

Package: group of related classes

To indicate that a class file is part of a package, include the package declaration at the beginning of the file :

package <package name>

Packages

```
// File Class1.java

package MyPackage

    class Class1
    {
            . . .
    }
```

```
// File Class2.java

package MyPackage

    class Class2
    {
            . . .
    }
```

```
// File TestingPackages.java
import MyPackage.*;
public class TestingPackages.java{
Class1 c1;
Class2 c2;
. . .
```

Access Modifiers

Java access modifiers to specify the type of access granted to *variables* and *methods*.

private, ***package,*** ***protected*** and *public*

No keyword

Public interface

That’s It!

Good Luck in your Finals!