# COP-2210
# Computer Programming I

Instructor: Dr. Antonio Hernandez

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

# Creating Your Own Classes

## 30. Access Modifiers

## Access Modifiers

1. The members can see and access each other
2. Specifiers:

    *private*: the “world” cannot use these members

    *public*: free use

3. Class name: acts as a “type”
4. Variables whose type is a class are called “objects”

# Overloaded Methods: *Try it yourself*

```
public class Circle
{
    private double radius=1;

    public double getRadius()
    {
        return radius;
    }

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

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

Accessor method

Mutator method

```
    public void printCircle()
    {
        System.out.println("Radius = " + radius +
                           ", area = " + area());
    }
}
```

## **Classes**: *Try it yourself*

```
import java.util.Random;

public class Prog30_01
{
    public static void main(String[] args)
    {
        new Prog30_01();
    }
    public Prog30_01()
    {
        Circle c = new Circle();

        c.printCircle();
        c.setRadius(2);
        c.printCircle();

        c.setRadius(new Random().nextInt(10));
        System.out.println("New radius = " + c.getRadius());
    }
}
```

# PRACTICE

**Program 30_02:**

Write a program that defines and tests a class **Sphere**. Make the class variable ***private*** and add accessor/mutator methods.

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

## PRACTICE

**Program 30_03:**

Write a program that defines and tests a class **Pyramid**. Make the class variables ***private*** and add accessor/mutator methods. Create a tester class **Prog30_03.**

V = (width * length * height) /3

## PRACTICE

**Program 30_04:**

Write a program that defines and tests a class **Person**. Make the class variables (first name, last name, age) ***private*** and add accessor/mutator methods. Create a tester class **Prog30_04.**

# PRACTICE

**Program 30_05:**

Write a program that defines and test classes **Student** and **Professor**. Make the class variables ***private*** and add accessor/mutator methods.

*Student features: first name, last name, GPA;*
*Professor features: first name, last name, research area.*

Create a tester class **Prog30_05.**

# Creating Your Own Classes

## 31. Constructors

## Class Constructors

**Constructor**: It is a special type of method of a class.

1) Java will execute whatever code is inside the constructor when the object is created
2) It is mainly used for **initialization** purposes
3) It is *invoked* in the declaration
4) It can only be called in conjunction with the **new** operator.

<u>Example</u>

$$\texttt{Scanner in = }\underbrace{\texttt{new Scanner ( System.in );}}_{\textit{invoking the constructor}}$$

## Class Constructors

5) A constructor has the **same name** as the class
6) It can be overloaded
7) It has no return value

## Constructors: *Try it yourself*

```
public class Circle
{
   private double radius;

   public Circle()
   {
      radius = 5;
   }

   public double getRadius()
   {
      return radius;
   }

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

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

   public void printCircle()
   {
      System.out.println("Radius = " + radius +
                        ", area = " + area());
   }
}
```

Constructor method

## Constructors: *Try it yourself*

```
import java.util.Random;

public class Prog31_01
{
    public static void main(String[] args)
    {
        new Prog31_01();
    }
    public Prog31_01()
    {
        Circle c = new Circle();

        c.printCircle();
        c.setRadius(2);
        c.printCircle();

        c.setRadius(new Random().nextInt(10));
        System.out.println("New radius = " + c.getRadius());
    }
}
```

## PRACTICE

**Program 31_02:**

Write a program that defines and tests a class **Sphere**. Make the class variable private, include accessor/mutator methods and a ***constructor***.

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

## PRACTICE

**Program 31_03:**

Write a program that defines and tests a class **Pyramid**. Make the class variables private, include accessor/mutator methods and a ***constructor***.

V = (width * length * height) /3

# Parameterized Constructors: *Try it yourself*

```
public class Circle
{
   private double radius;

   public Circle()
   {
      radius = 1;
   }

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

   public double getRadius()
   {
      return radius;
   }

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

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

   public void printCircle()
   {
      System.out.println("Radius = " + radius +
                         ", area = " + area());
   }
}
```

Default constructor

Parameterized constructor

## Parameterized Constructors: *Try it yourself*

```
public class Prog31_04
{
    public static void main(String[] args)
    {
        new Prog31_04();
    }
    public Prog31_04()
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(6);

        c1.printCircle();
        c2.printCircle();
    }
}
```

# PRACTICE

**Program 31_05:**

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

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

# PRACTICE

**<u>Program 31_06:</u>**

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

V = (width * length * height) /3

## Class Constructors (continued)

**Constructor**: It is a special type of method of a class.

1) Java will execute whatever code is inside the constructor when the object is created
2) It is mainly used for **initialization** purposes
3) It is *invoked* in the declaration
4) It can only be called in conjunction with the **new** operator.

<u>Example</u>

$$\texttt{Scanner in = new }\underbrace{\texttt{Scanner ( System.in )}}_{\textit{invoking the constructor}}\texttt{;}$$

## Class Constructors (continued)

5) A constructor has the **same name** as the class
6) It can be overloaded
7) It has no return value
8) It can have any number of parameters, including zero. Constructors with 0 parameters are called **default constructors**
9) If no constructor is defined, a default constructor is provided. This one will set all class variable with default values
10) If a class supplies at least one constructor but does not supply a default one, it is illegal to construct objects without arguments

# this Keyword

***this*** **keyword**:

Within a method or a constructor, **this** is a reference to the current object — the object whose method or constructor is being called.

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

```
public void setRadius(double radius)
{
    this.radius = radius;
}
```

# this Keyword : *Try it yourself*

```
public class Circle
{
    private double radius;

    public Circle()
    {
        radius = 1;
    }

    public Circle(double radius)
    {
        this.radius = radius;
    }

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

```
    public void setRadius(double radius)
    {
        this.radius = radius;
    }

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

    public void printCircle()
    {
        System.out.println("Radius = " + radius +
                                 ", area = " + area());
    }
}
```

## this Keyword : *Try it yourself*

```
public class Prog31_07
{
    public static void main(String[] args)
    {
        new Prog31_07();
    }
    public Prog31_07()
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(6);

        c1.printCircle();
        c2.printCircle();
    }
}
```

## toString Method

**toString():** special method used to obtain a string representation of an object.

Whenever we call *System.out.println()* with an object name, toString is called.

toString(): It is a convenient method that it is recommended to have implemented in our classes.

# Without toString: *Try it yourself*

```
//Same as Prog31_04 or Prog31_07

public class Circle
{
    private double radius;

    public Circle()
    {
        radius = 1;
    }

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

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

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

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

    public void printCircle()
    {
        System.out.println("Radius = " + radius +
                           ", area = " + area());
    }
}
```

## Without toString: *Try it yourself*

```
public class Prog31_08
{
    public static void main(String[] args)
    {
        new Prog31_08();
    }
    public Prog31_08()
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(6);

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

## With toString: *Try it yourself*

```
public class Circle
{
    private double radius;

    public Circle()
    {
        radius = 1;
    }

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

    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();
    }
}
```

## With toString: *Try it yourself*

```
public class Prog31_09
{
    public static void main(String[] args)
    {
        new Prog31_09();
    }
    public Prog31_09()
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(6);

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

## PRACTICE

**Program 31_10:**

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***.

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

# PRACTICE

**Program 31_11:**

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***.

V = (width * length * height) /3