Derived Classes and Inheritance

Chapter 9

Objectives

You will be able to:

- Create and use derived classes.
- Understand the concept of inheritance.

Derived Classes

Key Concept

- One of the key concepts of object oriented programming is the ability to create new classes from existing classes

  without changing the existing class.

- The new class is called a *derived class*
  - or *subclass*, or *child* class.
- The original class is called the *base class*
  - or *superclass* or *parent* class.

Derived Classes

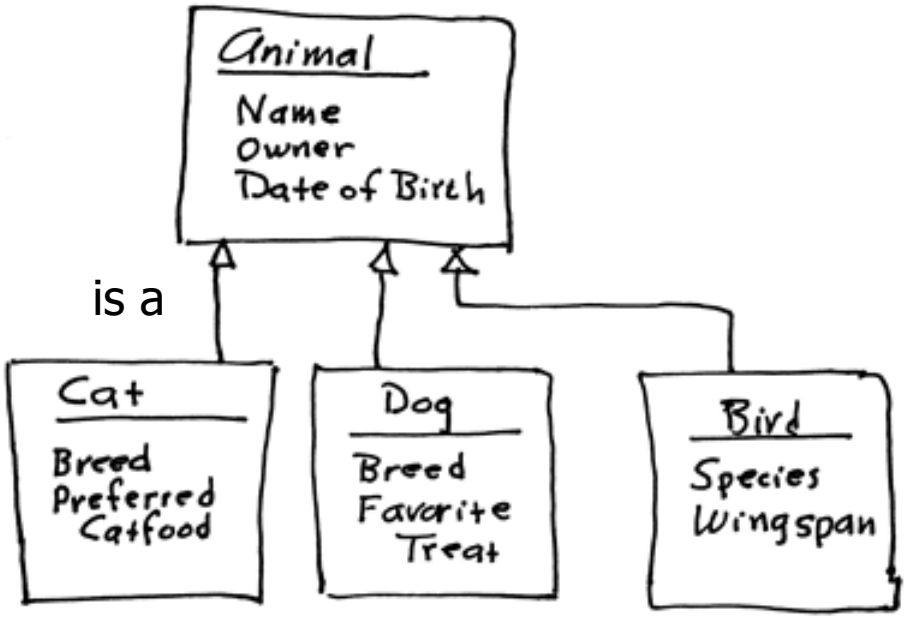
- A derived class *extends* the definition of an existing class.
  - Can add new attributes.
  - Can add new methods.

- All members of the base class are members of the derived class.

The "is a" Relationship

Base Class

Derived Classes



The "is a" Relationship

- The Liskov Substitution Principle

Key Concept

- Objects of a derived class can be used anwhere objects of the original class could be used.
  - Variables
  - Arguments to methods

https://en.wikipedia.org/wiki/Barbara_Liskov

Example: A hierarchy of geometrical shapes

- Shape
  - Triangle
  - Rectangle
  - Circle
- All shapes have certain attributes in common.
  - Number of sides
  - Area
- Each shape has some unique information.
- Let's create a base class for Shape
  - Then create derived classes for specific kinds of shapes.

```
//*************************************************
//  Shape.java
//
//  Represents a geometrical shape
//
//*************************************************
public class Shape
{
    private static int nr_shape_objects = 0;

    // Instance variables
    private String name;
    private int id;
    private int nr_sides;
    private double area;

    //-------------------------------------------------------
    // Constructor - Initializes instance variables
    //-------------------------------------------------------
    public Shape(String name, int nr_sides, double area)
    {
        this.name = name;
        this.nr_sides = nr_sides;
        this.area = area;
        this.id = ++nr_shape_objects;
    }
```

```
    //-----------------------------------------------------
    // Returns a string representation of a Shape.
    //-----------------------------------------------------
    public String toString()
    {
        return name +
               " id = " + id +
               " Number of sides = " + nr_sides +
               " Area = " + area;
    }

    //-----------------------------------------------------
    // Accessor methods
    //-----------------------------------------------------
    public static int Nr_shape_objects() { return nr_shape_objects; }
    public String Name() { return name; }
    public int Id() { return id; }
    public int Nr_sides() { return nr_sides; }
    public double Area() { return area; }
}
```

Compile

Shape_Tester.cpp

```
//*************************************************
//  Shape_Tester
//     A test driver for class Shape.
//*************************************************
import java.util.Scanner;

public class Shape_Tester
{
    public static void main (String[] args)
    {
        Shape[] all_shapes = new Shape[100];

        while (Shape.Nr_shape_objects() < 100)
        {
            String shapeName;
            int nrSides;
            double area;

            Scanner keyboardScanner = new Scanner(System.in);
            System.out.print ("Enter the name of a shape: ");
            shapeName = keyboardScanner.nextLine();
```

Shape_Tester.cpp

```
            System.out.print ("Enter the number of sides: ");
            nrSides = keyboardScanner.nextInt();
            System.out.print ("Enter the area: ");
            area = keyboardScanner.nextDouble()
            System.out.println();

            // Create a shape object
            Shape new_shape = new Shape(shapeName, nrSides, area);

            int count = Shape.Nr_shape_objects();
            all_shapes[count-1] = new_shape;

            // Print all shapes
            for (int i = 0; i < count; i++)
            {
                System.out.println(all_shapes[i]);
            }
            System.out.println();
        }
    }
}
```

Test Run

Command Prompt - java Shape_Tester

```
C:\test>
C:\test>javac Shape.java

C:\test>javac Shape_Tester.java

C:\test>java Shape_Tester
Enter the name of a shape: Triangle
Enter the number of sides: 3
Enter the area: 12.5

Triangle id = 1 Number of sides = 3 Area = 12.5

Enter the name of a shape: Rectangle
Enter the number of sides: 4
Enter the area: 20.0

Triangle id = 1 Number of sides = 3 Area = 12.5
Rectangle id = 2 Number of sides = 4 Area = 20.0

Enter the name of a shape: Circle
Enter the number of sides: 0
Enter the area: 314.16

Triangle id = 1 Number of sides = 3 Area = 12.5
Rectangle id = 2 Number of sides = 4 Area = 20.0
Circle id = 3 Number of sides = 0 Area = 314.16

Enter the name of a shape:
```

Derived Class

- Now, let's create some derived classes
  - Rectangle
  - Triangle
  - Circle

- All are Shapes.

Class Rectangle

```
//*************************************************
//  Rectangle.java
//
//  Represents a geometrical rectangle
//
//*************************************************
public class Rectangle extends Shape
{
    // Instance variables
    double length;
    double width;

    //-------------------------------------------------------
    // Constructor - Initializes instance variables
    //-------------------------------------------------------
    public Rectangle(double length, double width)
    {
        super("Rectangle", 4, length*width );
        this.length = length;
        this.width = width;
    }
```

Makes Rectangle a subclass of Shape

Invoke constructor for Shape

Class Rectangle

```
    public String toString()
    {
        return super.toString() + " length = " + length +
               " width = " + width;
    }
}
```

Invoke toString( ) for class Shape

Modify the Test Driver

```
while (Shape.Nr_shape_objects() < 100)
{
    String shapeName;
    int nrSides;
    double area;
    Shape new_shape;

    Scanner keyboardScanner = new Scanner(System.in);
    System.out.print ("Enter the name of a shape: ");
    shapeName = keyboardScanner.nextLine();
    if (shapeName.equals("Rectangle"))
    {
        double length, width;
        System.out.print ("Length: ");
        length = keyboardScanner.nextDouble();
        System.out.print ("Width: ");
        width = keyboardScanner.nextDouble();
        System.out.println();

        // Create a Rectangle object
        new_shape = new Rectangle(length, width);
    }
```

Test_Shape.java

```
            else
            {
                System.out.print ("Enter the number of sides: ");
                nrSides = keyboardScanner.nextInt();
                System.out.print ("Enter the area: ");
                area = keyboardScanner.nextDouble();
                System.out.println();

                // Create a shape object
                new_shape = new Shape(shapeName, nrSides, area);
            }

            int count = Shape.Nr_shape_objects();
            all_shapes[count-1] = new_shape;

            // Print all shapes
            for (int i = 0; i < count; i++)
            {
                System.out.println(all_shapes[i]);
            }
            System.out.println();
        }
    }
}
```

Test Run

```
Command Prompt - java Shape_Tester
C:\test>
C:\test>javac Shape.java

C:\test>javac Rectangle.java

C:\test>javac Shape_Tester.java

C:\test>java Shape_Tester
Enter the name of a shape: Rectangle
Length: 4.0
Width: 3.0

Rectangle id = 1 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0

Enter the name of a shape: Triangle
Enter the number of sides: 3
Enter the area: 12.5

Rectangle id = 1 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0
Triangle id = 2 Number of sides = 3 Area = 12.5

Enter the name of a shape: Circle
Enter the number of sides: 0
Enter the area: 314.16

Rectangle id = 1 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0
Triangle id = 2 Number of sides = 3 Area = 12.5
Circle id = 3 Number of sides = 0 Area = 314.16

Enter the name of a shape: Rectangle
Length: 5
Width: 4

Rectangle id = 1 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0
Triangle id = 2 Number of sides = 3 Area = 12.5
Circle id = 3 Number of sides = 0 Area = 314.16
Rectangle id = 4 Number of sides = 4 Area = 20.0 length = 5.0 width = 4.0

Enter the name of a shape: _
```

Things to Notice

- We stored our new Rectangle into a variable of class Shape.

```
// Create a Rectangle object
new_shape = new Rectangle(length, width);
```

- This works because a Rectangle *is a* Shape.
  - We can use a Rectangle anywhere we could use a Shape.

Things to Notice

```
// Print all shapes
for (int i = 0; i < count; i++)
{
    System.out.println(all_shapes[i]);
}
```

- When we passed a Rectangle to println, println used the toString method defined for class Rectangle.
- When we passed other shapes to println, println used the toString method defined for class Shape.

- This is *polymorphism*.
  - From Greek for *many forms*

Key Concept

Class Triangle

```
//*************************************************
//  Triangle.java
//
//  Represents a geometrical triangle
//
//*************************************************
public class Triangle extends Shape
{
    // Instance variables
    double base;
    double height;

    //-------------------------------------------------------
    // Constructor - Initializes instance variables
    //-------------------------------------------------------
    public Triangle(double base, double height)
    {
        super("Triangle", 3, base*height/2.0 );
        this.base = base;
        this.height = height;
    }
```

Class Triangle

```
    public String toString()
    {
        return super.toString() + " base = " + base +
                " height = " + height;
    }
}
```

Update Test Driver

```
}
else if (shapeName.equals("Triangle"))
{
    double base, height;
    System.out.print("Base: ");
    base  = keyboardScanner.nextDouble();
    System.out.print ("Height: ");
    height = keyboardScanner.nextDouble();
    new_shape = new Triangle(base, height);
    System.out.println();
}
else
```

Test Run

Command Prompt - java Shape_Tester

```
C:\test>
C:\test>javac Triangle.java

C:\test>javac Shape_Tester.java

C:\test>java Shape_Tester
Enter the name of a shape: Triangle
Base: 10
Height: 5

Triangle id = 1 Number of sides = 3 Area = 25.0 base = 10.0 height = 5.0

Enter the name of a shape: Rectangle
Length: 4
Width: 3

Triangle id = 1 Number of sides = 3 Area = 25.0 base = 10.0 height = 5.0
Rectangle id = 2 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0

Enter the name of a shape: Circle
Enter the number of sides: 0
Enter the area: 314.16

Triangle id = 1 Number of sides = 3 Area = 25.0 base = 10.0 height = 5.0
Rectangle id = 2 Number of sides = 4 Area = 12.0 length = 4.0 width = 3.0
Circle id = 3 Number of sides = 0 Area = 314.16

Enter the name of a shape:
```

Class Circle

```
//*****************************************
//  Circle.java
//
//  Represents a geometrical circle
//
//*****************************************
public class Circle extends Shape
{
    // Instance variables
    double radius;

    //-----------------------------------------------
    // Constructor - Initializes instance variables
    //-----------------------------------------------
    public Circle(double radius)
    {
        super("Circle", 0, Math.PI*radius*radius );
        this.radius = radius;
    }
```

Class Circle

```
    public String toString()
    {
        return super.toString() + " radius = " + radius;
    }
}
```

Shape_Tester.java

Add

```
else if (shapeName.equals("Circle"))
{
    double radius;
    System.out.print("Radius: ");
    radius  = keyboardScanner.nextDouble();
    new_shape = new Circle(radius);
    System.out.println();
}
```

Test Run

Command Prompt - java Shape_Tester

```
C:\test>
C:\test>javac Circle.java

C:\test>javac Shape_Tester.java

C:\test>java Shape_Tester
Enter the name of a shape: Circle
Radius: 10

Circle id = 1 Number of sides = 0 Area = 314.1592653589793 radius = 10.0

Enter the name of a shape: Triangle
Base: 10
Height: 5

Circle id = 1 Number of sides = 0 Area = 314.1592653589793 radius = 10.0
Triangle id = 2 Number of sides = 3 Area = 25.0 base = 10.0 height = 5.0

Enter the name of a shape: Rectangle
Length: 4
Width: 6

Circle id = 1 Number of sides = 0 Area = 314.1592653589793 radius = 10.0
Triangle id = 2 Number of sides = 3 Area = 25.0 base = 10.0 height = 5.0
Rectangle id = 3 Number of sides = 4 Area = 24.0 length = 4.0 width = 6.0

Enter the name of a shape: _
```

Summary

- Inheritance is a key concept of OOD.
  - Permits us to extend existing classes without modifying the original code.
- A derived class *extends* its base class.
  - New member variables.
  - New methods.