Abstract Classes and Inheritance

CIS 110

Abstract Class

- An abstract class is a class that cannot be instantiated
- It is similar to an interface with key differences
  - An abstract class can contain code, attributes, and a constructor
  - An abstract class cannot be implemented, but it can be extended (more on this in a bit)

Example, Triangle.java

workspace - Java - InClassApril5/src/Triangle.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Explorer | JUnit

- Counter
- CreationPatterns
- ExamDecorator
- FacadeExample
- HW7
- HW8
- InClassApril4
- InClassApril5
  - src
    - (default package)
      - Circle.java
      - EquilateralTriangle.java
      - Person.java
      - PersonDemo.java
      - RightTriangle.java
      - Shape.java
      - Square.java
      - Triangle.java
  - JRE System Library [JavaSE-1.8]
- InClassMarch13
- InClassMarch15
  - src
    - (default package)
      - Circle.java
      - OverloadedMethods.java
      - PezDispenser.java
      - Shape.java
      - Square.java
      - StringEquality.java
      - TypePromotion.java
  - JRE System Library [JavaSE-1.8]
- InClassMarch20
  - src
    - (default package)
      - Circle.java
      - EquilateralTriangle.java
      - PezDispenser.java

EquilateralTriangle.java | RightTriangle.java | Shape.java | Square.java | Triangle.java | Circle.java | Person.java | PersonDemo.java

```java

public abstract class Triangle implements Shape {
    protected double sideA, sideB, sideC;

    public Triangle(double sideA, double sideB, double sideC) {
        if (hasValidSize(sideA, sideB, sideC)) {
            this.sideA = sideA;
            this.sideB = sideB;
            this.sideC = sideC;
        }
        else {
            throw new IllegalArgumentException("Triangle Sides not valid.");
        }
    }

    private static boolean hasValidSize(double a, double b, double c) {
        return (a + b > c) && (b + c > a) && (a + c > b);
    }

    @Override
    public abstract double area();
```

Console

<terminated> PersonDemo [Java Application] C:\Program Files\Java\jdk1.8.0_101\bin\javaw.exe (Apr 5, 2017, 11:34:01 AM)

```
2017
```

Writable | Smart Insert | 20 : 14

Ideas behind Triangle

- All triangles have 3 sides
- All triangles have a perimeter calculated the same way.

- The area of triangles can be easily calculated in special cases
  - Right Triangle
  - Equilateral triangle

Extending a class

- **You can extend classes that aren't abstract!**
- However, it is most common to do this.
- Extending a class takes a class and can add functionality or change existing functionality

Example, Equilateral Triangle

- An Equilateral Triangle is a triangle with 3 equal sides
- It’s perimeter is calculated the same as a normal triangle
- Area is $sideLength^2 * ¼ * root(3)$

Equilateral Triangle

```java
public class EquilateralTriangle extends Triangle {

    public EquilateralTriangle(double sideLength) {
        super(sideLength, sideLength, sideLength);
    }

    @Override
    public double area() {
        // TODO Auto-generated method stub
        return Math.sqrt(3) * 0.25 * super.sideA * super.sideA;
    }

}
```

EquilateralTriangle.java

- Overrides
  - Unimplemented Area method
- Uses Parent methods for
  - perimeter, draw
- Has it’s own constructor
  - It calls parent constructor

super Keyword

- super refers to the parent object
- Whenever you create a child object, it creates a parent object

- You can access parent protected and public attributes and methods using `super.variableName` or `super.methodName()`

Equilateral Triangle Constructor

```
public EquilateralTriangle(double sideLength) {
    super(sideLength, sideLength, sideLength);
}
```

A child class MUST call the parent constructor in the very first line.

A child calls the parent constructor using super

Only static “helper” methods can be called.

In class activity

- Write RightTriangle.java
  - Extends Triangle.java
- Constructor takes in two arguments (not hypotenuse)
  - I.e., RightTriangle(3.0, 4.0) generates a Triangle with sides 3, 4, and 5
- Overwrite necessary methods

Extending a Class

- You can extend ANY class, abstract or otherwise
- You MUST implement any abstract methods
- The constructor must call the **super()** constructor
- All objects extend the **Object** class.