

Lesson 14-1: Polymorphism and Abstraction

Computer Science 46A: Introduction to Programming
San José State University

Announcements

- Grading issues – please email Mahima Chowdary Mannava mahimachowdary.mannava@sjsu.edu and cc Me
- After this class, we wont introduce new lesson
 - Only review
- Midterm 2 – when is it?
- Finals?

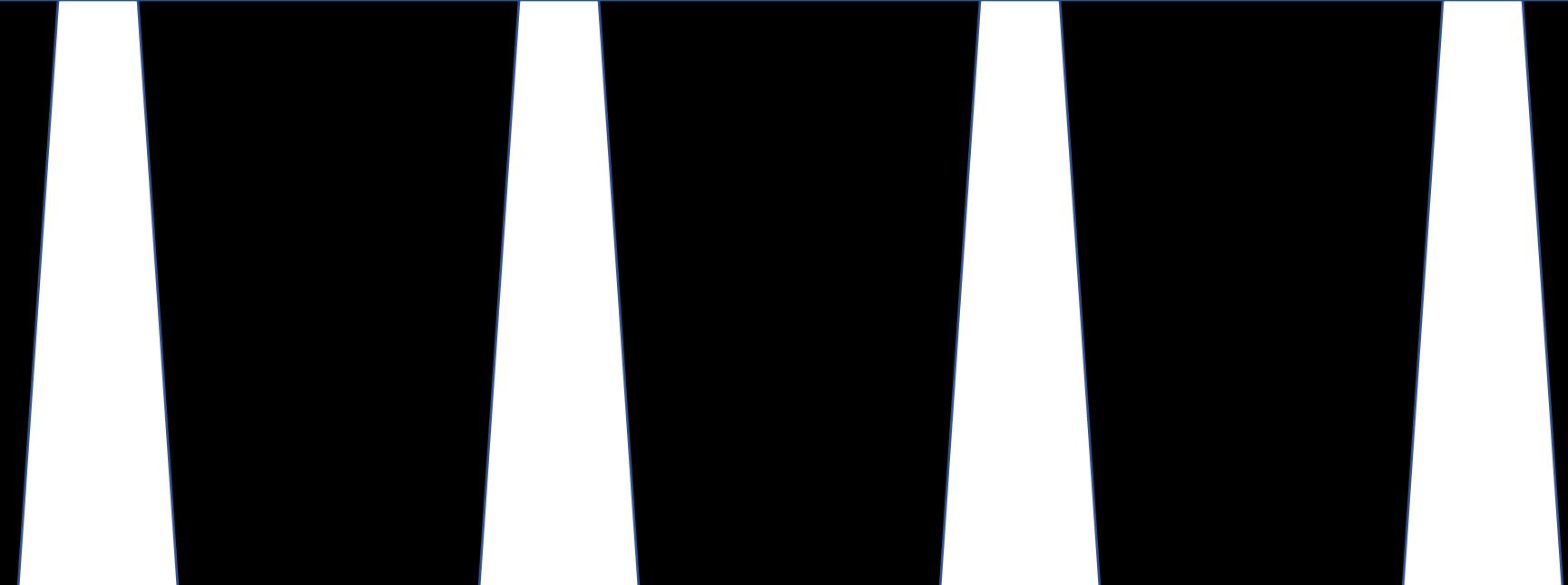
Learning Objectives

By the end of this lesson, you should be able to:

1. Describe the concept of polymorphism
2. Explain how classes extend the Object class by default
3. Describe the concept of abstraction
4. Summarize the 4 main principles of Object-Oriented Programming

4 Pillars of OOP

Object-Oriented Programming



Encapsulation

Inheritance

Polymorphism

Abstraction

So far, we've
discussed:

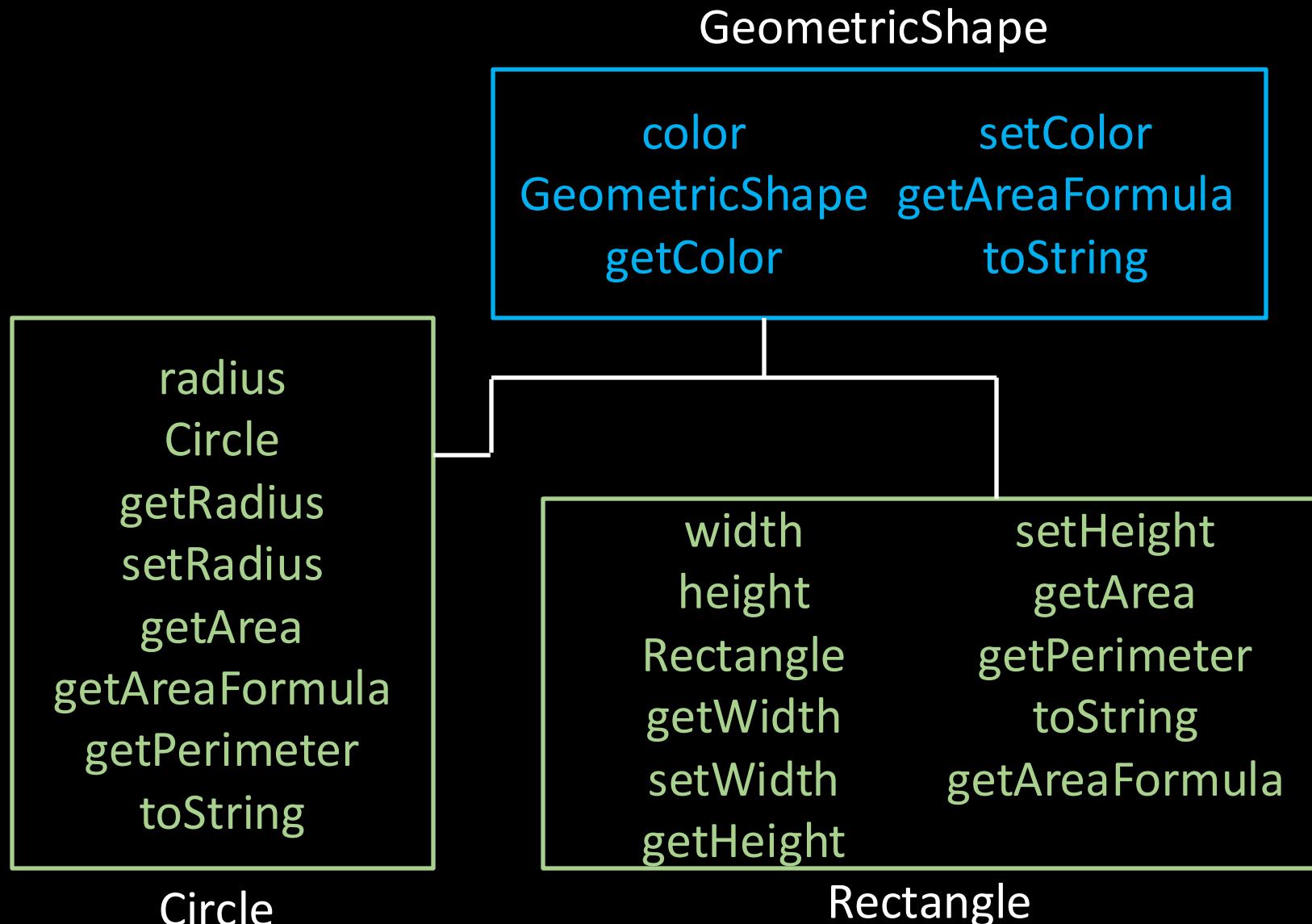


Polymorphism

Recall: Inheritance Defined

- Often, we want to create objects with similar behavior
 - These objects share some of the same instance variables and methods
- In Java, we can create a superclass that contains methods to be shared across other subclasses
 - Subclasses *extend* the functionality of a superclass with additional instance variables, constructors, and methods
 - Subclasses **inherit** the constructors and methods of the super class
- Inheritance is our second main pillar of Object-Oriented Programming

GeometricShape as a Superclass



See example code
PolymorphismDemo
in lec14-1a

Polymorphism: An Example

- The `PolymorphismDemo` class defines a method for `GeometricShape` objects
- `Circle` and `Rectangle` objects can be used in place of `GeometricShape` objects!
- We can call the same methods `getColor()` and `getAreaFormula()` for all three types of objects
 - Polymorphism – the methods (may) have different behavior based on the object being used

Poll Everywhere: Question 1

Complete this sentence:

For Circle objects, we used the getColor() method declared in the _____ class and the getAreaFormula() method defined in the _____ class.

- A) Circle, Circle
- B) Circle, GeometricObject
- C) GeometricObject, Circle
- D) GeometricObject, GeometricObject

The Object Class

Object

- The Object API can be found [HERE](#)
- All classes *extend* the generic Object class
 - If a given class can create an object, then the object has all of the methods available in the Object class
 - E.g. see equals(), hashCode(), etc in the `MinionObject` class
 - Classes still extend the Object class, even if they do not have a constructor

See example code for `MinionObject` in lec14-1b

Abstraction

Abstract Classes and Methods: Motivation

- In the **GeometricShape** example, we declared a specific method for `getAreaFormula()`
 - This is awkward because an area formula is not defined for any generic shape
 - we need to know what the shape object is
- Abstract classes and methods allow us to build a framework for implementing methods, similar to an interface
 - Abstract classes cannot be used to make objects
 - Abstract methods don't have a body

Declaring Abstract Classes and Methods

- To declare an abstract class, we just put abstract in the header:

```
public abstract class [class name]
```

- To declare an abstract method, we use the following syntax

```
public abstract [output type] [method name];
```

Abstract Method Example

- See abstract class **GeometricShape**
 - Class has abstract methods `getArea` and `getPerimeter`
 - Abstract methods are declared in the subclasses **Circle** and **Rectangle**
- Try it for yourself: what happens if you remove the abstract methods from the **GeometricShape** class?

Poll Everywhere: Question 2

In example 14-1c, we checked areas using the following line:

`equalArea(rectangle,circle)`

Complete this sentence:

In the above method, the `getArea()` method is declared as an abstract method in the _____ class.

- A) Circle
- B) Rectangle
- C) GeometricShape

4 Pillars of OOP

Object-Oriented Programming

Encapsulation



Inheritance



Polymorphism



Abstraction



Participation Exercise 14-1a: **PolymorphicBooks**

Goal: Write a method that can operate on objects of a superclass or a subclass in the same way

```
Printing the objects with the displayTitle method:  
Book[Title:The Lorax]  
TravelBook[Title:Blue Planet Guides:Costa Rica]  
FictionSeriesBook[Title:Harry Potter: The Prisoner of Azkaban]
```

Expected output of the **PolymorphicBooks** class

Key take-away for this exercise: when getTitle() is called, where is it declared?

Codecheck Link: [HERE](#) and on Canvas

Participation Exercise 14-1b: **CompareEnergyProduction**

Goal: Write an abstract method
to call methods between
disparate classes

The SolarPanel object:
SolarPanel[Energy Production:265.20MW]

The WindTurbine object:
WindTurbine[Energy Production:270.00MW]

Is the energy production similar?
true

Expected output of the **CompareEnergyProduction** class

Codecheck Link: [HERE](#) and on Canvas