# CS 2050
# Computer Science II

Lesson 02

METROPOLITAN STATE UNIVERSITY℠
OF DENVER
LIVES TRANSFORMED

# Agenda

- Types in Java
- Classes vs. Objects

# Types in Java

- Types in Java can be:
  - Primitive (also called built-in) or
  - Reference

# Types in Java

- Primitive types are simpler types and are built-in the Java programming language
- They provide an efficient way to declare variables to represent numbers, single characters, and true/false
- Examples: int, double, char, boolean

# Types in Java

- Reference types are associated with classes
- A class in OOP Languages is the most fundamental mechanism to build Abstract Data Types (ADTs)
- ADTs allow programmers to build concepts to model things related to the problems presented to them

# Types in Java

- A class type puts together in a single entity values (also called properties) and functions (called methods)
- Properties represent attributes and states, while methods are associated with code

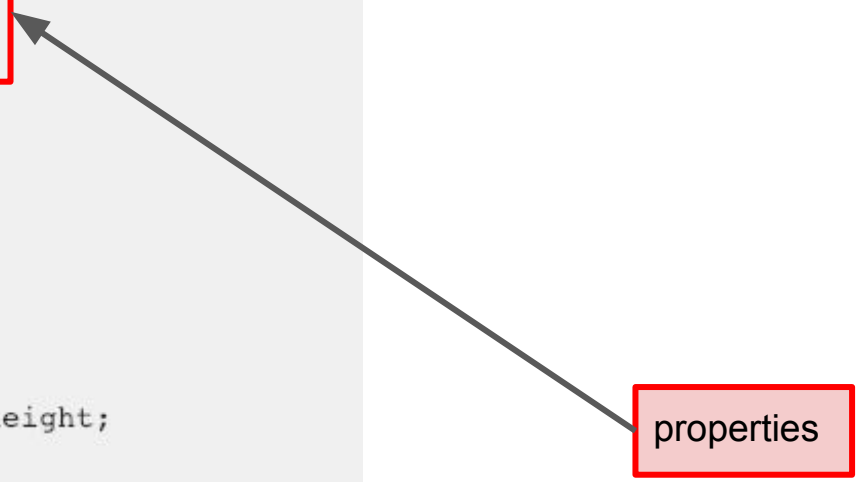
# Types in Java

- For example, consider a Rectangle class
- A rectangle is a quadrilateral with four right angles
- A rectangle is defined by its length and width
- Given a rectangle we are interested in computing its area, perimeter, and diagonal.

```
class Rectangle {

   // member variables (a.k.a. fields)
   int width;
   int height;

   // constructors
   Rectangle(int width, int height) {
     if (width <= 0)
       this.width = 1;
     else
       this.width = width;
     this.height = height <= 0 ? 1 : height;
   }

   Rectangle() {
     width = height = 1;
   }

   // methods
   double area() {
     return width * height;
   }

   double perimeter() {
     return 2 * width + 2 * height;
   }

   double diagonal() {
     return Math.sqrt(width * width + height * height);
   }
 }
```

A rectangle class definition

```
class Rectangle {

  // member variables (a.k.a. fields)
  int width;
  int height;

  // constructors
  Rectangle(int width, int height) {
    if (width <= 0)
      this.width = 1;
    else
      this.width = width;
    this.height = height <= 0 ? 1 : height;
  }

  Rectangle() {
    width = height = 1;
  }

  // methods
  double area() {
    return width * height;
  }

  double perimeter() {
    return 2 * width + 2 * height;
  }

  double diagonal() {
    return Math.sqrt(width * width + height * height);
  }
}
```




# A rectangle class definition

```
class Rectangle {

  // member variables (a.k.a. fields)
  int width;
  int height;

  // constructors
  Rectangle(int width, int height) {
    if (width <= 0)
      this.width = 1;
    else
      this.width = width;
    this.height = height <= 0 ? 1 : height;
  }

  Rectangle() {
    width = height = 1;
  }

  // methods
  double area() {
    return width * height;
  }

  double perimeter() {
    return 2 * width + 2 * height;
  }

  double diagonal() {
    return Math.sqrt(width * width + height * height);
  }
}
```

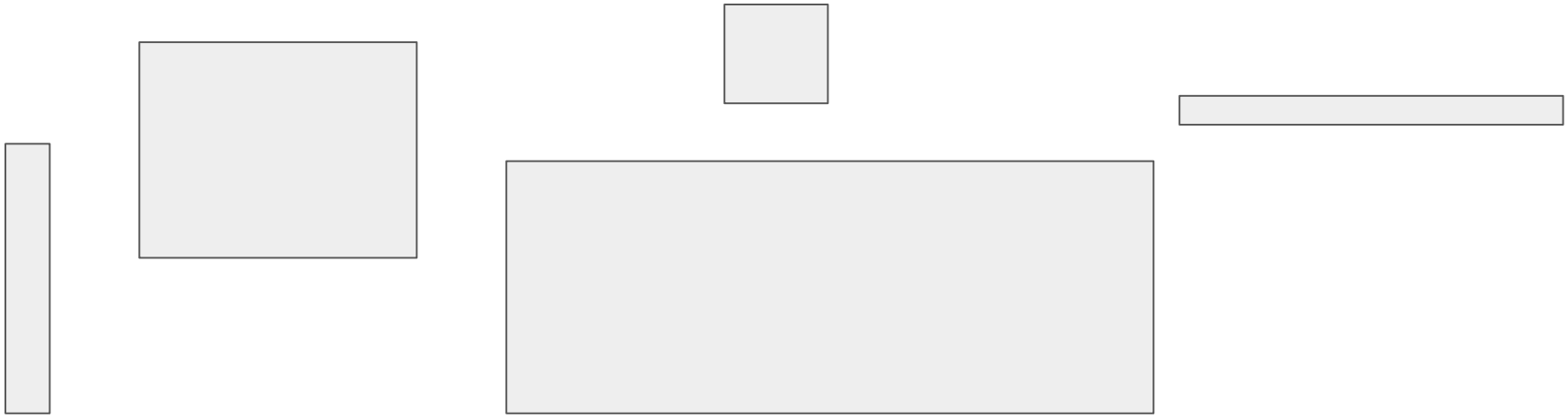
properties

methods

# A rectangle class definition

# Classes vs. Objects

- From a class you can create as many objects as you want
- Think of a class as a model for objects



# Classes vs. Objects

- Objects are created from classes in a process called “object instantiation”
- We use the “new” operator in Java for “object instantiation”
- Examples:

```
new Rectangle(10, 5)

new Rectangle(2, 8)
```

# Classes vs. Objects

- After its instantiation, an object is created in a specific location in your computer's memory
- In order for you to be able to use objects in your program you need to save a "reference" to the location where you those objects were created
- References to objects are saved in variables declared using the object's class type

# Classes vs. Objects

- For example:

```
Rectangle a = new Rectangle(10, 5)

Rectangle b = new Rectangle(2, 8)
```

- Now that you have saved the references for your rectangle objects you can start using them

# Classes vs. Objects

- For example:

```
Rectangle a = new Rectangle(10, 5)

Rectangle b = new Rectangle(2, 8)

System.out.println("Area of a is " +
a.area());

System.out.println("Perimeter of b is " +
b.perimeter());
```