

# Classes and UML

## Model 1: Die objects

When you define a class in Java, you are defining a new reference type. For example, the class below represents Die objects. Each object has its own attributes (data) and methods (code).

```
/**
 * Simulates a Die object.
 *
 * @author Chris Mayfield
 * @version 11/09/2015
 */
public class Die {

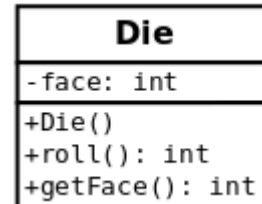
    private int face;

    /**
     * Constructs a new die with a random face value.
     */
    public Die() {
        roll();
    }

    /**
     * Simulates the roll of the die.
     *
     * @return new face value of the die
     */
    public int roll() {
        face = (int) (Math.random() * 6) + 1;
        return face;
    }

    /**
     * Gets the current face value of the die.
     *
     * @return current face value of the die
     */
    public int getFace() {
        return face;
    }

}
```



## Questions (10 min)

1. What are the attributes of Die? What are the methods?
2. In the class diagram, what do the - and + symbols represent?

3. Notice how the `roll` method refers to `face`, yet that variable is not declared in the method. Why does that work? What does the `roll` method change, in terms of the `Die` object?

4. Let's say we are in another class with a `main` method, and we want to declare a `Die` object named `lucky`. What would that *declaration* statement look like?

5. What would the statement look like to *instantiate* the `Die` object? (Hint: use `new`)

6. When we instantiate an object, we are actually calling a *constructor*. This method has no return type and the same name as the class itself. What does the `Die` constructor do?

7. What's wrong with the following code fragment?

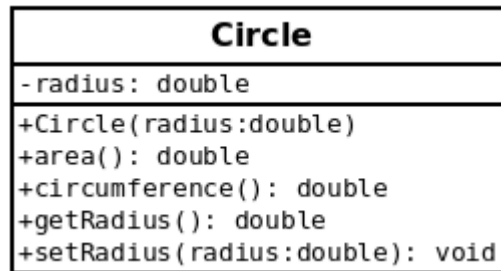
```
int value;  
value = Die.getFace();
```

6. We've decided to add a method to the `Die` class that will allow us to change the face value directly. What's wrong with the following implementation?

```
/**  
 * Sets the current face value of the die.  
 *  
 * @param newFace The new face value of the die  
 */  
public static void setFace(int newFace) {  
    face = newFace;  
}
```

## Model 2: Circle objects

Unified Modeling Language (UML) provides a way of graphically illustrating a class's design, independent of the programming language.



### Questions (15 min)

7. What are the attributes and methods of `Circle`, and what is their visibility?
8. Based on Model 1 and Model 2, what is typically public and what is typically private?
9. How would you declare a variable named `unit` that is a `Circle` object? How would you instantiate a `Circle` with radius 1 and assign it to `unit`?
10. Write a `main` method that creates a circle object with radius 2 and displays its area and circumference on the screen.

11. Draw the contents of memory at the end of your `main` method.

12. Write the complete code for the `Circle` class. When the value of `radius` is set, make sure it is not negative; if it is, set the radius to zero. Recall that the area of a circle is  $\pi r^2$  and the circumference is  $2\pi r$ .