Classes and UML

When you define a class in Java, you are designing a new type of object. Each object has its own copy of the variables and methods in the class.

Manager:	Recorder:
Presenter:	Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Define the terms: attribute, method, constructor, scope.
- Implement non-static methods based on a UML diagram.
- Distinguish static, instance, parameter, and local variables.

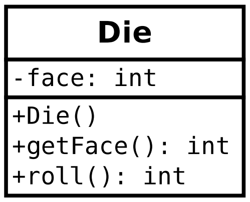
Process Skill Goals

During the activity, students should make progress toward:

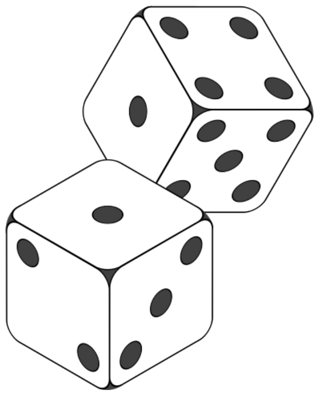
- Writing method signatures exactly as shown in a UML diagram. (Information Processing)

Model 1 The Die Class

The following class represents an individual "die" in a game of dice. The diagram on the right is a graphical summary of the *attributes* (variables) and *methods* of the class.



```java
/**
 * Simulates a die object.
 */
public class Die {

    private int face;

    /**
     * Constructs a die with face value 1.
     */
    public Die() {
        this.face = 1;
    }

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

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

}
```



Questions (10 min) Start time:

1. Consider the `Die` class:

a) What are the attributes?

b) What are the methods?

2. In the class diagram (on the upper right):

a) What do the `+` and `-` symbols represent?

b) What does the `:` represent?

3. Open the provided *Die.java* and run the program several times. Then answer the following questions about the `main` method:

a) What is the data type of `d1` and `d2`?

b) What are the initial values of the dice?

c) What method changed the dice values?

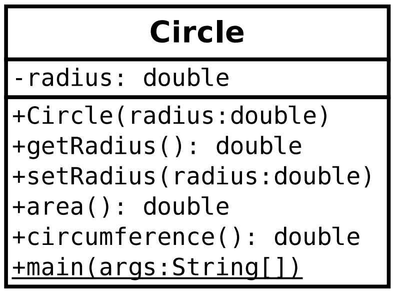
4. Write a statement that declares and initializes a `Die` variable named `lucky`.

5. When you create an object, Java invokes a ***constructor***. This method has no return type and has the same name as the class itself. What does the `Die()` constructor do?

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

Model 2 The Circle Class

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



```
Circle
-radius: double
+Circle(radius:double)
+getRadius(): double
+setRadius(radius:double)
+area(): double
+circumference(): double
+main(args:String[])
```

Questions (15 min) Start time:

7. Consider the `Circle` class:

a) How many attributes does the class have?

b) How many methods does the class have?

8. Based on Model 1 and Model 2, what is typically `public` and what is typically `private`?

The following questions will have you implement the `Circle` class exactly as shown in the UML diagram above. Do not worry about writing Javadoc comments for this activity.

9. Write the code that declares the `radius` attribute (above the first comment). An outline of *Circle.java* is provided below for context.

```
public class Circle {


    // constructor goes here

    // other methods go here
}
```

10. Write the code for the `Circle` constructor. Notice that, in contrast to Model 1, the `Circle` constructor has a parameter. Assign the parameter `radius` to the attribute `this.radius`.

11. Write the code for `getRadius`. (Refer to Model 1 for an example.)

12. Write the code for `setRadius`. Like the constructor, it should assign the parameter to the corresponding attribute.

13. Write the code for `area`. The area of a circle is πr^2. `PI` is in the `Math` class.

14. Write the code for `circumference`. The circumference of a circle is $2\pi r$.

15. Write a `main` method that creates a `Circle` object with a radius of 2.0 and displays its area and circumference (using `println`).

Model 3 Variable Scope

As a team, review and discuss the provided *SwapCircle.java* and *SwapDriver.java* source files. Then identify the ***scope*** of each variable (i.e., where it can base used) based on the table below.

	Where declared?	**Where used?**	**Example**
static variables (“class variables”)	declared outside of all methods (typically at the start of the class)	anywhere in the class (or in other classes if also `public`)	`circleCount` in the `SwapCircle` class
instance variables (“attributes”)	declared outside of all methods (typically after any static variables)	any non-static method (or in static methods when another object has been created)	`radius` in the `SwapCircle` class
parameters	declared inside the ()’s of a method signature	anywhere within the method where they are declared	`radius` in the `SwapCircle` constructor
local variables	declared inside a method (or inside another block of code, like a `for` loop)	anywhere within the method or code block after they are declared	`temp` in the `swapInts` method

Questions (20 min) Start time:

16. Identify one static variable from the `SwapCircle` class.

a) What is the name and purpose of the variable?

b) What is the scope of the variable?

c) What is one example of somewhere it cannot be used?

17. Identify one instance variable from the `SwapCircle` class.

a) What is the name and purpose of the variable?

b) What is the scope of the variable?

c) What is one example of somewhere it cannot be used?

18. Identify one parameter from the `SwapCircle` class.

a) What is the name and purpose of the variable?

b) What is the scope of the variable?

c) Where can the variable not be used?

19. Identify one local variable from the `SwapCircle` class.

a) What is the name and purpose of the variable?

b) What is the scope of the variable?

c) Where can the variable not be used?

20. Run the `SwapDriver` program and summarize what you learn based on the output.

21. Notice that `getRadius` returns `this.radius` (from `this` object). In contrast, `getCircleCount` does not use the keyword `this`. Why not?

22. Identify an example of where an instance variable is used within a static method.

a) In which method does this occur?

b) Why is the method able to access these instance variables, even though they are private?

c) Explain how this method is not a violation of the rule that instance variables cannot be accessed inside a static method.