Classes and UML

The `String` class provides methods for working with text. The `Random` class provides methods for generating random numbers. In this activity, you'll learn how to make your own classes that represent everyday objects.

Content Learning Objectives

After completing this activity, students should be able to:

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

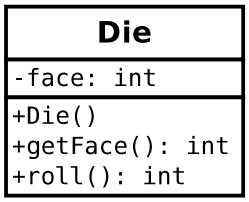
Process Skill Goals

During the activity, students should make progress toward:

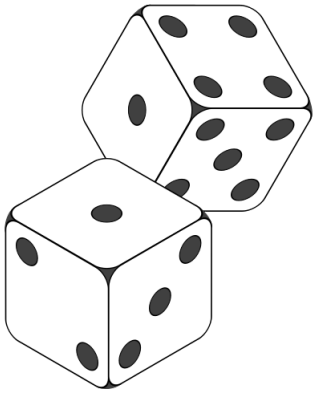
- Writing method signatures exactly as shown in a UML diagram. (Information Processing)

Model 1 The Die Class

When you define a `class` in Java, you are defining a new type. Classes have ***attributes*** (data) and ***methods*** (code). A ***class diagram*** is a graphical summary of the attributes and methods.




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

    private int face;

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

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

    /**
     * Simulates the roll of 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. What are the attributes of `Die`? What are the methods?

2. In the class diagram, what do the - and + symbols represent? What does the : represent?

3. Write a statement that ***declares*** a `Die` variable named `lucky`.

4. Each ***instance*** of a class (in memory) is called an object. Write a statement that ***instantiates*** a `new` `Die` object and assigns it to `lucky`.

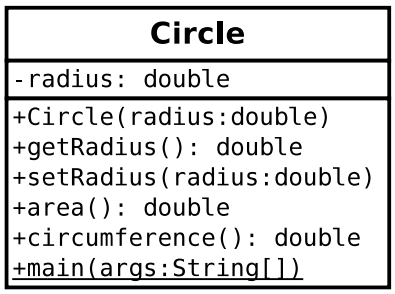
5. When you instantiate an object, you invoke a ***constructor***. This method has no return type and has the same name as the class itself. What does the `Die` constructor do?

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

7. What is the purpose of the `getFace` method? Show how you would use it in a `main` method of another class.

Model 2 The Circle Class

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



Questions (15 min) Start time: _____

8. What are the attributes and methods of `Circle`, and what is their ***visibility***?

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

10. How would you declare a variable named `unit` that is a `Circle` object? How would you instantiate a circle with a radius of 1.0 and assign it to `unit`?

11. Write the code (inside `Circle.java`) that declares the `radius` attribute.

12. Write the code for `getRadius`. (Don't worry about Javadoc comments for this activity.)

13. Write the code for `setRadius`. Note there are two variables named `radius`: the parameter of `setRadius`, and `this.radius` for the object itself. Before you set the radius, first check if the parameter is negative, and if it is, set `this.radius` to zero instead.

14. Write the complete code for `area` and `circumference`. The area of a circle is πr^2, and the circumference is $2\pi r$. Ideally, each method should be one line of code.

15. Write a `main` method that creates a `Circle` object with a radius of 2.0 and displays its area and circumference on the screen.

Model 3 Variable Scope

As a team, first review and discuss the `SwapCircle` and `SwapDriver` classes found at the end of the questions. Then identify the ***scope*** of each variable 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 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.

19. 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?

20. 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?

21. In the space below, predict the output of the `SwapDriver` program. Why are the results different when swapping integers and swapping objects?

SwapCircle.java

```java
public class SwapCircle {

    private static int circleCount = 0;

    private double radius;

    public SwapCircle(double radius) {
        circleCount++;
        if (radius > 0) {
            this.radius = radius;
        } else {
            this.radius = 1;
        }
    }

    public static int getCircleCount() {
        return circleCount;
    }

    public double getRadius() {
        return this.radius;
    }

    public static void swapInts(int x, int y) {
        System.out.println("\tInside swapInts");
        System.out.println("\tswapping integers " + x + " and " + y);
        int temp = x;
        x = y;
        y = temp;
        System.out.println("\tfinished swapping " + x + " and " + y);
    }

    public static void swapCircles(SwapCircle c1, SwapCircle c2) {
        System.out.println("\tInside swapCircles");
        System.out.println("\tswapping circles " + c1 + " and " + c2);
        double r = c1.radius;
        c1.radius = c2.radius;
        c2.radius = r;
        System.out.println("\tfinished swapping " + c1 + " and " + c2);
    }

    public String toString() {
        return String.format("Circle(%.0f)", this.radius);
    }
}
```

SwapDriver.java

```java
public class SwapDriver {

    public static void main(String[] args) {

        // first try swapping integers
        int a = 7, b = 4;
        System.out.println("BEFORE swap:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        SwapCircle.swapInts(a, b);
        System.out.println("AFTER swap:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println();

        // next try swapping SwapCircle radii
        SwapCircle first, second;
        first = new SwapCircle(7);
        second = new SwapCircle(4);
        System.out.println("BEFORE swap:");
        System.out.println("first = " + first);
        System.out.println("second = " + second);
        SwapCircle.swapCircles(first, second);
        System.out.println("AFTER swap:");
        System.out.println("first = " + first);
        System.out.println("second = " + second);
        System.out.println();

        System.out.printf("This program created %d circles",
                          SwapCircle.getCircleCount());
        System.out.println();
    }
}
```