Activity 11: Objects

Previously we explored how classes define attributes and methods. Static variables and methods apply to the whole class, whereas non-static variables and methods apply to specific objects.

Model 1 Variable Scope

As a team, review and discuss the `Circle` and `SwapDriver` classes found on the next page at the end of the questions. 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 `Circle` 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 `Circle` class
parameters	declared inside the ()'s of a method signature	anywhere within the method where they are declared	`radius` in the `Circle` 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: _____

1. Identify one static variable from the `Circle` 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?

2. Identify one instance variable from the `Circle` 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?

3. 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.

4. Identify one parameter from the `Circle` 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?

5. Identify one local variable from the `Circle` 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?

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

```java
public class Circle {

    private static int circleCount = 0;

    private double radius;

    public Circle(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(Circle c1, Circle 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);
    }
}
```

```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);
        Circle.swapInts(a, b);
        System.out.println("AFTER swap:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println();

        // next try swapping Circle radii
        Circle first, second;
        first = new Circle(7);
        second = new Circle(4);
        System.out.println("BEFORE swap:");
        System.out.println("first = " + first);
        System.out.println("second = " + second);
        Circle.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",
                          Circle.getCircleCount());
        System.out.println();
    }
}
```

Model 2 Class Design

Classes are often used to represent abstract data types, such as `Color` or `Point`. They are also used to represent objects in the real world, such as `CreditCard` (see next page) or `Person`.

Color
-red: int -green: int -blue: int
+Color(red:int,green:int,blue:int) +Color(other:Color) +add(other:Color): Color +darken(): Color +equals(obj:Object): boolean +lighten(): Color +sub(other:Color): Color +toString(): String

Point
-x: int -y: int
+Point() +Point(x:int,y:int) +Point(p:Point) +equals(obj:Object): boolean +getX(): int +getY(): int +setX(x:int): void +setY(y:int): void +toString(): String

Classes generally include the following kinds of methods:

- **constructor** methods that initialize new objects
- **accessor** methods (getters) that return attributes
- **mutator** methods (setters) that modify attributes
- **object** methods such as `equals` and `toString`

Questions (15 min)

Start time: _____

7. Identify the constructors for the `Color` class. What is the difference between them? What arguments do they take? What do these methods return?

8. Identify an accessor method in the `Point` class.
   a) Which instance variable does it get?
   b) What arguments does the method take?
   c) What does the method return?

9. Identify a mutator method in the `Point` class.
   a) Which instance variable does it set?
   b) What arguments does the method take?
   c) What does the method return?

For the remaining questions, you will design a class that represents an individual's credit card.

10. List two or more attributes that would be necessary for this `CreditCard` class. For each attribute, indicate what data type would be most appropriate.

11. When constructing (or updating) a `CreditCard` object, what values would you need to check? What are the valid ranges of values for each attribute?

12. List two accessor methods would be appropriate for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.

13. List two mutator methods would be appropriate for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.