Programs as Models

- remember: we write programs to solve real-world problems
- programs act as models of the real-world problem to be solved
- one way to model problems is to use a mathematical approach
  - model the problem using variables and formulas/functions
- so far, all of our programs have used this approach

Procedural Paradigm

- known as the Procedural Programming Paradigm
  - under this paradigm, programs consist of two basic building blocks:
    1. data
    2. methods

Class Methods

- Java implements the procedural programming paradigm using class methods
  - method header contains the keyword static
  - invoked using ClassName.methodName...
- so far all of our methods have been class methods
Class Methods (cont)

✴ class methods work fine for:
  • mathematical functions
    EX: Math.sin
  • methods associated with a single object
    EX: Input class, where all input comes from the keyboard (not from multiple sources)

EX: Cylinder

✴ using this approach, we can model a cylinder using:
  • two variables representing the cylinder’s height and radius
  • class methods that calculate the surface area, volume, etc. for a given height and radius (passed as arguments)

Multiple Cylinders

✴ works fine for one cylinder
✴ what if we have two cylinders?
  • need variables for each:
    height1 and radius1
    height2 and radius2
✴ but what if we have two thousand cylinders?
✴ what if each also has a weight, a name, a color, a location, etc.?

Objects

✴ need a better way to model the real world
✴ when we look around us, we don’t see data and methods
✴ we see objects that have:
  1. attributes: associated data values
  2. behavior: what the object can do
EX: Student

For example, each student has
- attributes: name, ID#, age, grades, etc.
- behavior: talk, walk, take quiz, etc.

Many Students

- Each student has different attribute values
  - Bob has ID#555
  - Mary has ID#101
- Each student has independent behavior
  - Bob talks, while
  - Mary takes a quiz

OOP

- When used in programming this approach to modeling is called the Object Oriented Programming (OOP) Paradigm
- Basic building blocks of programs are objects, each consisting of attributes and behavior
- Java is an OOP language

Class

- Want each student to be represented by a “student object” in our program
- In Java we do this by expanding our definition of what a class is
- Class (new definition): a blueprint or template for creating objects
Classes & Objects

払い each class defines the attributes and behavior associated with an object of that class

・every object is an instance of some class
・the attributes associated with an object are called instance variables
・the behaviors associated with an object are defined by methods called instance methods

払い classes are like data types for objects

Using Objects

払い how do we use objects in Java?
払い for our examples, let's assume there exists a class named Square that specifies the attributes and behavior of a square object

Creating an Object Variable

払い creating a variable that represents an object requires two steps:

1. declaration - creates a variable that will refer to the object
2. instantiation - creates the actual object

Variable Declaration

払い syntax:

```
ClassName variableName;
```

払い EX: 

```
Square sqr;
```

払い assigns the label sq to a memory location
払い tells Java that this location is going to refer to an instance of the Square class
Object Instantiation

- **syntax:**
  
  ```java
  variableName = new ClassName();
  ```

- **EX:**
  
  ```java
  sqr = new Square();
  ```

  - creates an instance of the `Square` class
  - makes the variable `sqr` refer to the new `Square` object

One Step

- declaration and instantiation can be done in a single statement

- **syntax:**
  
  ```java
  ClassName variableName = new ClassName();
  ```

- **EX:** create two `Square` object variables

  ```java
  Square s1 = new Square();
  Square s2 = new Square();
  ```

Instance Methods

- once we have an object, we can invoke instance methods on it

- instance methods act on a specific object

- **syntax:**
  
  ```java
  objectName.methodName(arguments);
  ```

  - the available instance methods are given in the class API

Class API

- a class API can contain three different types of methods:
  1. constructors - will discuss in next module
  2. class methods - have keyword `static` in method header
  3. instance methods - method header exactly like class methods except no `static`
Square Class

- the Square class API is attached
- note it contains no class methods
- EX: Create two Square objects. Change the color of one to red, and the other to blue.

```java
Square sqrA = new Square();
Square sqrB = new Square();
sqrA.changeColor("red");
sqrB.changeColor("blue");
```

Shape Example

- three shape classes (Square, Circle, and Triangle) are provided
- ShapeDemo.java is a program that demonstrates how to use these classes
- the compiled shape class files must be placed in the same folder with ShapeDemo.java
- spend some time understanding this Example

Module 18 Vocabulary

- procedural programming paradigm
- class method
- object
- attribute
- object oriented programming (OOP) paradigm
- class
- instance
- instance variable
- instance method
- instantiation

Questions?

Email Kevin at sahrk@sou.edu