Classes and Methods Part I

writing your own classes and methods

data scope and visibility

WRITING YOUR OWN CLASSES

Writing Classes

- Classes define a blueprint of what all objects of that class will look like.
- Classes define:
  - Characteristics (state)
    - Instance data variables
  - Functionality (behaviors)
    - Methods

Practice

- Write a class to represent a six-sided die.
  - What is the state?
  - What is the behavior?

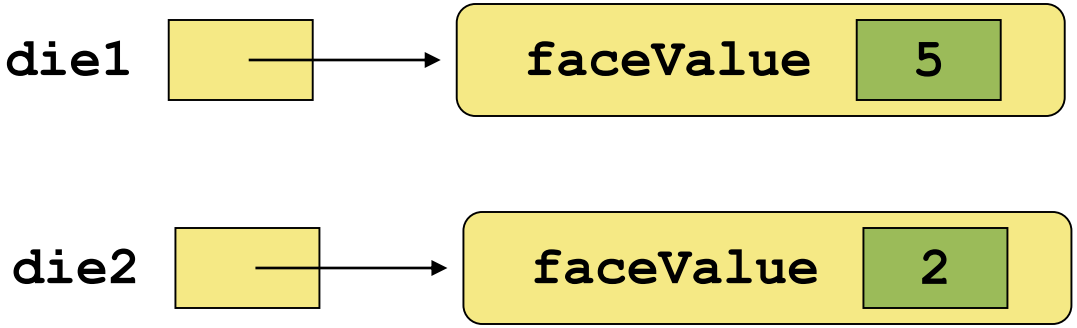
Class Components

- instance data variables
  - characteristics of objects
- constructor
  - sets up an object
- getters and setters
  - access to the instance data variables
- `toString` method
  - provide a text representation
- other class-specific methods

Instance Data

- Represents the characteristics of the object
- Declared within a class but outside of methods
  - Can be accessed by all methods within the class
- Each object has its **own copy** of instance data variables

Instance Data (cont.)



Constructor

- Sets up the object
- Called when an object is created
- Has the same name as the class
- Has **no** return type
- A *default constructor* is a constructor that has no parameters
  - If you don't define *any* constructor for a class, the default constructor will be created automatically behind-the-scenes by Java

Methods

- A method *declaration* defines or declares a method.
- A method *invocation* or *call* makes the method run.

Method Declaration

- To declare a method, specify the *method header*:
  - visibility
  - return type
  - method name
  - formal parameters
    - type and name of each parameter
  - *(static, when appropriate… more to come on this)*
- Method body is surrounded by { }

Method Declaration

- Visibility
  - public: can be invoked from anywhere
  - private: can only be invoked from within the class
- Return type
  - void: nothing is returned
  - class: an object is returned
  - primitive: a primitive value is returned
- Formal parameters
  - type and name of each parameter

Method Declaration Examples

```
public void printDescription() {
   …
}

private int calculate(int x1, int x2) {
   …
   return …;
}

public String getName() {
   …
   return …;
}
```

Method Declaration Examples

```
public void printDescription() {
   …
}

private int calculate(int x1, int x2) {
   …
   return …;
}

public String getName() {
   …
   return …;
}
```

no formal parameters

if return type is not void, must have a return statement

`return` Statement

- `void` methods do not return a value

- Methods not declared void must contain a `return` statement
  - Must return a value that matches the return type

Getters and Setters

- Methods that provides access to the instance data
- Also called *accessors* and *mutators*
- Getter
  - Returns the current value of a variable
- Setter
  - Updates the value of a variable
  - Can define rules for valid values

Getters and Setters (cont.)

- Getter

```
public TYPE getVARNAME() {
  return VARNAME;
}
```

- Setter

```
public void setVARNAME(TYPE NEWVALUE) {
  VARNAME = NEWVALUE;
}
```

instance variable

`toString` Method

- Returns a text representation of the object
- Automatically called when the object is concatenated with a `String` or put inside a `println` method
- Has this exact header:

```
public String toString() {
  return STRINGVAL;
}
```

DATA SCOPE

Data Scope

- The *scope* of a variable is the area where it can be used or referenced
  - Determined by where the variable is declared
- Instance data
  - Declared in the class and outside of any method
  - Used anywhere in the class
  - Lives as long as the object lives
- Local data
  - Declared inside of a method
  - Used only in that method
  - Dies when the method ends

Encapsulation

- Each objects protect its own information (instance data)
- Classes provide methods to appropriately access information

Encapsulation (cont.)

- A client is a class that uses another class
- Changes to an object's state (instance data) are made only through methods
  - Clients cannot access instance data directly
- A client is only allowed access to an object's information through provided methods
  - Client can request services by invoking methods
  - Client does not know how methods are implemented

Visibility Modifiers

- public
  - Can be referenced anywhere
- private
  - Can be referenced within a class
- protected
  - Related to inheritance… more to come later…
- default
  - can be referenced within the package

Visibility Modifiers

use just these for now

- public
  - Can be referenced anywhere
- private
  - Can be referenced within a class
- protected
  - Related to inheritance… more to come later…
- default
  - can be referenced within the package

Public Variables… DON’T DO IT!

- Public instance variables violate encapsulation
  - Allow a client to modify values directly
- Instance variables should only be modified through provided methods
- Keep instance data variables private!

Constants

- It’s okay to make constants public because they cannot be changed
  - No way to violate encapsulation

Method Visibility

- Public methods can be invoked by clients
  - Also called *service methods*
- Support or helper methods assist a service method
  - Not intended to be used by a client
  - Should be declared private

Visibility Modifiers

	Private	Public
Variables	support encapsulation	violate encapsulation
Methods	support methods within the class	provide services to clients

Summary of Classes

- Blueprints of what objects look like
- Contain:
  - private instance variables
  - constructors
  - public getters and setters
  - toString method
  - public and private methods

WRITING METHODS

Writing Methods

- Think about:
  - return type (output)
  - parameters (input)
  - name (follow conventions!)
  - visibility (who will need this?)

Practice: Planning Methods

- Write a method to print out common mathematical formulas.
- Write a method to calculate the area of a triangle with integer base and height.
- Write a method to calculate the distance between two points.
- Write a method to determine whether a triangle is a right triangle (i.e., it satisfies the Pythagorean theorem).

Practice: Planning Methods

- Write a method to print out common mathematical formulas.

  public void printFormulas()

- Write a method to calculate the area of a triangle with integer base and height.

  pubilc double calculateArea(int base, int height)

- Write a method to calculate the distance between two points.

  public double calculateDistance(int x1, int y1, int x2, int y2)

  public double calculateDistance(Point p1, Point p2)

- Write a method to determine whether a triangle is a right triangle (i.e., it satisfies the Pythagorean theorem).

  public boolean isRightTriangle(int sideA, int sideB, int sideC)

Practice

- Add a roll method to the Die class.
  - For now, it’s going to be a very predictable Die!

- Write a class to represent an item sold at an audio store (e.g., music, audio book, etc.).
  - All items sold are described by title, price, duration.

Practice

- Write an Employee class that represents employees at a company.
  - An employee is represented by a name, id, and phone number.