Intro to Data Structures

Lecture #5 – Classes, Objects, and OOP
September 2, 2014

Mark Stehlik

Outline for Today

- HW1 out (read your email!)
- OOP terminology – classes and objects
- Let's look at the Dice class (handout)
- Using objects created from the Dice class
- Creating another class
- Thursday – quiz (no arrays)

Object-oriented programming

- Objects allow us to collect related data and the methods that operate on that data into one entity
  - data -> fields ([common] attributes)
  - methods -> actions (what an object can do)
  - Ask not what you can do to an object, but what an object can do for you…
- Objects are *instances* of a *class*
- What classes/objects have we seen so far?

Object-oriented programming (visibility)

- Classes (and their parts) have *visibility modifiers*:
  - public: accessible to everyone
  - protected: inside class, inside package, inside subclass
  - *package-private* (default, no modifier used): inside class, inside package
  - private: accessible only within the class
- Data (attributes):
  - can be whatever you want/need for that object
  - usually **private**

Object-oriented programming (methods)

- Methods (actions):
  - constructors: create (instantiate) objects (instances) from the "blueprint" (the class)
  - "regular" methods that operate on/alter/display the data
  - visibility:
    - methods that are to be used outside the class (in other classes/by other objects) should be **public**
    - "helper" methods (used inside the class) should be **private**

Object-oriented programming…

- To create an object (an instance of a class), you call a constructor
  - new ClassName(parameters, if any)
- To call/invoke a method on a particular object:
  - object_reference.method(parameters)
- Let’s look at one that I wrote…

An example

The Dice class…

- fields/attributes? ⇒ instance variables
- “functions”/behavior? ⇒ methods

An example revisited

How does the existence of a Dice class change our DiceExperiment code?

As an aside: are there other possible pairs of 6-sided dice that have the same distribution when summed? Only 1 such pair exists (using non-negative numbers) – proved by Sicherman

Object-oriented programming (details)

- Method components:
  - modifiers (visibility, static/non-static)
  - return type
  - name
  - parameter list
  - body (definition)
- Method *signature*
  - name and parameter list

Object-oriented programming (details)

- Methods can be
  - *overloaded* (same name, different parameters -> different signature)
  - inherited (everything inherits from the Object class)
  - *overridden* (an inherited method with the same signature and return type, but different behavior)
    - always include a toString (and where can we find its sig?)
- The main method
  - every class can have a main, but at least one has to…

Object-oriented programming…

- Let’s create another class (Person)