U N I V E R S I T Y

CS 180: Problem Solving and Object-Oriented Programming

Lecture 9: Methods and Classes

Dr. Jeff Turkstra

Announcements

- Project 1 available today
- Dr. Turkstra office hours this Thursday 9/23 moved to 12pm-1:30pm (this week only)
- Midterm Exam 1 coming up on October 4 at 6:00pm
  - Available for 24 hours, two hours to complete after starting

Lecture 9

- Classes
- Variables
- Access modifiers
- Class variables and methods
- Constants
- Methods
- Constructors
- Scope and Extent


```
public class Converter {
  String convertToBinary(int n) {
    String result = "";

    // handle special cases...
    if (n < 0)
      return null; // failure
    if (n == 0)
      return "0";

    // loop while n > 0, accumulating a bit and dividing by 2...
    while (n > 0) {
      if (n % 2 == 0)
        result = "0" + result;
      else
        result = "1" + result;
      n = n / 2;
    }
    return result;
  }
}
```


Object

- Collection of some data along with pieces of code to manipulate said data
- In Java, an instance of a class

Class

- Basic building block in the OOP paradigm
- Templates that define state and behavior
  - Variables and methods
  - Mechanism(s) for instantiating the class
- An object is an instance of a class
- Some classes cannot be instantiated

Class declarations

- Modifiers (public, private, etc)
- Class name
  - Capital first letter
- extends followed by name of parent or superclass
  - Optional, but only one
- implements followed by list of interfaces
  - Optional, one or more
- Class body


```
class SomeClass extends ParentClass implements AnInterface {
  // class body
}
```


Class body

- Fields
- Constructor(s)
- Methods

Member variables

- Field: member variable in a class
- Local variable: variable in a method or block of code
- Parameter: variable in a method declaration

Field declaration

- Zero or more modifiers
- Type
- Name

```
public int anInt;
```


Access modifiers

- Determine if other classes can use a field or method
- Top (class) level
  - public or package-private (no modifier)
- Member level
  - public, private, protected, package-private

- public: visible to all classes everywhere
- package-private: visible only within its own package
  - Packages are named groups of related classes. More later.
- private: member can only be accessed in its own class
- protected: package-private and by subclass(es) in any package

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
No modifier	Y	Y	N	N
private	Y	N	N	N

Encapsulation

- Constructs that aid in the bundling of data with methods operating on that data
- Mechanism for restricting access to some of an object's components
  - Information hiding – abstraction
- Improves readability and manageability of code
- Procedural languages do not have these constructs

Encapsulation

- Fields are commonly marked private
- Accessor and mutator methods are used instead
  - E.g., public get/set
- Access levels help avoid errors from misuse
- Rule of thumb: use the most restrictive access level possible
  - private unless good reason not to
  - Avoid public fields (except for constants)

Class members

- Objects have their own copies of instance variables
  - radius in Wheel
  - Different memory locations
- Sometimes want a variable that is common across all objects
  - Called static fields or class variables
  - Associated with the class instead of an object
  - One memory location

Class variables

- Created by the static modifier
- Any object can change the value
- Can also be accessed without creating an instance

- Class variables should always be referenced using the class name `WheelStatic.numberOfWheels`

Class methods

- Java allows static methods as well as static variables
- Class methods should be invoked using the class name
  `ClassName.methodName(args)`

Access

- Instance methods can access instance variables and other instance methods
- Instance methods can access class variables and class methods
- Class methods can access class variables and class methods
- Class methods cannot directly access instance variables or instance methods
  - Must use an object reference
  - Also cannot use this

Constants

- Programs sometimes use constant values that should not be changed during execution
- Can designate a variable as unchangeable with final keyword
  - Can always initialize it
- Math.PI is a public static final double
- Convention dictates all caps for the name
- Often combined with static

Purdue Trivia

- Indiana Pi Bill
- In 1894, Indiana physician Edward J. Goodwin proposed a bill to state representative Taylor I. Record

  "A Bill for an act introducing a new mathematical truth and offered as a contribution to education to be used only by the State of Indiana free of cost by paying any royalties whatever on the same, provided it is accepted and adopted by the official action of the Legislature of 1897"
- Purdue University Professor Clarence. A. Waldo was able to educate many of the senators, preventing the bill from becoming law

Methods

- A parameterized block of code that may return a value
- Every method exists inside some class
- Reusability: reduce redundancy in code
- Readability: identify logical operation by name (abstraction)
- Modularity: can code and test independently

Method declarations

- Modifiers – public, private, etc
- Return type (can be void)
- Method name
- Parameter list
- Exception list (later)
- Method body

- Method signature: Method name & parameter types

```
modifiers return_type methodName(param_list) {
  statements;
  return if_needed;
}
```


Parameters vs. arguments

- Parameters allow a method to work on different data values
- Parameter: a variable in a method definition
- Argument: value or instance passed during runtime
  - Must match the number and types of parameters for a given method

Control flow

- No standardized calling convention
  - All done internal to JVM, unlike C
- In general:
  - Before call: argument values are copied to parameter variables
  - Calling method is "suspended"
  - Called method begins execution at top of method body, runs until return or end
  - Return value from called method is provided to callee
  - Calling method resumed

Pass by value

- Java passes arguments by value
- Even for objects
  - But, it's the reference that is passed by value
  - The object itself is not copied

Constructors

- Create objects from the class "blueprint"
- Similar to method declarations
  - Use class name
  - No return type
- Provided automatically if not specified
  - Be careful

Constructors

- Similar to a method
  - Not directly invoked
- Invoked by new
  - Or another constructor
- May access all class fields
- Does not explicitly return a value
- But, the new operator returns a reference

Scope

- Where in the code a variable is usable
- Basic rule: a variable is usable from the point of declaration to the end of the enclosing block
- Cannot have two variables with the same name active in same scope
  - Except: parameters and local variables can shadow fields with the same name

this Keyword

- Most commonly used if a field is shadowed by a method or constructor parameter
  - Look at wheel again
- Can also be used to call another constructor in the same class
  - Explicit constructor invocation
- Reference to the “current object”

Extent

- How long a value is maintained
  - Its “lifetime”
- Local variables: same as scope
- Parameters: created on method call, destroyed on return
- Objects: created by new, destroyed when garbage collected
  - Reference counting

Boiler Up!