**󀁸Chapter 3 Goals:**

* I understand the difference between abstraction and modularization in designing and writing a program
  + Abstraction is the process of ignoring the details of the parts and just looking at the big picture, or higher level of a program.
  + Modularization is the process of dividing a whole into well-defined parts that can be built and examined separately/breaking the problem down.
  + e.g abstraction: using a method as a single instruction but not looking at how that method is implemented step-by-step
  + e.g modularization: looking at how a method is implemented step-by-step
  + e.g modularization: separating a program into separate classes with separate methods to organize the code
* I can instantiate an object and understand the state of that object
  + Instantiate an object 🡪 Use constructor
  + State = current values of its fields
* I understand the difference between an object diagram and a class diagram
  + Object diagram: shows how objects interact with one another and the fields and their values and methods of the object
  + Class diagram: shows classes and their relationships, can show fields but will not show values
* I understand the difference between class and local variables, and when to use each.
  + Class variables: instance variables (fields), accessible throughout the entire class (class scope)
    - Should be used when they are required in more than 1 method of their values must be saved between calls to the class’s methods
  + Local variables: variables that are either parameters or defined in a method, accessible only within that method (block scope)
    - Should be used when they are only going to be used in 1 method (such as parameters)
    - Values do not need to be saved in between method calls
* I can write and use multiple constructors for a single class.
  + Constructors written with same name but different types and numbers of parameters
* I can create an instance of a class from within another class.
  + Make it a field
  + In constructor do:
    - hours = new NumberDisplay(24);
* I can make both internal and external method calls
  + Internal method call: methods call other methods of the same class
    - No variable name and a dot before the method name
  + External method call: method calls other methods of other objects using dot notation.
    - object.methodName(parameter-list)
* I know how to use dot notation to call methods from another class
  + See above (name of object variable . name of method (parameters))
* I understand the order of operations for logical and arithmetic operators
  + Logical: <, <=, >, >=, ==, !=, &&, ||, ^, !
  + Arithmetic: (), \*, /, %, +, -,
    - Multiplication, division, and modulus = left to right
    - Addition and subtraction = left to right
* I can use BlueJ debuggers to find and fix errors in a program
  + Set breakpoints
  + Click step button when stopped at a breakpoint to execute single line of code and then stop again
    - See path of execution
    - Abstraction
  + Stepping into method
    - See how the methods themselves execute step by step
* I can create a random number using Math.random()
  + int i = (int)(Math.random()\**number*) 🡪 0 to (number - 1)
  + int i = (int)(Math.random()\**number*) + 1 🡪 1 to number
* I can create an array of primitive data type and can understand the basic advantages of using arrays (page 251 in the text book)
  + Field: private int[ ] hour
  + Constructor: hour = new int[24]
  + Create array of character + populates it with all the uppercase letters
    - char[] alphabet = “ABCDEFGHIJKLMNOPQRSTUVWXYZ”.toCharArray
  + Basic advantage of using arrays: flexible but fixed capacity, useful when you know in advance exactly how many values you need to store
    - Access to items held in an array is often more efficient than access to the items in a comparable flexible-size collection
    - Arrays are able to store either objects or primitive-type values, flexible-size collections can only store objects
* I can use Scanner and its methods to read input from the keyboard and from a file (Java API).
  + Import: import java.util.Scanner;
  + Field: private Scanner scan;
  + Constructor: scan = new Scanner(text); 🡪 Reads file (text = new File(“TexasCountryPopulation.txt”);
  + Constructor: scan = new Scanner(System.in); 🡪 Reads input from keyboard

String answer = “ ”;

answer = scan.nextLine();

**Vocabulary**

Abstraction: focus on high level, ignore details

Modularization: divide the whole into smaller parts that all interact with one another, break down the problem

Object Creation: objects can create other objects, using the new operator (usually in the constructor), some objects cannot be constructed unless extra information is provided

Instantiation: the new keyword = Java operator that creates the object

Object Diagram: shows the objects and their relationships at one moment during the execution of an application, give info about objects at runtime and presents the dynamic view of the program

Class Diagram: shows the classes of an application and the relationships between them, gives information about the source code and presents the static view of the program

Primitive Type: non-object types (int, boolean, char, double, long, float)

Method Calls: invoking a method, can be internal or external, makes the objects do things, objects can communicate by calling each other’s methods

Local Variables: variables declared and used within one method, scope and lifetime are limited to that of the method

Arrays: a special type of collection that can store a fixed number of elements

Random Numbers: use (int)(Math.random()\**number*) in order to generate a random number 0 – (number -1)