Use the following outline to guide your self-assessment and notetaking

Week 2 – Introduction to Using Classes

Data Conversion (Ch 2.5)

Conversion techniques

- Can occur in 3 ways:
  - Assignment conversion
  - Promotion
  - Casting

- Assignment conversion is when a value is assigned to another value with a different type
  - Can convert int to float
    - Ex: int money = 25; to double amount = money (Amount's value is 25.0)
    - BUT this cannot work in reverse from double to int (needs casting)
- Promotion is when a type gets converted to a higher precedence level to perform certain operations
  - A floating-point value is divided by an int value. The int is converted to a float automatically before division to ensure the result is another float
  - String concatenation of numbers
- Casting is the most general conversion:
  - Convert float to int
    money = 25.52;
    dollars = int(money);
  - Note:
    - For this example,

      result = (float) total / count;

      The casting is performed on total and not the whole result because the division operator has lower precedence.

Creating Objects (Ch 3.1)

Diagramming variables

- Variables can be either declared or both declared and initialized
  - Initialized just means that the variable contains data
- The object type (class) is declared followed by the variable name
- Instantiation is when you create an object using the new operator
  - The new operator returns the memory address of that object
  - Dot operator is used to access methods of that object

Aliases

- More than one variable can hold the same address to a single object
- Changing the value in an object will mean that all references will access the current changed data
- When managing objects, a programmer must be careful because overriding one object with another will cause the 2 variables to have the same memory address
    - String a = “Hello”;
    - String b = “World”

      These have different memory addresses.

But after stating a = b, the variables both will refer to the same object and memory location. This does not occur with primitive data types.

The String Class (Ch 3.2)

Important methods

- charAt(index) – find character at the specified index
- compareTo(str) – find whether str is canonically before (negative return value), after (positive return value), or equal to (zero return value)
- concat(str) – return a new string that consists of this string + str
- equals(str) – true if the string contains the same characters as str
- equalsIgnorCase(str) – similar to equals but ignores the case of each character
- length() – number of characters
- replace(oldChar, newChar) – return a new string that replaces all instances of oldChar with the newChar
- substring(offset, endIndex) – return a new string that is a subset of the original at the offset index and extends towards endIndex – 1. By, default, endIndex is the length of the string – 1
- toLowerCase() – new string with all lowercase letters
- toUpperCase() – new string with all uppercase letters

Immutable means a string cannot be modified

Packages (Ch 3.3)

Import declaration

- Used to bring classes from other packages into your program
- Used to simplify referring to each class
- Import declarations will not work if 2 classes from different packages have the same name

The Random Class (Ch 3.4)

Pseudorandom means not actually random because a computer has to perform a complex calculation to generate anything random

Important methods

- nextFloat() – return random decimal number between 0 and 1 inclusive
- nextInt() – return random number that ranges over all possible integers
- nextInt(num) return random number in range 0 to num -1

Achieving given ranges

- To achieve a given range with Math.random() the formula is

  lower_bound + Math.random() * (upper_bound – lower_bound)

- A similar approach can be used with the Random class
  - lower_bound + Random.nextFloat() * (upper_bound – lower_bound)
  - lower_bound + Random.nextInt(upper_bound – lower_bound + 1)

    You need to add 1 with nextInt because the ending value is not inclusive

Formatting Output (Ch 3.6)

NumberFormat class

- Provides a way for formatting numbers
- Can be used to return a currency with the getCurrencyInstance() method or a percent with the getPercentInstance() method
- Not instantiated with the new operator, instead object is requested from one of the static methods invoked through the class name
  - Ex: NumberFormat.getCurrencyInstance()

DecimalFormat class

- This is instantiated through the new operator
- Can be used to apply a pattern to a DecimalFormat object
- Also can be used to convert a number to a string representing the number pattern

The printf method (this is the way we will use in this course)

Enumerated Types (Ch 3.7)

- Used as a type of variable when declared
- Establishes all possible values of a variable when declared and has no limit to the number of values
  - Ex: enum Seasons {fall, winter, spring, summer}
- Type-safe means that the variable can only take on values that correspond to a certain data type. Any other type results in a compile-time error

Wrapper Classes (Ch 3.8)

Autoboxing – Automatic conversion between a primitive value and a corresponding wrapper object

- Ex: int to Integer -> new Integer(69)
- Reverse conversion also exists from wrappers to primitives