Humna Sultan
Mr. Hendricks
AP CS A
25 September 2020

9/21-9/25 Work

3.1: Levels of Abstraction

- Blackbox: used to describe any device whose inner workings are hidden
    - Interaction with outside world is still defined
    - Provides encapsulation and hides unimportant details
- The most common approach for structuring computer programming is the object oriented approach
    - Blackboxes = objects
- Software is more flexible than hardware because it doesn't have physical restrictions

3.2: Specifying The Public Interface of a Class

- Operations are expressed as method calls
    - harrysChecking.deposit(2000);
- Mutator: modify values, do not return values
- Accessor: returns a value that can printed or stored
- Method body: statements that execute when methods are called
- Constructor: BankAccount harrysChecking = new BankAccount();
    - Same as class name
    - Initializes objects

3.3: Commenting the Public Interface

- Documentation comments: /** **/
    - Comment about classes and methods
- The javadoc utility copies the first sentence of each comment to a summary table in the HTML documentation.
- @param is used for methods with parameters, @return is used for methods with a return type that is not void

3.4: Instance Fields

- Instance fields: where objects store data
    - Instance of a class = object of a class
- Instance field = access specifier (usually private), type of instance field, name of the instance field
- Private = variables can only be accessed by methods of the same class

3.5: Implementing Constructors and Methods

- Constructors contain instructions to initialize the instance fields of an object.
- Constructors contain initialization statements for objects
- The return value becomes the value of the method call expression.
- The return statement is a special statement that instructs the method to terminate and return an output to the statement that called the method.

3.6: Unit Testing

- Testing in isolation, outside a complete program, is called unit testing.
- Two choices to test class
    - Use commands in IDE to construct objects and invoke methods
    - Create a tester class
- A tester class is a class with a main method that contains statements to run methods of another class.
    - 1. Construct one or more objects of the class that is being tested.
    - 2. Invoke one or more methods.
    - 3. Print out one or more results.
    - 4. Print the expected results.

3.7: Categories of Variables

- Instance fields belong to an object. Parameter variables and local variables belong to a method—they die when the method exits.
- The second major difference between instance fields and local variables is initialization. You must initialize all local variables.
- Parameter variables are initialized with the values that are supplied in the method call.
- Instance fields are initialized with a default value if you don't explicitly set them in a constructor.
    - Instance fields that are numbers are initialized to 0.

3.8: Implicit and Explicit Variables

- The implicit parameter of a method is the object on which the method is invoked.
    - The this reference denotes the implicit parameter.
- “this” refers to the object on which the method is called
- The implicit parameter is always this and is not explicitly stated
- Use of an instance field name in a method denotes the instance field of the implicit parameter.

3.9: Shape Classes

- You should make a class for any part of a drawing that occurs more than once
- Steps to making shapes:
    - 1. Determine the shapes that you need for the drawing.
    - 2. Find the coordinates for the shapes.
    - 3. Write Java statements to draw the shapes.
        - EX: Rectangle leftRectangle = new Rectangle(100, 100, 30, 60);
    - 4. Combine the drawing statements with the component "plumbing".
    - 5. Write the viewer class.
- The viewer class is completely routine; you only need to change a single line to show a different component.

★★ **Exercise P3.3.** Write a class `SavingsAccount` that is similar to the `BankAccount` class, except that it has an added instance field `interest`. Supply a constructor that sets both the initial balance and the interest rate. Supply a method `addInterest` (with no explicit parameter) that adds interest to the account. Write a `SavingsAccountTester` class that constructs a savings account with an initial balance of $1,000 and an interest rate of 10%. Then apply the `addInterest` method and print the resulting balance. Also compute the expected result by hand and print it.