recitation2.md

Recitation 2

September 16, 2021

Topics to Recap

- Rules about Identifiers
  - Identifiers can contain aA through zZ, as well as digits 0 through 9 and underscores
    - Don't start with a number!
  - variablesAndMethodNamesShouldLookLikeThis
  - ClassNamesLookLikeThis
  - CONSTANTS_LOOK_LIKE_THIS
    - but you don't need to worry about these
- Rules about Variable Updates
  - the left-hand side (LHS) must be a single variable that will store the result of the right-hand side (RHS)
  - the RHS can be determined in terms of the variable that you're updating
    - i.e. `x = x + 1` is OK
  - Variable updates capture the state of the variables at one point in time:
    - `x = y` doesn't change `y`, it merely puts the value held in `y` into variable `x`.
      - CHANGING `y` LATER DOES NOT THEN CHANGE `x`!
- Doubles
  - Like ints, but they express numbers with fractional parts
  - double literals are written as numbers with a decimal point (`5.9, 0.0, 1.0001`)
  - Most arithmetic works with both doubles and ints
    - Division works a little differently!
    - Modulo (`%`) is easier to reason about with ints than doubles
  - ints can automatically be promoted to doubles but not vice versa
    - `double seven = 7` is ok, `int eight = 8.0` is not.
- Methods
  - The syntax of a method
    - A **method call** is a statement that invokes an object's method, allowing the program to perform a particular operation on that object.
    - Calling a method on an object requires appending the "." operator and the method's name to the object's name. The "." operator is also known as the member access operator.
    - A class' **public methods** are the operations that a program can directly invoke on objects of that class type. The class' public methods are often called the **class interface**.
    - Programmers influence the behavior of methods by providing additional input values, called **method arguments**, in a method call. A method call is an invocation of a method's name, causing the method's statements to execute.
    - The **method's name** can be any valid identifier.
    - A **block** is a list of statements surrounded by braces.
    - Methods can also perform operations that return a value, known as a return value, to the program.
  - Learn about methods that are already written using javadocs for those methods
    - We did this with `StringTokenizer`
  - Some String methods:
    - `public char charAt(int index)`

        - name: charAt
        - return type: char
        - parameters: one int, representing the index of the string to look up
    - `public int length()`
        - name: length
        - return type: int
        - parameters: none!
    - `public String substring(int start, int stop)`
        - name: substring
        - return type: String
        - parameters: one int representing the start index of the substring and then one int representing the stop index.

Examples for Recap

1. You've been tasked with writing a class `BankAccount.java`
    - What are the necessary private fields? Write them out with good identifier style.
    - What are some public member methods that a `BankAccount` would need?
        - For each method you come up with, specify its name, return type, and what parameters it might take in.
2. List the names, return types, and parameter lists for all of the public methods of the `Car` class that you can interpret from its main method:

```
public class Car {
    // private fields withheld

    // public methods hidden ...

    public static void main(String[] args) {
        Car ferrari = new Car("Ferrari", 2019);
        Car buick = new Car("Buick", 1987);


        ferrari.unlock();
        ferrari.setGear("Drive");

        // accelerate to 50.5 mph over 3 seconds
        ferrari.accelerate(50.5, 3.0);
        double currentSpeed = ferrari.getSpeed()

        // prints "beep beep"
        buick.lock();

        ferrari.crash(buick);
    }
}
```

Recitation Problem Set

> Submit your answers to the following questions to Gradescope. You'll be awarded points primarily on completeness, although some degree of correctness and effort is necessary to earn credit.

GROUPS:

(Each row is a group of four)

Hu, Lucy Qian	Zhang, Han	Zhang, Yang	Williams, Levester Randall
Nguyen, Tai D	Ren, Yue	Ng, Wai Chung	Lai, Qimei
Guo, Zhaosen	Liu, Xinyue	Jiang, Yao	Hu, Yuxin
Choi, Jae Ho	Huang, Wenyi	Chheda, Shagun Pritesh	Yu, Qingyu
Wang, Kehan	Chen, Xiyue	Kallas Jatene, Rafael	Biscaro, Denise
Ye, Huifang	Lim, Xi Zhen	Carnation, Kayla Rae	Zhang, Zhihui
Patel, Siddharth Bhagwanji	Wu, Jeng-Ru	Wang, Liujia	Bernat, Kevin Bruno
Sheng, Xinyue	He, Donglun	Cai, Jialin	Bales, Elijah
Patel, Rishi	Arguello-Gonzalez, Marcos Abraham	Yiu, Hon-Cheung	Kim, Yunchae
Xiao, Zijian	Tims, George	Cheema, Sardar Asfandy	Cho, Suebin Grace
He, Ziyi	Xue, Mingxin	Pizzico, Tyler R	Rigas, Andrew
Thenappan, Bala Sundar	Nojoomi, Radin	Zhang, Miaoyan	Pace, Benjamin Michael
Cruz, Marye I	Kong, Rachel	Lee, Jaeyoung	Zhang, Minzheng
Schnall, Aaron Hewitt	Pinheiro, Benjamin B	Chen, Zheyi	Richmond, Christian
Graham, Alexander Richard	Wang, An-Jie	Qiu, Chengzhuo	Liu, Shufan
Gallagher, John Manus	Chou, Randy	Shah, Rushabh	Mammadov, Elmar
Li, Yunhe	Qiu, Xi	Sha, Yumeng	Liu, Jiayun
Kung, Ling-Hsin	Zhang, Yihong	Sabri, Rita	Wang, Yuanqi

> If members of your group are not present, that's OK. If you're alone, join another group. If you're working remotely, feel free to congregate and work together on gather.town. These are the same groups as last time, as I understand we couldn't get that to work before.

1. Correct the following program so that it compiles.

```
public class MultiplePizzaAreas {

    private int pizzasCreated;

    public void addNewPizza() {
        pizzasCreated++1;
    }

    public int getPizzasCreated() {
        return pizzasCreated
    }

    public void printPizzaArea(double pizzaDiameter) {
        int pizzaRadius;
        int pizzaArea;
        int piVal = 3.14159265;

        // calculate pizza area given a diameter
        pizzaRadius = pizzaDiameter / 2.0;
        pizzaArea = (piVal * pizzaRadius) * pizzaRadius);

        // display results
        System.out.print(pizzaDiameter + " inch pizza is ")
```

```
        System.out.println(pizzaArea + " inches squared.")
    }

    public static void main (String [] args) {
        MultiplePizzaAreas pizzas = new MultiplePizzaAreas();

        System.out.println(pizzas.getPizzasCreated + " created so far");
        pizzas.printPizzaArea(12.0);
        pizzas.printPizzaArea(16);
    }
}
```

2. Complete LAB 3.11 from the zybook. You can work in the textbook to get instant feedback, **although you'll need to submit your solution to Gradescope either way.** I've copied the problem below.

   Build the Student class with the following specifications:

   - Private fields
     - String name - Initialized in default constructor to "Louie"
     - double gpa - Initialized in default constructor to 1.0
   - Default constructor
   - Public member methods
     - setName() & getName()
     - setGPA() & getGPA()

```
public class Student {
    // TODO: Build Student class with private fields and methods listed above

    // TODO: Define two private member fields

    public Student() {
        // TODO: Define default constructor
    }

    public void setName(String n) {
        // TODO: Assign parameter to instance field
    }

        // TODO: Add three more methods


    public static void main(String[] args) {
        Student student = new Student();
        System.out.println(student.getName() + "/" + student.getGPA());

        student.setName("Felix");
        student.setGPA(3.7);
        System.out.println(student.getName() + "/" + student.getGPA());
    }
}
```