Name _____ Period _____ Logic and Computation

# *Programming Lab: Designing Classes with Conditionals*

**Directions**: The purpose of this lab is to practice conditional statements and the logical operators in Java: ! (NOT), && (AND), and || (OR). It is also about being able to read instructions carefully and make the classes behave exactly the way they are specified in the word problem. You should carefully test your classes to make sure that they are following the rules provided and trigger an error message in the right circumstances.

**Setup:** Create a new project called: **P<#><LastFirst>Lab1**

- But in place of <#>, write the number of the class period that you are in.
- And in place of <LastFirst> write your last name followed by your first name, capitalizing the beginning letter of each word.
- Save the project to your flash drive.

For example, if your name is Sherlock Holmes and you are in period 9, your project name would be **P9HolmesSherlockLab1**

## *1. Ecosystem*

Develop class definitions for a program that will help ecologists simulate an ecosystem consisting of foxes, rabbits, and grass. Each organism has a position consisting of an x and y coordinate. In addition to that, foxes have a hunger level and a status saying whether or not they are currently hunting. Rabbits have a hunger level, a status saying whether or not they are ready to breed, and another status saying whether or not they are hiding. Grass has a level of growth.

Create a constructor for all classes.

The class constructors should enforce the following constraints:

- A fox's hunger level is a whole number between 0 and 5 inclusive.
- A fox cannot be currently hunting if its hunger is 0.
- A fox must be hunting if its hunger is 5.
- A rabbit's hunger level is a whole number between 0 and 3 inclusive.
- A rabbit cannot be ready to breed if its hunger is 3.
- A rabbit cannot be hiding if its hunger is 3 or if it is ready to breed.
- The level of growth of grass is a decimal number between 0 and 3 inclusive.

A user entering data that breaks one of these constraints should receive an error triggered by the syntax:

```
throw new IllegalArgumentException("Error Message");
```

and a sensible error message should be provided with information on what was entered incorrectly.

## *2. Train Schedules*

Develop class definitions for a program that can assist railway travelers with the arrangement of train trips. The available information about a specific train includes its schedule, its route, and whether it is an express train. The route information consists of the names of the origin and the destination station. A schedule specifies the date, the departure time, and the arrival time. There are only five stations: Happy Town, Smilesberg, Pleasanton, Sunnyville, and Frowningham.

Create constructors for all classes.

The class constructors should enforce the following constraints:

- A route cannot have the same station name for the origin and destination.
- The month of the date must be a number between 1 and 12 inclusive.
- The day of the date must be a number between 1 and 31 inclusive.*
- The year of the date cannot be zero (no such year exists).
- The hour of a time is a number between 0 and 23 (we're using the 24-hr clock convention)
- The minute of a time is a whole number between 0 and 59. (Seconds are irrelevant for train schedules.)

A user entering data that breaks one of these constraints should receive an error triggered by the syntax:

```
throw new IllegalArgumentException("Error Message");
```

and a sensible error message should be provided with information on what was entered incorrectly.

* Don't worry about making the months have the correct number of days. Assume all months can have 31 days.

---

## *3. Food Delivery*

The startup company "DineNDash" needs your help to make yet another food delivery app. Develop the class definitions needed to fulfil the program's data requirements.

The app requires users to create a user profile that includes their name, their credit card number, and their address. An address has a street name such as "12345 N. Fake St.", it has a city, a state, and a zip code. The app also requires restaurants to create a profile that includes the restaurant name, address, and times that they open and close. The delivery driver's profile also has a name and an employee id

number. An order receipt on the app contains information about the customer, restaurant, and delivery driver. It also records the date, the total price of the order, and the tip.

Create constructors for all classes.

The class constructors should enforce the following constraints:

- Zip codes must be positive five-digit numbers.
- The delivery drivers do not leave the city to make deliveries and they certainly don't leave their state. An order is invalid if the city or state of the customer is different than the city or state of the restaurant.
- The hour of a time is a number between 0 and 23 (we're using the 24-hr clock convention)
- The minute of a time is a whole number between 0 and 59 (seconds are irrelevant).
- The month of the date must be a number between 1 and 12 inclusive.
- The day of the date must be a number between 1 and 31 inclusive.*
- The year of the date cannot be zero (no such year exists).
- The total price and the tip cannot be negative numbers.

## *Bonus:*

1. Add a snake to the ecosystem. A snake has a position, a hunger level, a hiding status, and a hunting status.

The snake class should enforce the following constraints:

- A snake's hunger level is a whole number between 0 and 3 inclusive.
- If a snake's hunger is 0, it must be hiding.
- If a snake's hunger is 3, it must be hunting.
- A snake cannot hide and hunt at the same time if its hunger is greater than 1.

A user entering data that breaks one of these constraints should receive an appropriate error message.

2. Ensure that the arrival time of a train is *after* the departure time. If not, raise an appropriate error message. Perform a similar check that the restaurant's closing time is after the opening time.

3. Add a new piece of data to the train schedule: a time keeping track of the duration. The duration is the length of the train trip in hours and minutes and is automatically calculated based on the arrival and departure times.

4. Ensure that the months have the correct number of days:

*Thirty days hath September,*
*April, June, and November,*
*All the rest have thirty-one,*
*But February's twenty-eight.*

5. Create class definitions for a geometry program. The program will plot points, segments, and triangles. Points have an x and y coordinate, segments have two endpoints, and triangles have three points and three segments.

The constructor for points takes two doubles (x and y). The constructor for segments takes two points (A and B). The constructor for triangles takes three points (A B and C) and constructs its own segments (AB BC CA) from those points.

Write overloaded constructors for segment and triangle. The overloaded constructor for segment takes four doubles (x1 y1 x2 y2) and constructs its own endpoints from those numbers. The overloaded constructor for triangles takes six doubles (x1 y1 x2 y2 x3 y3) and constructs its own points and segments from those numbers.

The endpoints of a segment cannot be identical. The three vertices (points) of a triangle cannot lie on the same straight line. Throw exceptions to enforce these constraints.