**Part 0: Preamble**

**Filename**: (none)

**Constructors**

So before you start coding the other methods, you should have a constructor, which is basically a preparation to create an object. Usually, if you don’t have a constructor, the compiler automatically makes a default constructor. However, if you’re making your own constructor then you are allowed to have specific parameters. Check out this code so that you can understand better:

Dog.java

01 public class Dog() {

02 private String name;

03 private int age;

04 private String gender;

05

06 public Dog(String name, int age, String gender) {

07 this.name = name; //this.name refers to 02, while name refers to 06

08 this.age = age;

09 this.gender = gender;

10 }

11 }

Note that the constructor has the same method name as the class. It always have to be that name. You can make multiple constructors using the SAME name, but they can’t have the same parameters. Try making two constructors, a default and a non-default, for class Fruit using the following instance variables: private boolean seeds, private String type, private double size, private int calories. For the default, assign seeds as false, type as “apple”, size as 10.5, and calories as 80.

**Accessor Methods**

As you know, accessors are methods that return a value from a private field. For example, if there is a class called “Bird” and there is a String instance variable *color*. To return the circle color, you would use an accessor like the following:

public String getColor() {

return color;

}

Therefore, if you create a Bird object *robin*, then by using robin.getColor(), you can access the name in another class. Now, try to create accessor methods for the following instance variables:

private int height

private boolean wings

private double weight

private String[] listOfPatterns

**Mutator Methods**

So, after you finish the accessor methods, then you’re ready for the next step: the mutator. Mutators don’t return anything but instead they change the instance variables. Therefore, they should be void. Here is an example:

public void changeColor(String c) {

color = c;

}

Now, the String color should be changed to String c. In another class, you can use robin.changeColor(“red”) so that color = “red”. Code the following mutators using the variables from the previous:

public void changeWeight(double a)

public void wings(boolean b)

public void grow() ← this method should add one to height

public void addPattern(String pat) ← try adding another String onto listOfPatterns without using ArrayList

When you’re done, you are ready to do the assignment!

**Part 1:**

**Filename**: PartyGoer.java

In this lab, you will be practicing how object-oriented programming works by using two classes: Party and PartyGoer. The Party class will keep track of admission and determine whether a PartyGoer is admitted into the party or not. The Party also has a method that can kick out people or check to see if the person is already admitted. There is some mandatory extra practice not related to the code so that it can help you write the actual assignment.

All comments refer to a method or variable below.

**private int** *age*

   /\* You have to be on the list \*/

**private String** *name*

   /\* Once you are admitted, you must bring a type of food. \*/

**private String** *food*

   /\* There is a certain dress code you must follow at the party. \*/

**private boolean** *dressCode*

   /\* Can only be admitted if you have a ticket \*/

**private boolean** *ticket*

**public** PartyGoer (**String** name, **String** food, **int** age**, boolean** dressCode, **boolean** ticket)

**public** **String** getName()

**public** **int** getAge()

**public** **String** getFood()

**public** **boolean** getDressCode()

**public** **boolean** getTicket()

/\* Checks if the food given is different from the original food and will print out that the member is bringing a different food. Otherwise it will print out that they are bringing the same food. \*/

**public** **void** changeFood(**String** food)

   /\* Checks if the dress code is different and tells the user. \*/

**public** **void** changeDressCode(**boolean** dressCode)

**Part 2:**

**Filename**: Party.java

All comments refer to a method or variable below.

   /\* Default age range is between 13 and 18, inclusive \*/

**private int** *maxAge*

**private int** *minAge*

   /\* Default maximum amount of people is 10 \*/

**private int** *maxPeople*

   /\* The Default list of things they can bring include chicken, cake, salad, pie, punch, chips, wings, pretzels, salsa, pasta \*/

**private String[]** *allowableFoods*

**private String[]** *listOfAdmitted*

   // Basic Default constructor sets numbers

**public** Party()

   // Basic constructor assigns variables

**public** Party(int maxAge, int minAge, int maxPeople, String[] foodList)

   /\* These are accessors and return what is specified \*/

**public** **int** getMaxAge()

**public** **int** getMinAge()

**public** **int** getMaxPeople()

**public** **String[]** getFoodList()

**public** **String[]** getlistOfAdmitted()

**public** **boolean** isAdmitted(PartyGoer person)

   /\* Checks if they have food (if food is in the foodList), if they are admitted, if they are older than the age limit, if they have the ticket, makes sure they are not already on the people list, and if they follow the dress code. Adds the person to the people list \*/

**public** **void** admit(**PartyGoer** person)

   /\* Checks if the person is in the list. If so, it will make that spot null \*/

**public** **void** kickOut(**PartyGoer** person)

/\* Checks to see if the person is admitted already. \*/

**public** **boolean** isAdmitted(**PartyGoer** person)