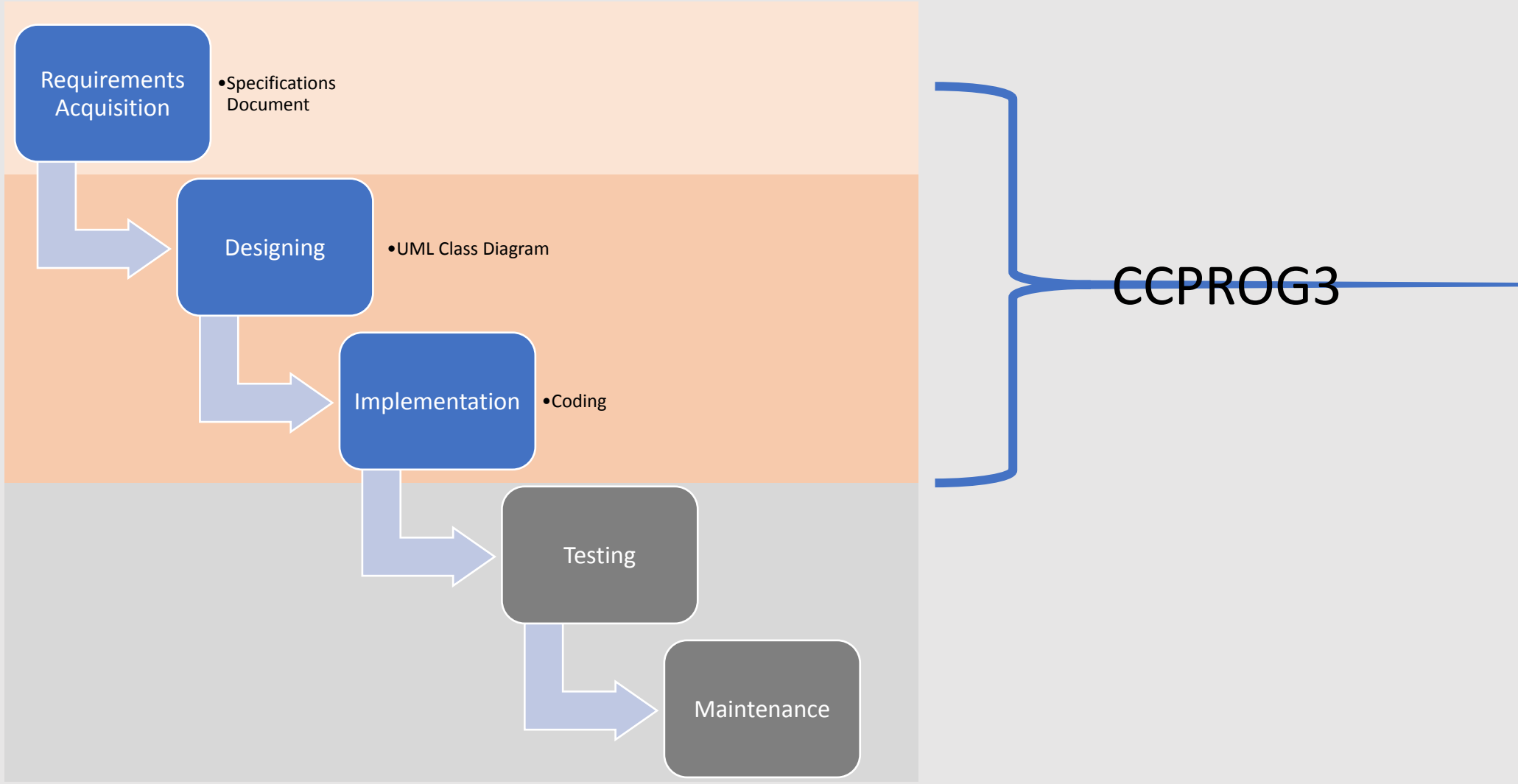
Object-Oriented Programming

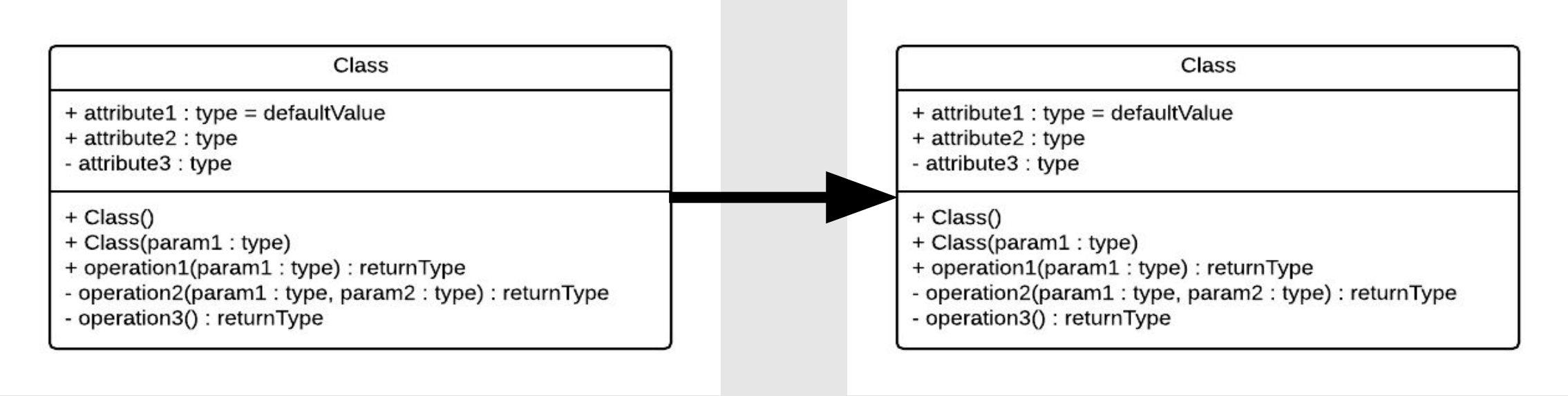
Class Diagramming and Specifications / Requirements Mindset

Software Development Life Cycle



Recall: UML Class Diagram

- Classes and Relationships



Recall: Model and View Classes

- **View**
  - Input / Output Class
  - Interfaces with the user

- **Model**
  - Handles the “remembering” of data of an instance
  - “Answers” something about the data of an instance

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

View Class

Model Class

“Single Responsibility Principle”

```java
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String petName;
        int petYearAdopted;
        char petSex;
        Pet pet1, pet2;

        System.out.print("Pet 1 name: ");
        petName = scanner.next();
        System.out.print("Pet 1 year adopted: ");
        petYearAdopted = scanner.nextInt();
        System.out.print("Pet 1 sex(m/f): ");
        petSex = scanner.next().charAt(0);

        pet1 = new Pet(petYearAdopted, petName, petSex);

        System.out.print("Pet 2 name: ");
        petName = scanner.next();
        System.out.print("Pet 2 year adopted: ");
        petYearAdopted = scanner.nextInt();
        System.out.print("Pet 2 sex(m/f): ");
        petSex = scanner.next().charAt(0);

        pet2 = new Pet(petYearAdopted, petName, petSex);

        scanner.close();

        Main main = new Main();
        main.generatePetCard(pet1, 1);
        main.generatePetCard(pet2, 2);

    }

    public void printRule(int count){
        for(int i = 0; i < count; i++) {
            System.out.print("=");
        }
        System.out.print("\n");

    }

    public void generatePetCard(Pet pet, int count) {
        String cuteName = "Cute Name: " + pet.getCuteName();
        printRule(cuteName.length() + 2);
        System.out.println("Pet " + count + " (" + pet.getName() + ")");
```

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

Model Class

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

Model Class

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
  - Are we accommodating changes to the data in the future? YES. (Scenario: Clerical error)
- How old is the pet?
- How do I call your pet's attention?

Model Class

If you're not accommodating changes to the data, then setSex() method should not be there. Remember to only expose attributes only when needed.

This is where **understanding the requirements** comes to play.

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
  - Are we accommodating changes to this attribute? YES (Scenario: Clerical error)
- How do I call your pet's attention?

Model Class

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
  - Are we accommodating changes to this attribute? YES (Scenario: Clerical error)
- How do I call your pet's attention?

Model Class

getYearAdopted() is optional because it already have the getAdoptionAge() method. On the other hand, if we just have getYearAdopted() method for this question, then **we're leaving the actual computation of adoption age to the user (not a good design)**.

This is where **understanding the requirements** comes to play.

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

Model Class

Sometimes, there are multiple ways to handle 1 requirement. This usually arises when there are multiple types of users of your program.

This is where **understanding the requirements** comes to play.

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

Model Class

getCuteName() is dependent on the name and sex attributes of Pet. We can directly access them from within the class or we can also use the getter methods of those class.

Direct Access – Assumes no retrieval rules

Getter method – allows room for implementation of retrieval rules

This is where **understanding the requirements** comes to play.

```java
public class Pet {

    private int yearAdopted;
    private String name;
    private char sex;

    public Pet(int yearAdopted, String name, char sex){
        this.yearAdopted = yearAdopted;
        this.name = name;
        this.sex = sex;
    }

    public void setSex(char sex) {
        this.sex = sex;
    }

    public char getSex() {
        return sex;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setYearAdopted(int yearAdopted) {
        this.yearAdopted = yearAdopted;
    }

    public String getName() {
        return name;
    }

    public int getYearAdopted() {
        return yearAdopted;
    }

    public int getAdoptionAge(){
        return 2023 - this.yearAdopted;
    }

    public String getCuteName(){
        String cuteName = "";

        if(this.sex == 'm'){
            cuteName += "Mr. ";
        } else {
            cuteName += "Ms. ";
        }
```

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

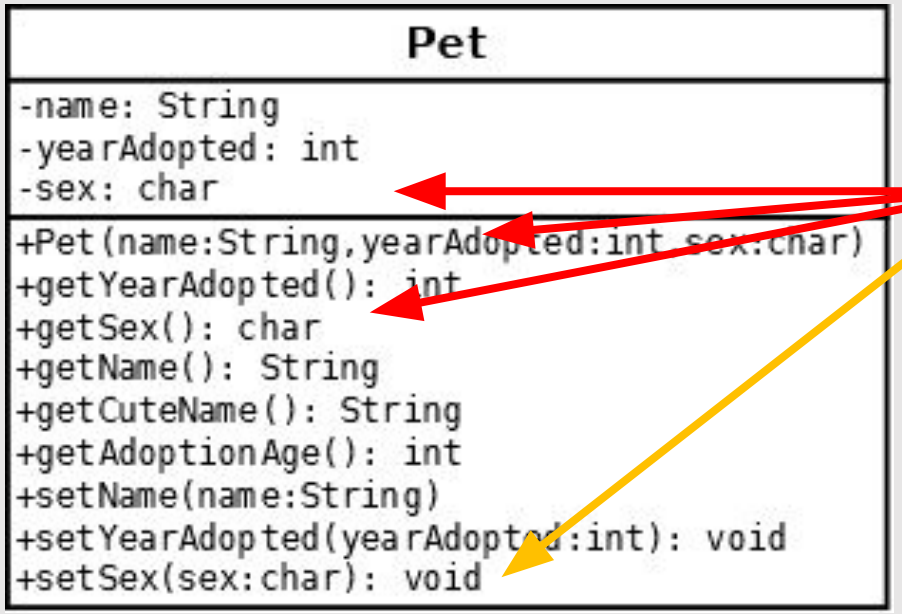
Model Class

getCuteName() is dependent on the name and sex attributes of Pet. We can directly access them from within the class or we can also use the getter methods of those class.

Direct Access – Assumes no retrieval rules

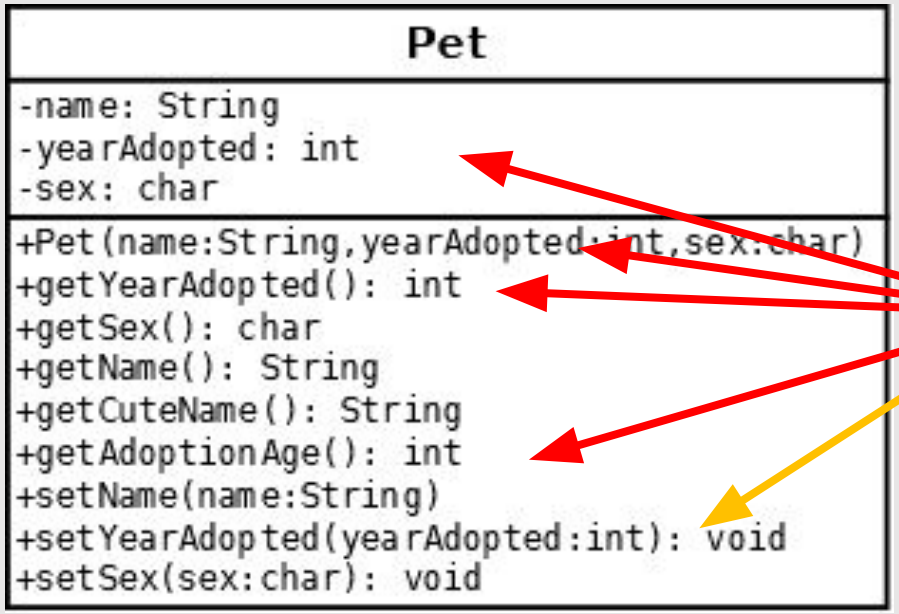
Getter method – allows room for implementation of retrieval rules

This is where **understanding the requirements** comes to play.



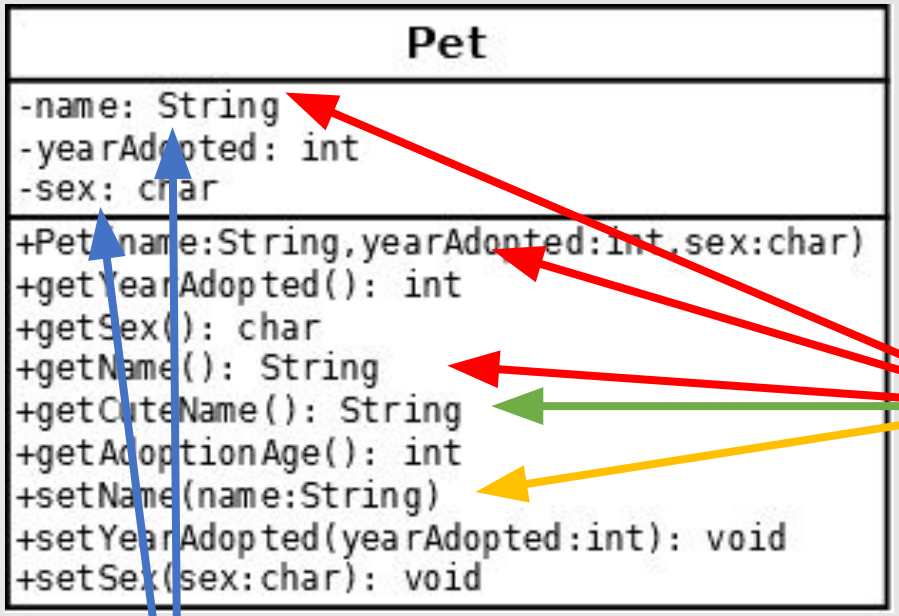
What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?



What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

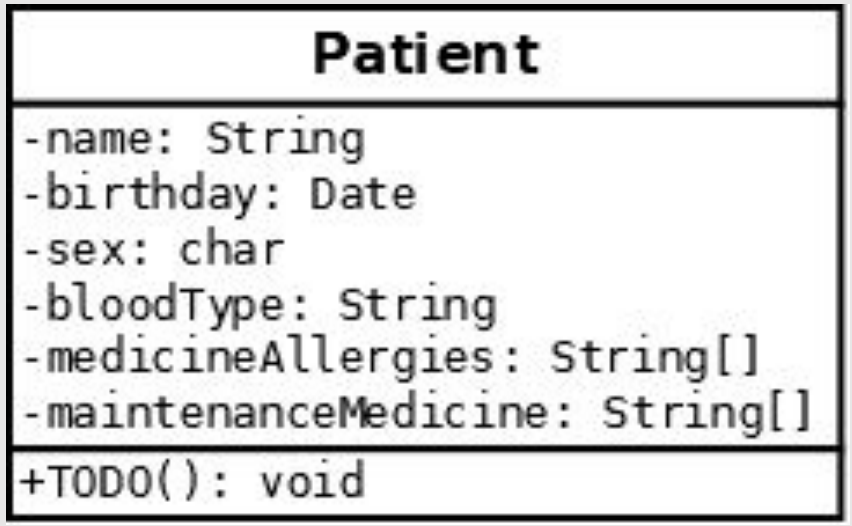


Recall that we need the name and sex of the Pet to generate its cute name.

What questions can you ask the Model?

- Is the pet male or female?
- How old is the pet?
- How do I call your pet's attention?

Let’s Examine Another Model Class



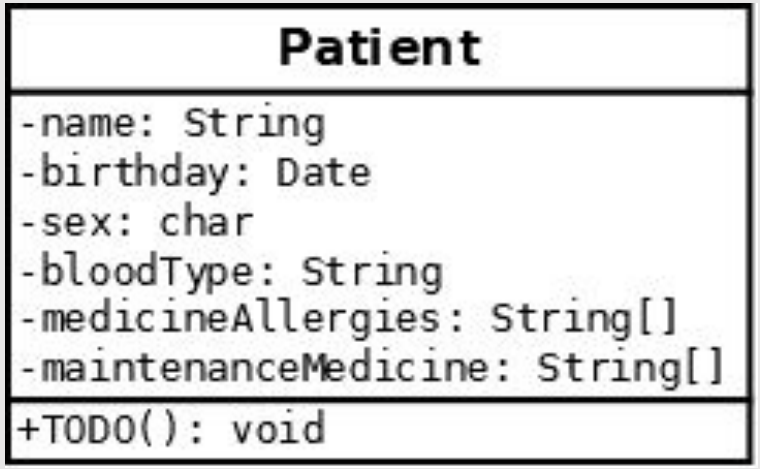
- What questions can we ask from its attributes?
- Based on the questions asked, what methods do we need create?

Attributes of Classes and Requirements

- Observe that we cannot ask questions about a patient's medical history. (Why?)
  - There are no attributes in the Patient (Model) class pertaining to a patient's medical history (e.g., Family of diabetics, hereditary heart disease, genetically predisposed to certain types of cancers).
- Can you think of other questions that we cannot ask about a patient with respect to the class?
  - Hospital Dietitian related questions
    - What food should I prepare for the patient?

    (no weight, no food allergy, no religion fields)
  - Billing related questions
    - How much should I charge the patient?

    (No type of room, no treatments done, no doctor)

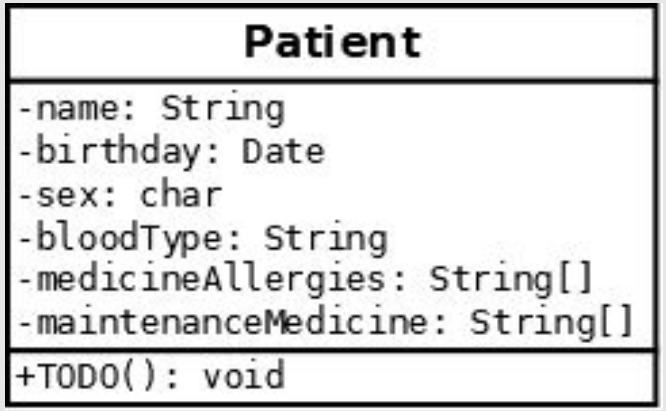


Methods of Classes and Requirements

- “is the patient still operable with respect to his age?”
  - Age of a patient pays a big role in making medical decisions.
  - We can answer this question because we have the birthday attribute.
  - What’s left is to plan the methods needed to answer this question
    - + getAge():int
  - *Notice that getBirthday() is not pertinent for this question*

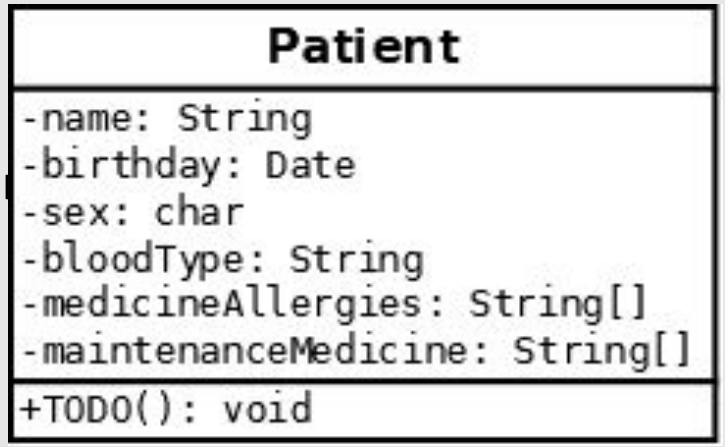
- “which blood type should I prepare for the procedure of this patient?”
  - +getBloodType(): String

- “can I give blood thinners to the patient?”
  - + getMedicineAllergies(): String[]
  - + getMaintenanceMedicined(): String[]



Features of a Program

- **Questions needed to be answered by a program are translated as features**
- “is the patient still operable with respect to his age?”
  - Retrieve Patient Age (getAge())
  - Input Birthday of Patient (Patient Constructor or setBirthday())
- “I’d like to get to know my patient before I do my rounds. Can I read his ad info?”
  - Input Patient Info (Patient Constructor or setter methods)
  - Retrieve Patient Info (getter methods)
- “which blood type should I prepare for the procedure of this patient?”
  - Retrieve patient blood type (getter methods)
  - Input patient blood type (setBloodType() or Patient Constructor)
- “The encoder made an error in the patient’s name. Can the encoder change the patient’s name?”
  - Change patient’s name (setName())



Summary

- Questions lead to features
- Attributes will determine if these questions are answerable
- Methods aid in answering the questions using a Model's attributes

- *Design should be specs-driven. There are certain design decisions that you must make, and hope that it will withstand future requirements to the program. But don't over engineer your program as well; otherwise, there's no end to it.*

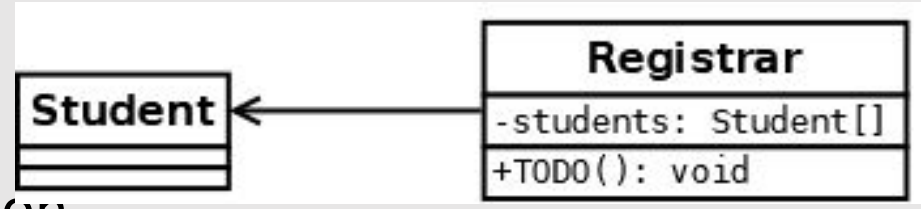
Questions? ☺️

Practice Exercise

- Upon admission, D-University students are required to fill out a bio information sheet. In this sheet, their name, date of birth, age, preferred username, high school, Degree Code, student ID number and year of admission are captured. By default, the status of a student is "admitted". Your task is to design a system that can accommodate the following features:
  - Determine the classification of a student [Freshman, Sophomore, Junior, Senior or Terminal (year of admission >= 5)]
  - Change Student Status
    - Expelled student
      - flips the status of student to "expelled"
      - Ask for year of expulsion
    - Graduate student
      - flips the status of student to "graduated"
      - Ask for year of graduation
  - Generate Info Sheet of a Student (Show full student info)
  - Generate Diploma Fields
    - Display the following: Name, Year Admitted, Degree Code, Year Graduated
    - Your system should print an error message if the student has not graduated
- *Convert your class diagram to Code and create a Main Class (that acts as your View Class) triggering the defined features.*

Interaction of Classes: Expanding from Practice Exercise

- Assuming that your Student class was implemented properly, let's expand your program.
- Create a Registrar class that has an array of student objects as its attribute. You may assume that the system can only accommodate 10 student profiles. **Observe that Registrar Class is still a model class.**
- Implement the following features:
  - Create a new student profile
  - Display the number of students graduated
  - Display the number of students per classification
  - Search* for a student based on the substring name (case insensitive) and display student info (e.g., name: "Joseph" vs. search keyword: "Jose")
  - Search* for a student based on their ID number (full match) and display student information



** For search functions, system should display "no match found" if there are no matches in the search. If there are multiple matches, display only first result.*

Keep learning…