Northeastern University

# Lecture 6: Java Review - 6 (Self-Study)

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from

Data Structures and Abstractions with Java, 5th edition. By Frank M. Carrano and Timothy M. Henry.
ISBN-13 978-0-13-483169-5 © 2019 Pearson Education, Inc.

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133766268 © 2014 Pearson Education, Inc.

# What have we reviewed?

- Encapsulation & Abstraction
- Methods
- Interfaces
- System & Classes Design (UML)
- Classes, Methods, Objects, Inheritance (Labs)
- Classes and Methods

# Outline

- Objects and Methods
  - » Constructors
  - » Overloading

# Objects and Methods

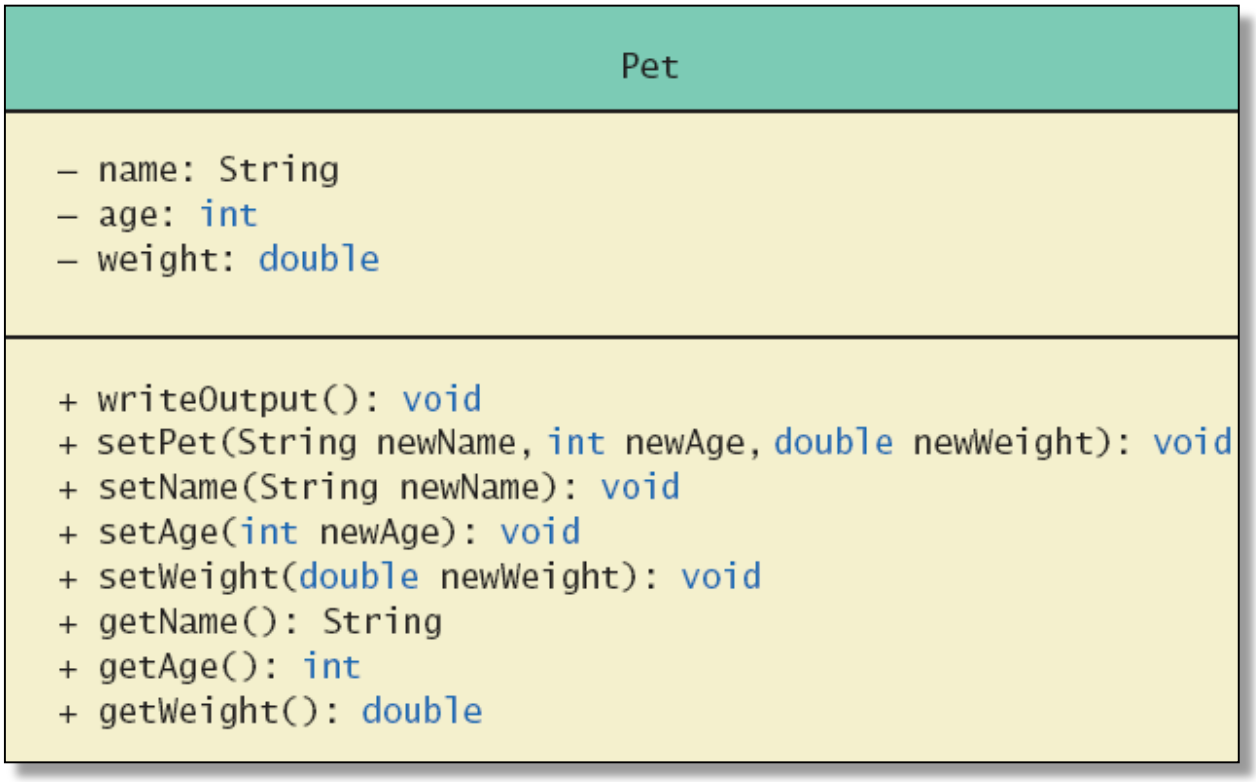
# Constructors

# Defining Constructors

- A special method called when instance of an object created with new
  - » Create objects
  - » Initialize values of instance variables
- Can have parameters
  - » To specify initial values if desired
- May have multiple definitions
  - » Each with different numbers or types of parameters

# Defining Constructors

- Example class to represent pet




Class Diagram for a Class **Pet**

# Defining Constructors

- Note sample code: `class Pet`
- Note different constructors
  - » Default
  - » With 3 parameters
  - » With 1 int parameter
  - » With 1 String parameter
  - » With 1 double parameter
- Note sample program: `class PetDemo`

# Defining Constructors

```
My records on your pet are inaccurate.
Here is what they currently say:
Name: Jane Doe
Age: 0
Weight: 0.0 pounds
Please enter the correct pet name:
Moon Child
Please enter the correct pet age:
5
Please enter the correct pet weight:
24.5
My updated records now say:
Name: Moon Child
Age: 5
Weight: 24.5 pounds
```

Sample screen output

## LISTING 6.1 The Class Pet: An Example of Constructors and Set Methods *(part 1 of 3)*

```
/**
 Class for basic pet data: name, age, and weight.
*/
public class Pet
{
    private String name;
    private int age;       //in years
    private double weight;//in pounds

    public Pet()  <-- Default constructor
    {
        name = "No name yet.";
        age = 0;
        weight = 0;
    }
```

```
public Pet(String initialName, int initialAge,
           double initialWeight)
{
    name = initialName;
    if ((initialAge < 0) || (initialWeight < 0))
    {
        System.out.println("Error: Negative age or weight.");
        System.exit(0);
    }
    else
    {
        age = initialAge;
        weight = initialWeight;
    }
}
public void setPet(String newName, int newAge,
                   double newWeight)
{
    name = newName;
    if ((newAge < 0) || (newWeight < 0))
    {
        System.out.println("Error: Negative age or weight.");
        System.exit(0);
    }
    else
    {
        age = newAge;
        weight = newWeight;
    }
}
```

```java
public Pet(String initialName)
{
    name = initialName;
    age = 0;
    weight = 0;
}

public void setName(String newName)
{
    name = newName; //age and weight are unchanged.
}
```

```java
public Pet(int initialAge)
{
    name = "No name yet.";
    weight = 0;
    if (initialAge < 0)
    {
        System.out.println("Error: Negative age.");
        System.exit(0);
    }
    else
        age = initialAge;
}

public void setAge(int newAge)
{
    if (newAge < 0)
    {
        System.out.println("Error: Negative age.");
        System.exit(0);
    }
    else
        age = newAge;
    //name and weight are unchanged.
}
```

```
public Pet(double initialWeight)
{
    name = "No name yet";
    age = 0;
    if (initialWeight < 0)
    {
        System.out.println("Error: Negative weight.");
        System.exit(0);
    }
    else
        weight = initialWeight;
}
public void setWeight(double newWeight)
{
    if (newWeight < 0)
    {
        System.out.println("Error: Negative weight.");
        System.exit(0);
    }
    else
        weight = newWeight; //name and age are unchanged.
}
```

```
    public String getName()
    {
        return name;
    }

    public int getAge()
    {
        return age;
    }

    public double getWeight()
    {
        return weight;
    }

    public void writeOutput()
    {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age + " years");
        System.out.println("Weight: " + weight + " pounds");
    }
}
```

## LISTING 6.2 Using a Constructor and Set Methods

```
import java.util.Scanner;
public class PetDemo
{
    public static void main(String[] args)
    {
        Pet yourPet = new Pet("Jane Doe");
        System.out.println("My records on your pet are inaccurate.");
        System.out.println("Here is what they currently say:");
        yourPet.writeOutput();

        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please enter the correct pet name:");
        String correctName = keyboard.nextLine();
        yourPet.setName(correctName);

        System.out.println("Please enter the correct pet age:");
        int correctAge = keyboard.nextInt();
        yourPet.setAge(correctAge);

        System.out.println("Please enter the correct pet weight:");
        double correctWeight = keyboard.nextDouble();
        yourPet.setWeight(correctWeight);

        System.out.println("My updated records now say:");
        yourPet.writeOutput();
    }
}
```

### *Sample Screen Output*

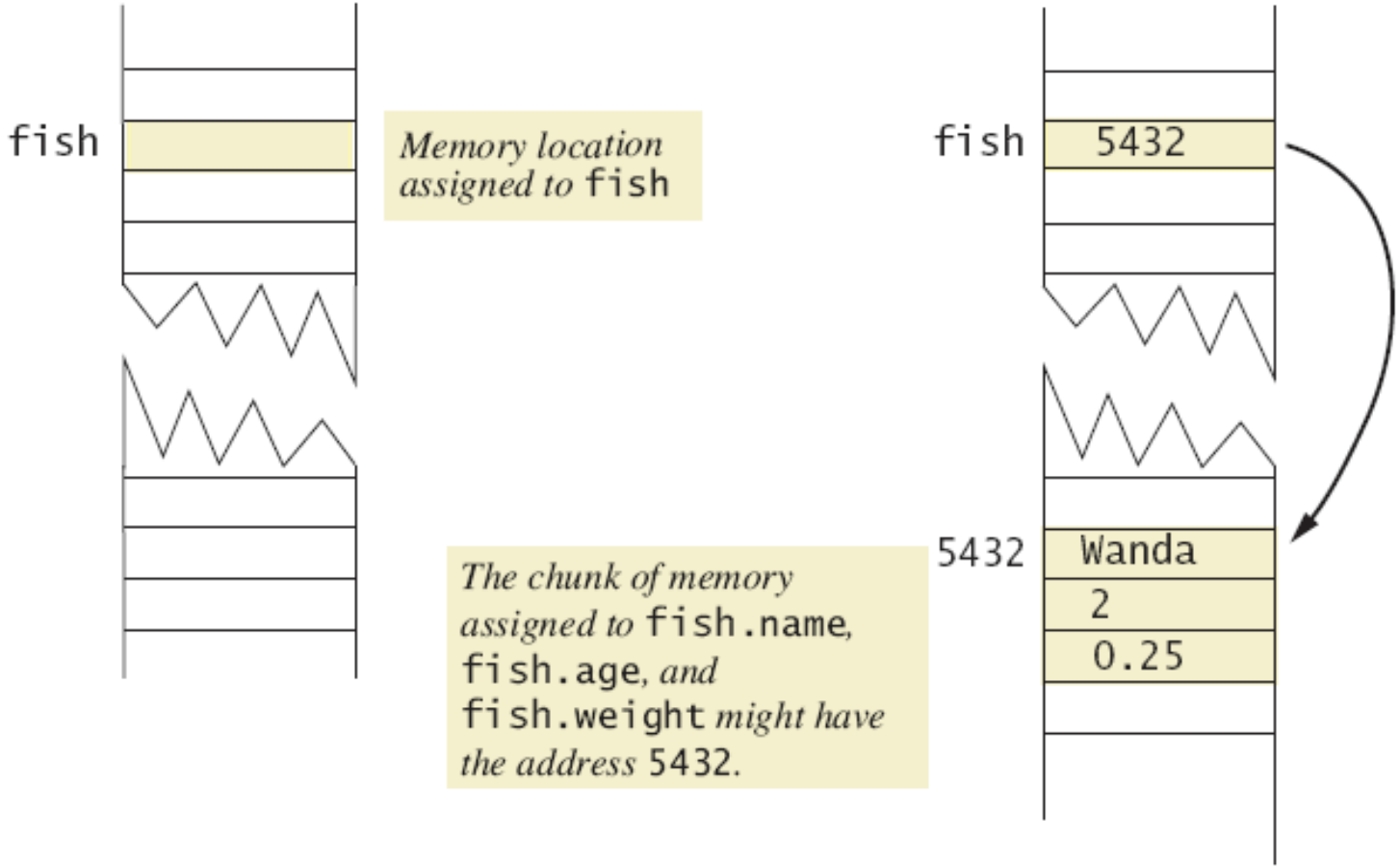
```
My records on your pet are inaccurate.
Here is what they currently say:
Name: Jane Doe
Age: 0
Weight: 0.0 pounds
Please enter the correct pet name:
Moon Child
Please enter the correct pet age:
5
Please enter the correct pet weight:
24.5
My updated records now say:
Name: Moon Child
Age: 5
Weight: 24.5 pounds
```

# Defining Constructors

- Constructor without parameters is the default constructor
  - » Java will define this automatically if the class designer does not define any constructors
  - » If you do define a constructor, Java will not automatically define a default constructor
- Usually default constructors are not included in class diagram

```
Pet fish;
```
*Assigns a memory location to* `fish`

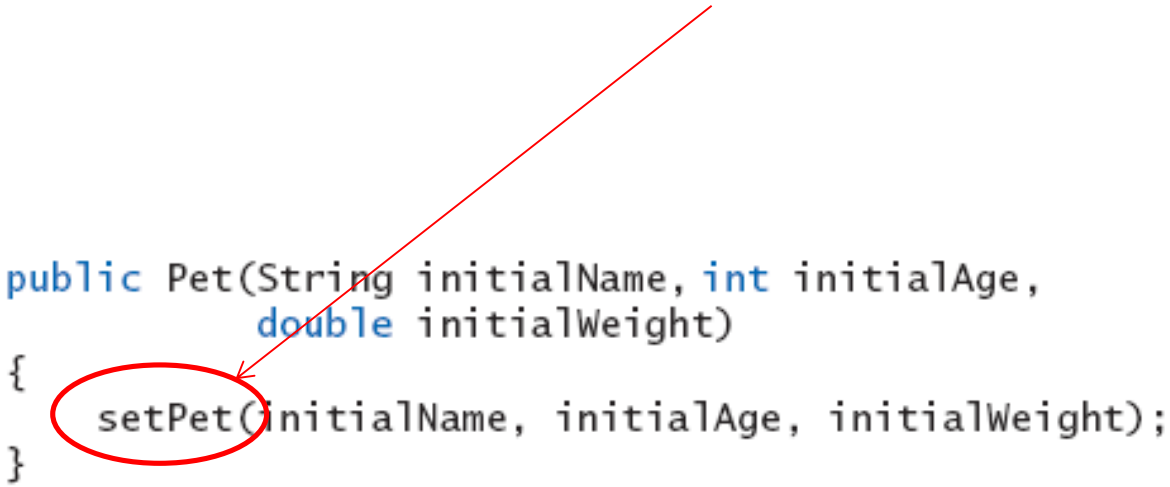
```
fish = new Pet();
```
*Assigns a chunk of memory for an object of the class* `Pet`*—that is, memory for a name, an age, and a weight—and places the address of this memory chunk in the memory location assigned to* `fish`




Constructor returns a reference

# Calling Methods from Other Constructors

- Constructor can call other class methods



# Exercise (10 minutes)

- Consider a class RatingScore that represents a numeric rating for something such as a movie.
  - » Attributes:
    - A description of what is being rated
    - The maximum possible rating
    - The rating
  - » Methods:
    - Get the rating from a user
    - Return the maximum rating possible
    - Return the rating
    - Return a string showing the rating in a format suitable for display
- Write a default constructor and a second constructor
- Implement and test the class

# Answer

```
import java.util.Scanner;
public class RatingScore {

    private String description;
    private int maximumRating;
    private int theRating;

    public RatingScore() {
        description = "this from 0 to 10";
        maximumRating = 10;
        theRating = -1;
    }

    public RatingScore(String desc, int max) {
        description = desc;
        maximumRating = max;
        theRating = -1;
    }

    public void inputRating(){
        System.out.println("What is your rating for " + description + "?");
        System.out.println("Please enter an integer from 0 to " + maximumRating);

        Scanner reader = new Scanner(System.in);
        int data = -1;
        boolean needTheRating = true;

        while(needTheRating){
            data = reader.nextInt();
            if(data>=0 && data<=maximumRating){
                needTheRating = false;
            } else {
                System.out.println("Sorry, that rating is out of range.");
                System.out.println("Please enter an integer from 0 to " + maximumRating);
            }
        }
        theRating = data;
    }
```

# Answer

```
    public int getMaxRating(){
        return maximumRating;
    }

    public int getRating(){
        return theRating;
    }

    public String getRatingString(){
        return "the rating is " + theRating + "/" + maximumRating;
    }

    public static void main(String[] args) {
        RatingScore movieRating = new RatingScore("Iron Man movie", 5);
        RatingScore restaurantRating = new RatingScore();

        movieRating.inputRating();
        System.out.println("Displaying the attributes for the movie rating.");
        System.out.println("The rating is " + movieRating.getRating() + " out of "
                + movieRating.getMaxRating());
        System.out.println();

        System.out.println("How was the quality of your food?");
        restaurantRating.inputRating();
        System.out.println("Displaying the rating for the restaurant.");
        System.out.println( restaurantRating.getRatingString());
    }
}
```

# Results

```
What is your rating for Iron Man movie?
Please enter an integer from 0 to 5
4
Displaying the attributes for the movie rating.
The rating is 4 out of 5

How was the quality of your food?
What is your rating for this from 0 to 10?
Please enter an integer from 0 to 10
3
Displaying the rating for the restaurant.
the rating is 3/10
```

# Overloading

# Overloading Basics

- When two or more methods have same name within the same class
- Java distinguishes the methods by number and types of parameters
  - » If it cannot match a call with a definition, it attempts to do type conversions
- A method's name and number and type of parameters is called the *signature*

## LISTING 6.15 Overloading

```
/**
 This class illustrates overloading.
*/
public class Overload
{
    public static void main(String[] args)
    {
        double average1 = Overload.getAverage(40.0, 50.0);
        double average2 = Overload.getAverage(1.0, 2.0, 3.0);
        char average3 = Overload.getAverage('a', 'c');
        System.out.println("average1 = " + average1);
        System.out.println("average2 = " + average2);
        System.out.println("average3 = " + average3);
    }

    public static double getAverage(double first, double second)
    {
        return (first + second) / 2.0;
    }

    public static double getAverage(double first, double second,
                                    double third)
    {
        return (first + second + third) / 3.0;
    }

    public static char getAverage(char first, char second)
    {
        return (char)(((int)first + (int)second) / 2);
    }
}
```

### *Sample Screen Output*

```
average1 = 45.0
average2 = 2.0
average3 = b
```