Chapter 6

# More About Objects and Methods

- 6.1 Constructors.
- 6.2 Static Variables and Static Methods
- 6.3 Writing  Methods
- 6.4 Overloading
- 6.5 Information Hiding Revisited
- 6.6 Enumeration as a Class
- 6.7 Packages
- 6.8 Graphics Supplements

# 6.1 Constructors

- A ***constructor*** is a special method
  designed to initialize [redacted]
- Automatically called when an object is [redacted] using `new`
- Has the [redacted] name as the class
- Often [redacted] (more than one constructor for the same class definition)
  - » [redacted] versions to initialize all, some, or none of the instance variables
  - » each constructor has [redacted] (a different number or sequence of [redacted] types)

# Defining Constructors

- Constructor headings
  - » do **not include the word** [redacted]
  - » do **not include** [redacted]
- ***a default constructor***
  - » A constructor with no [redacted]
- If no constructor is provided, Java automatically creates a default constructor.
  - » If *any* constructor is provided, **then *no* constructors are created automatically.**

<u>Programming Tip</u>

- Include a constructor that initializes *all* instance variables.
- Include a constructor that has [redacted] parameters.
  - » Include **your own *default constructor.***

# Constructor Example from **PetRecord**

```
public class PetRecord
{
   private String name;
   private int age; //in years
   private double weight; //in pounds
. . .
   public PetRecord(String initialName)
   {
      name = initialName;
      age = 0;
      weight = 0;
   }
```

Initializes three instance variables: `name` from the parameter and `age` and `weight` with default initial values.

Sample use:

```
PetRecord pet1 = new PetRecord("Eric");
```

# Fig 6.1 Class Diagram for a Class PET

```
Pet

- name: String
- age: int
- weight: double

+ writeOutput(): void
+ setPet(String newName, int newAge, double newWeight): void
+ setName(String newName): void
+ setAge(int newAge): void
+ setWeight(double newWeight): void
+ getName(): String
+ getAge(): int
+ getWeight(): double
```

**// Listing 6.1**

```
/**
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 ()   //[redacted] 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;
    }
}

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

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


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

```
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 ();
    }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
My records on your pet are inaccurate.
Here is what they currently say:
Name: Jane Doe
Age: 0 years
Weight: 0.0 pounds
Please enter the correct pet name:
Hosik
Please enter the correct pet age:
12
Please enter the correct pet weight:
20
My updated records now say:
Name: Hosik
Age: 12 years
Weight: 20.0 pounds
계속하려면 아무 키나 누르십시오 . . .
```

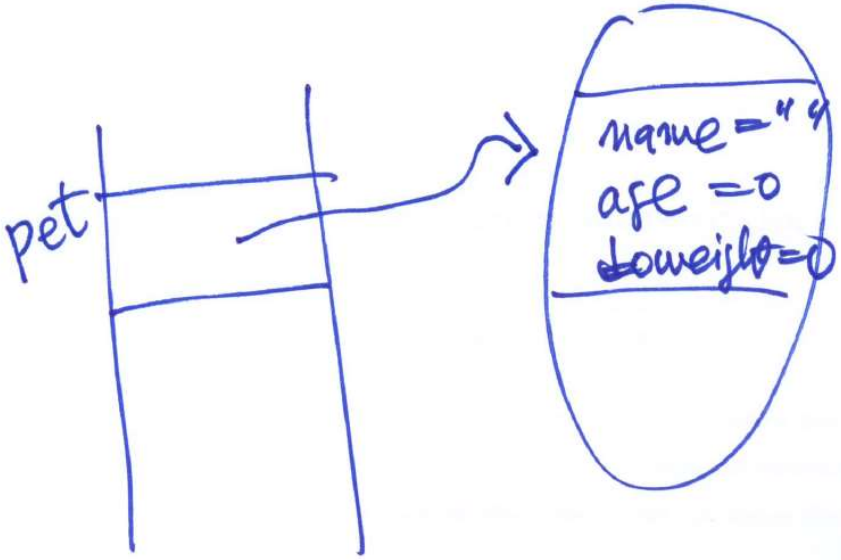
# Using Constructors

- Always use a constructor **after** [redacted]
- For example, using the `Pet` class in text:
  `Pet myCat = new Pet("Calvin", 5, 10.5);`
  - » this calls the `Pet` constructor with `String`, `int`, `double` parameters
- If you want to change values of instance variables after you have created an object, **you must use** [redacted] **for the object**
  - » you cannot call a constructor for an object after it is created
  - » [redacted] **methods** should be provided for this purpose

# Constructors Return a Reference

- Constructor invocation : returning a reference to an object, the memory address of an object.

- ex) pet = new PetRecord()



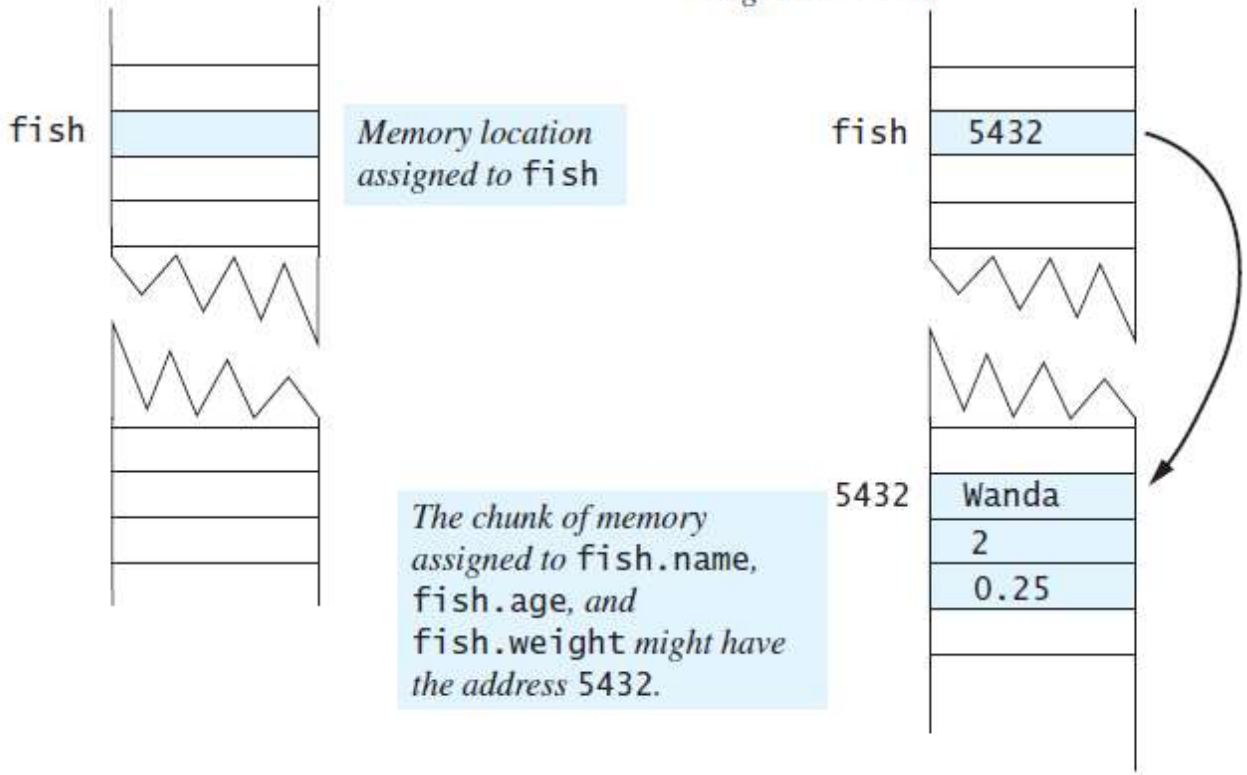
## FIGURE 6.2 A Constructor Returning a Reference

```
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`



# *Programming Tip*: You Can Use Other Methods in a Constructor

```
public PetRecord(String initialName,
            int initialAge, double initialWeight)
{
   name = initialName;
   if ((initialAge < 0) || (initialWeight < 0))
   {
      System.out.println("Error…");
   }
   else
   {
      age = initialAge;
      weight = initialWeight;
   }
}
```

- avoids possible confusion about **`set`** parameters
- less method invocation overhead

- shorter
- possibly more consistent with other constructors and methods

```
public PetRecord(String initialName,
            int initialAge, double initialWeight)
{
   set(initialName, initialAge, initialWeight);
}
```

Use the one your instructor or supervisor prefers.

# Constructors and Set Methods that Call a Private Method

```
// Listing 6.3
/**
Revised class for basic pet data: name, age, and weight.
*/
public class Pet2
{
    private String name;
    private int age; //in years
    private double weight; //in pounds
    public Pet2 (String initialName, int initialAge,
            double initialWeight)
    {
        set (initialName, initialAge, initialWeight);
    }
    public Pet2 (String initialName)
    {
        set (initialName, 0, 0);
    }
```

```
public Pet2 (int initialAge)
{
    set ("No name yet.", initialAge, 0);
}

public Pet2 (double initialWeight)
{
    set ("No name yet.", 0, initialWeight);
}

public Pet2 ()
{
    set ("No name yet.", 0, 0);
}

public void setPet (String newName, int newAge,
        double newWeight)
{
    set (newName, newAge, newWeight);
}
```

```
public void setName (String newName)
{
    set (newName, age, weight); //age and weight unchanged
}


public void setAge (int newAge)
{
    set (name, newAge, weight); //name and weight unchanged
}


public void setWeight (double newWeight)
{
    set (name, age, newWeight); //name and age unchanged
}
```

```
private void set (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;
    }
}

/*  The methods getName, getAge, getWeight, and writeOutput are
    the same as in Listing 6.1. >  */
}
```

# Calling Constructors from Other Constructors

```
// Listing 6.4
/**
Revised class for basic pet data: name, age, and weight.
*/
public class Pet3
{
    private String name;
    private int age; //in years
    private double weight; //in pounds
    public Pet3 (String initialName, int initialAge
            double initialWeight)
    {
        set (initialName, initialAge, initialWeight);
    }
    public Pet3 (String initialName)
    {
        (initialName, 0, 0);
    }
```

```
    public Pet3 (int initialAge)
    {
      [redacted] ("No name yet.", initialAge, 0);
    }

    public Pet3 (double initialWeight)
    {
      [redacted] ("No name yet.", 0, initialWeight);
    }

    public Pet3 ()
    {
      [redacted] ("No name yet.", 0, 0);
    }

    /*  The rest of the class is like Pet2 in Listing 6.3.  */
}
```

끝