THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Fall 2015

Time: TR 9:30 – 10:45

Room: AR 121 (Hanes Art Center)

Jay Aikat

aikat@cs.unc.edu

Previous Class

- What did we discuss?

Today

- Nested loops review in lab today: 5-8 PM
- Quiz on Thursday
- Assignment 4 (last assignment! ☺) will be announced soon; part A due next week
- Today – Exercises in Eclipse

Remember this example?

```
public class Student {
    public String name;
    public int classYear;
    public double GPA;
    public String major;

// ...

    public String getMajor() {
         return major;
    }

    public void increaseYear() {
         classYear++;
    }
}
```

```
public class StudentTest {
  public static void main(String[] args) {
         Student jack = new Student();
         jack.name = "Jack Smith";
         jack.major = "Computer Science";
         jack.classYear = 1;
         jack.GPA = 3.5;

         String m = jack.getMajor(); //
         System.out.println("Jack's major is " +
m);

         jack.increaseYear();

         System.out.println("Jack's class year is
now " + jack.classYear);

  }
}
```

Let's create our own example

- And now, let's practice all the constructor, getter and setters method details we discussed last two class sessions…
- Slides from last class follow – for reference

Summary: Constructor

- A special method with the same name as the class, and no return type
- Called only when an object is created
- It can take parameters to initialize instance variables
- You can define multiple constructors with different parameter lists

Default Constructor

- Constructor that takes no parameters

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

- Java automatically defines a default constructor if you do not define any constructors
  - You have not written a constructor explicitly, but you can still create objects

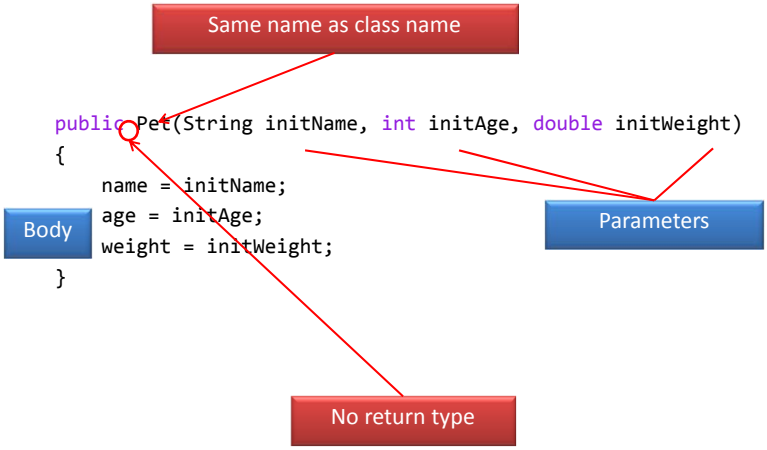
Constructor with Parameters

```
public class Pet
{
    private String name;
    private int age;
    private double weight;

    public Pet(String initName, int initAge, double initWeight)
    {
        name = initName;
        age = initAge;
        weight = initWeight;
    }

    public void setPet(String newName, int newAge, double newWeight)
    {
        name = newName;
        age = newAge;
        weight = newWeight;
    }
}
```

A Closer Look



Constructor with Parameters

- If you define at least one constructor, a default constructor will **not** be created for you
- Now you **must** create a Pet object like this:
  - `Pet odie = new Pet("Odie", 3, 8.5);`
  - `Pet odie = new Pet(); // WRONG! No default constructors!`

```
public class Pet {
    private String name;
    private int age;
    private double weight;
    public Pet(String initName, int initAge, double initWeight)
    {
        name = initName; age = initAge;   weight = initWeight;
    }
}
```

Multiple Constructors

- You can have several constructors per class
  - They all have the same name, just different parameters
    - Remember that the name is **the same as the class name**
  - The methods (with the same name) will be called according to its parameters

Multiple Constructors

```
public class Pet {
    private String name;
    private int age;
    private double weight;

    public Pet() {
        name = "No name yet.";
        age = 0;
        weight = 0;
    }

    public Pet(String initName, int initAge, double initWeight) {
        name = initName;
        age = initAge;
        weight = initWeight;
    }

     public static void main(String[] args)       {
              Pet p = new Pet();
              Pet q = new Pet("Garfield", 3, 10);
        }
}
```

Multiple Constructors

```
public class Pet {
    private String name = "No name yet.";
    private int age = 0;
    private double weight = 1; // The instance variables are initialized

    public Pet() {
        name = "No name yet.";
        age = 0;
        weight = 0;
    }

    public Pet(String initName, int initAge, double initWeight) {
        name = initName;
        age = initAge;
        weight = initWeight;
    }

    public Pet(String initName) {
        name = initName;
    }

    public static void main(String[] args) {
          Pet p = new Pet(); // p.weight is 0 – it is overwritten by constructor

          Pet q = new Pet("Garfield", 3, 10);

          Pet w = new Pet("Odie"); // w.weight is 1, as only one constructor
          //can be called. Variables will get initial value if not set in
          constructor.
    }
}
```


Calling a Constructor

- A constructor can be only called once when the object is created
  - `Pet odie = new Pet("Odie", 3, 8.5);`
- You can not invoke a constructor from an object
  - `odie.Pet("Odie", 3, 8.5);`
    `// Wrong! A constructor can not be invoked this way`
  - `odie.setPet("Odie", 3, 8.5);`
    `// Yes. You can use a setter instead`

Calling a Setter from the Constructor

```
public class Pet
{
    private String name;
    private int age;
    private double weight;

    public Pet(String initName, int initAge, double initWeight)
    {
        setPet(initName, initAge, initWeight);
    }

    public void setPet(String newName, int newAge, double newWeight)
    {
        name = newName;
        age = newAge;
        weight = newWeight;
    }
}
```


Initialization and Setting Instance Variables

- Initialization values give values to instance variables that are the same (or commonly the same) for all objects
- Constructors give values to instance variables that should be decided for each object
- Setters give values to instance variables that can be changed over time
  - If a value is never going to be changed, no setter is needed

Example: Initialize, Construct and Set

```
public class Pet {
    private String name;
    private int age = 0;
    // Age is always 0 (assuming newly-born pets are registered immediately)
    private double weight;

    public Pet(String initName, double initWeight){
        name = initName;
        weight = initWeight;
                // Name is given when registering, and can not be changed
    }

    public void setPetWeight(double newWeight) {
        weight = newWeight;
        // Weight changes every time you weigh your pet
    }

    public void setPetAge(double newAge) {
        age = newAge;
        // Surely age can change, too
    }
}
```


Next class

- Quiz on calling methods from other methods, constructors, setters and getters
- Sorting!