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)

Ben Newton for Jay Aikat

bn@cs.unc.edu

Previous Class

- What did we discuss?

Today

- Assignment3: DUE Thu, 11/5 @ 11:55 PM
- Hack 110 – Keep Away Tournament Server is now live.

- Today – Constructors and more

Calling a Method from a method

```
import java.util.*;
public class Exercise1 {
        public static void main(String[] args){
                int num1 = 5, num2 = 10, num3 = 15;
                int sum = addNumbers(num1, num2, num3);
                System.out.println("The result is " + sum);
        }
        static String getUserInput() {
                Scanner keybrd = new Scanner(System.in);
                System.out.println("Please input your full name");
                String user1 = keybrd.nextLine();
                return user1;
        }
        static int addNumbers(int n1, int n2, int n3){
                String s1 = getUserInput();
                System.out.println("Welcome to COMP110, " + s1);
                int result = n1+n2+n3;
                return result;
        }
}
```

public / private Modifier

- public void setMajor()
- public int classYear;

- public: object can directly access the method or instance variable outside the class (no restrictions)

public / private Modifier

- private void setMajor()
- private int classYear;

- private: object cannot directly call the method or us the instance variable outside of the class

public / private Modifier

```
public class Student
{
    public int classYear;
    private String major;
}
public class StudentTest{
   public static void main(String[] args){
        Student jack = new Student();
        jack.classYear = 1;
        jack.major = "Computer Science";
   }
}
```

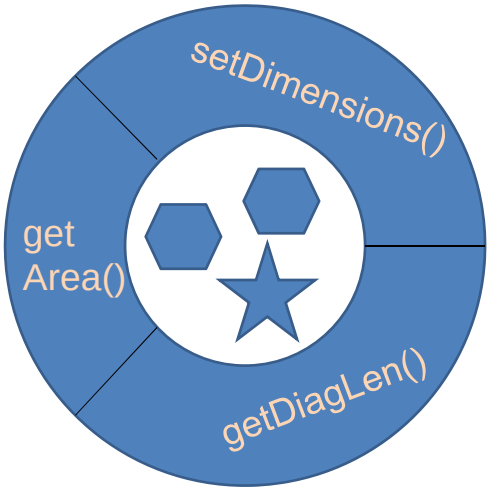
OK, *classYear* is *public*

Error!!! *major* is *private*

More about private

- Hides instance variables and methods inside the class/object. The private variables and methods are still there, holding data for the object.
- Invisible to external users of the class
  - Users cannot access private class members directly
- **Information hiding and data encapsulation**

Information Hiding



Example: Rectangle

```
public class Rectangle
{
    public int width;
    public int height;
    public int area;

    public void setDimensions(int newWidth,
        int newHeight){
        width = newWidth;
        height = newHeight;
        area = width * height;
    }

    public int getArea(){
        return area;
    }
}
```

```
Rectangle box = new Rectangle();
box.setDimensions(10, 5);
System.out.println(box.getArea());


// Output: 50


box.width = 6;
System.out.println("The rectangle
   with edges " + box.width + "
   and " + box.height + " has area
   size " + box.getArea());


// Output: The rectangle with
   edges 6 and 5 has area size 50

// Wrong answer!
```

Accessors and Mutators

- How do you access private instance variables?
- Accessor methods (a.k.a. get methods, **getters**)
  - Allow you to look at data in private instance variables
- Mutator methods (a.k.a. set methods, **setters**)
  - Allow you to change data in private instance variables

Example: Student

```
public class Student
{
    private String name;
    private int age;

    public void setName(String studentName) {
        name = studentName;
    }
    public void setAge(int studentAge) {
        age = studentAge;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
}
```

Mutators

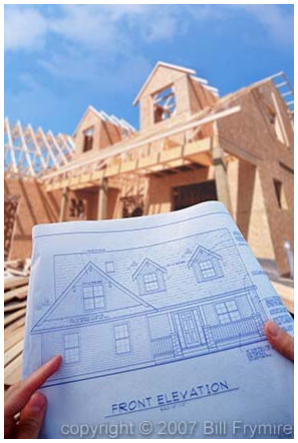
Accessors

Private Methods

- **Why make methods private?**
- Helper methods that will only be used from inside a class should be private
  - External users have no need to call these methods
  - Can change a private method without needing to change external code.

Constructors

- A class is like a blueprint
- Constructors "build" and initialize new objects
- Special methods that are called when (and **only** when) creating a new object



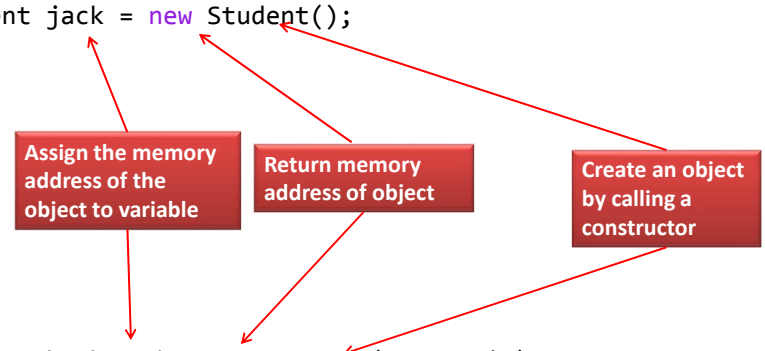

```
Student jack = new Student();
```

Calling a constructor

Creating an Object

Create an object *jack* of class *Student*

```
Student jack = new Student();
```




```
Scanner keyboard = new Scanner(System.in);
```

Create an object *keyboard* of class *Scanner*

Constructors

- Can perform any action you write into a constructor's definition
  - There are no specific rules about what's in a constructor
- Meant to perform initializing actions
  - Usually, initializing values of instance variables by the creator of the object

Similar to Setter Methods

- However, constructors *create* an object in addition to setting the values of instance variables
- Like methods, constructors can have parameters
- Syntax: like method declarations, but no return type, and have the same name as the class.

public Name() {….}

Example: Pet Class

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

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

    public static void main(String[] args)
    {
        Pet p = new Pet();
    }
}
```

Default constructor

Call constructor

The Same as Initialization

```
public class Pet
{
    private String name = "No name yet.";
    private int age = 0;
    private double weight = 0;

    public static void main(String[] args)
    {
        Pet p = new Pet();
    }
}
```

Default constructor not declared – but still exists

Call default constructor (so an object is created)

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've never written a constructor but you can still create objects

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

Next class

- Continue with methods…