THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Tuesday November 25, 2014

Jay Aikat
Fall 2014
TR 9:30 – 10:45, GS-G100

Previous Class

- What did we discuss?

Announcements

- **Assignment 6: Due Wed, Dec 3**
- **Last Class is Tue, Dec 2 – Final exam review**
- **Study guides will be up by Mon, Dec 1**
- **FINAL EXAM: Tue, Dec 9 – 8-11 AM**

Announcements

- **Remember – one lowest assignment grade is dropped**
- **One lowest quiz grade is dropped**
- **Labs – 5% extra credit**
- **Survey quiz next class – 2% extra credit**

Assignment 6

http://cs.unc.edu/~aikat/courses/comp110/assignments/Assignment6.htm

But first, a little review…

Defining a Class

```
public class Student
{
    public String name;
    public int classYear;
    public double gpa;
    public String major;
    // ...

    public String getMajor()
    {
        return major;
    }

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

Class name

Data
(or attributes, or instance variables)

Methods

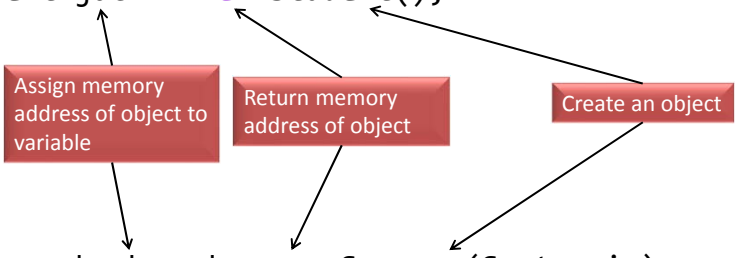
Creating an Object

- Syntax
  - **ClassName objectName = new ClassName();**
- What does the statement do?
  - The computer will create a new object, and assign its memory address to **objectName**
  - **objectName** is sometimes called a class type variable
    - It is a variable of class type **ClassName**
- Why do we need new?
  - So we know ClassName() is not executing a method but creating an object

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*

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

Methods

```
public class Student
{
    public String name;
    public int classYear;
    public double gpa;
    public String major;
    // ...

    public String getMajor()
    {
        return major;
    }

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

Methods

Methods

- Two kinds of methods
  - Methods that return a value
    - Examples: String's **substring()** method, String's **indexOf()** method, String's **charAt()** method, etc.
  - Methods that return nothing
    - Example: **System.out.println()**
- "Return" means "give back"
  - A method can give back a value so that other parts of the program can use it, or simply perform some actions

Defining Methods that Return a Value

- Method heading: keywords
  - public: no restriction on how to use the method (more details later)
  - *Type*: the type of value the method returns
- Method body: statements executed
  - **Must be inside a pair of brackets**
  - **Must have a return statement**

```
public String getMajor()
{
    return major;
}
```

return Statement

- A method that returns a value must have *at least one* return statement
- Terminates the method, and returns a value
- Syntax:
  - return expression;
- expression can be any expression that produces a value of type specified by **the return type** in the method heading

Methods that Return a Value

As usual, inside a block (defined by braces), you can have multiple statements

```
public String getClassYear()
{
    if (classYear == 1)
        return "Freshman";
    else if (classYear == 2)
        return "Sophomore";
    else if ...
}
```

Calling Methods that Return a Value

- Object, followed by dot, then method name, then ()
  - **objectName.methodName();**
- Use them as a ***value*** of the type specified by the method's return type

```
Student jack = new Student();
jack.major = "Computer Science";

String m = jack.getMajor(); // Same as String m = "Computer Science"

System.out.println("Jack's full name is " + jack.getName());
// Same as System.out.println("Jack's full name is " + "Jack Smith");
System.out.println("Jack's major is " + m);
```


Defining Methods That Return Nothing

- Method heading: keywords
  - public: no restriction on how to use the method (more details later)
  - void: the method returns nothing
- Method body: statements executed when the method is called (invoked)
  - **Must be inside a pair of brackets**

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

Methods that Return Nothing

```
public void printData()
{
    System.out.println("Name: " + name);
    System.out.println("Major: " + major);
    System.out.println("GPA: " + gpa);
}
```


Calling Methods that Return Nothing

- Object, followed by dot, then method name, then ()
  - The same as a method that returns a value
  - **objectName.methodName();**
- Use them as *Java* ***statements***

```
Student jack = new Student();
jack.classYear = 1;

jack.increaseYear();

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

Assignment 6

Now, let's look at Assignment 6 again…

http://cs.unc.edu/~aikat/courses/comp110/assignments/Assignment6.htm

Classes

- MYOChocolates (main method here)
- KitKatFactory (list of ingredients – your database here)

KitKatFactory

- Constructor – initialize ingredient list and total price
- Properties – think variables
- Ingredients - array of three integers:
  - Cocoa
  - Wafers
  - Sugar
- Total price

KitKatFactory

```
// Update the ingredientList array with new quantities
    // Also update totalPrice variable
    public static void updateIngredientList (int numOfKitKats){...}

    // Returns the ingredientList array
    public int[] getIngredientList (){...}

    // Sets the totalPrice variable to new value
    public static void setTotalPrice(double newTotal) {...}

    // Returns the totalPrice
    public double getTotalPrice(){...}
    public static void printIngredientList(){...}
```

MYOChocolates

```
KitKatFactory myFactory = new KitKatFactory();

//Calling methods
myFactory.methodname
```

What if input is not what you expect?

Next class (Tue, Dec 2)

- Final Exam review
- Survey quiz – 2% extra credit