



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

FB 314, aikat@cs.unc.edu



Previous Class

- What did we discuss?



Today

- Quiz today
- Assignment3: DUE Thu, 11/5 @ 11:55 PM
- Today – Classes and Methods



Using more than one 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++;
    }
}
```

```
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);
    }
}
```



Control Flow

- Program control flow
 - execution always begins with the first statement in the method `main`
 - other methods execute only when called
- Method control flow
 - when a method is invoked, the **flow of control** jumps to the method and the computer executes its code
 - when complete, the flow of control returns to the place where the method was called and the computer continues executing code

COMP 110 - Fall 2015

5



Instance Variable and Local Variable

- Instance variables
 - Declared in a class
 - Confined to the class
 - Can be used anywhere in the class that declares the variable, including inside the class' methods
- Local variables
 - Declared in a method
 - Confined to the method
 - Can only be used inside the method that declares the variable

COMP 110 - Fall 2015

6



Local Variable Example

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

    public void printInfo()
    {
        String info = name + ": " + classYear;
        System.out.println(info);
    }

    public void increaseYear()
    {
        classYear++;
    }

    public void decreaseYear()
    {
        classYear--;
    }
}
```

- *classYear* and *name* are instance variables
- can be used in any method in this class

- *info* is a local variable declared inside method *printInfo()*
- can only be used inside method *printInfo()*

COMP 110 - Fall 2015

7



Local Variable Example

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

    public void printInfo()
    {
        String info = name + ": " + classYear;
        System.out.println(info);
    }

    public void increaseYear()
    {
        classYear++;

        info = "My info string"; // ERROR!!!
    }

    public void decreaseYear()
    {
        classYear--;
    }
}
```

The compiler will not recognize the variable *info* inside of method *increaseYear()*

COMP 110 - Fall 2015

8



Local Variable Example

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

    public void printInfo()
    {
        String info = name + "; " + classYear;
        System.out.println(info);
    }

    public void increaseYear()
    {
        classYear++;

        String info = "My info string"; // OK
    }

    public void decreaseYear()
    {
        classYear--;
    }
}
```

Variable *info* in *increaseYear* method not affected by variable *info* in *printInfo* method in class *Student*

COMP 110 - Fall 2015

9



Local Variable Rule

- Usually, a variable is only accessible in its surrounding brackets

```
public class Variable {
    String a = "a";

    public void f() {
        String b = "b";
        if (a.equals("b")) {
            String c = "c";
        }
    }
}
```

COMP 110 - Fall 2015

10



Methods with Parameters

- Compute the square of this number
 - 5
 - 10
 - 7
- I could give you any number, and you could tell me the square of it
- We can do the same thing with methods

COMP 110 - Fall 2015

11



Methods with Parameters

- Parameters are used to hold the value that you *pass* to the method
- Parameters can be used as (local) variables inside the method

```
public int square(int number)
{
    return number * number;
}
```

Parameters go inside the parentheses of method header

COMP 110 - Fall 2015

12



Calling a Method with Parameters

```
public class Student
{
    public String name;
    public int classYear;
    // ...
    public void setName(String studentName)
    {
        name = studentName;
    }
    public void setClassYear(int year)
    {
        classYear = year;
    }
}
```

COMP 110 - Fall 2015

13



Calling a Method with Parameters

```
public static void main(String[] args)
{
    Student jack = new Student();
    jack.setName("Jack Smith");
    jack.setClassYear(3);
}
```

Parameters/
Arguments

COMP 110 - Fall 2015

14



Methods with Multiple Parameters

- Multiple parameters separated by commas

```
public double getTotal(double price, double tax)
{
    return price + price * tax;
}
```

- When calling a method, the order, type, and number of arguments must match parameters specified in method heading

COMP 110 - Fall 2015

15



Methods with Multiple Parameters

```
public class SalesComputer
{
    public double getTotal(double price, double tax)
    {
        return price + price * tax;
    }
    // ...
    SalesComputer sc = new SalesComputer();
double total = sc.getTotal("19.99", Color.RED);
double total = sc.getTotal(19.99);
    double total = sc.getTotal(19.99, 0.065);
    int price = 50;
    total = sc.getTotal(price, <del>0.065</del>);
```

Automatic typecasting

COMP 110 - Fall 2015

16



Calling Methods from Methods

- A method body can call another method
 - Done the same way:
receiving_object.method();
- If calling a method in the same class, do not need receiving_object:
 - method();
- Alternatively, use the **this** keyword (can be omitted)
 - **this**.method();

COMP 110 - Fall 2015

17



Calling Methods from Methods

```
public class Student
{
    public String name;
    public int classYear;
    public void setName(String studentName)
    {
        name = studentName;
    }
    public void setClassYear(int year)
    {
        classYear = year;
    }
    public void setNameAndYear(String studentName, int year){
        this.name = studentName; // or this.setName(studentName);
        this.classYear = year; // or this.setClassYear(year);
    }
}
```

COMP 110 - Fall 2015

18



Next class

- More on Classes and Methods