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
- Assignment 3: DUE Thu, 11/5 @ 11:55 PM
- Today – Classes and Methods

Using more than one Class

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

    // ...
    public String getMajor() {
        return major;
    }
    public void increaseYear() {
        classYear++;
    }
}
```

```java
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

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
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()

The compiler will not recognize the variable info inside of method increaseYear()
Local Variable Example

```java
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`

Local Variable Rule

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

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

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

public int square(int number)
{
    return number * number;
}
### Calling a Method with Parameters

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

### Calling a Method with Parameters

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

• Multiple parameters separated by commas
  
  ```java
  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

```java
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, 0.065);
}
```

Automatic typecasting
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();

```java
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);
    }
}
```
Next class

- More on Classes and Methods