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

- Assignment 3: DUE Thu, 11/5 @ 11:55 PM
- Women’s meetup “110 Girl Code” Tuesday 7PM in SN 014 [Email: kris@cs.unc.edu for further info]
- Today – Classes and Methods

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
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

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

Parameters go inside the parentheses of method header

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
Methods with Multiple Parameters

```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);
}
```

Calling Methods from Methods

- A method body can call another method
  - Done the same way:
    ```java
    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();`
Calling Methods from Methods

```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);
    }
}
```

Pre- and Postcondition Comments

- **Precondition comment**
  - States conditions that must be true before method is invoked

- **Example**

```java
/**
   * Precondition: The instance variables of the calling object have values.
   * Postcondition: The data stored in (the instance variables of) the receiving object have been written to the screen.
   */
public void writeOutput()
```
Pre- and Postcondition Comments

- Postcondition comment
  - Tells what will be true after method executed
- Example

```java
/**
 * Precondition: years is a nonnegative number.
 * Postcondition: Returns the projected population of the receiving object after the specified number of years.
 */
public int predictPopulation(int years)
```

Calling a Method from main

```java
public class Exercise1 {
    public static void main(String[] args){
        display();
    }
    static void display() {
        System.out.println("This is an exercise to work with methods");
    }
}
```
Calling a Method that takes input

```java
public class Exercise1 {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 10;
        int num3 = 15;
        addNumbers(num1, num2, num3);
    }

    static void addNumbers(int n1, int n2, int n3) {
        int result = n1+n2+n3;
        System.out.println("The result is "+result);
    }
}
```

Calling a Method that returns a string

```java
import java.util.*;

public class Exercise1 {
    public static void main(String[] args) {
        String s1 = getUserInput();
        System.out.println("Welcome to COMP110, "+s1);
    }

    static String getUserInput() {
        Scanner keybrd = new Scanner(System.in);
        System.out.println("Please input your full name");
        String user1 = keybrd.nextLine();
        return user1;
    }
}
```
Calling a Method that takes input and returns a value

```java
public class Exercise1 {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 10;
        int num3 = 15;

        int sum = addNumbers(num1, num2, num3);
        System.out.println("The result is " + sum);
    }

    static int addNumbers(int n1, int n2, int n3) {
        int result = n1+n2+n3;
        return result;
    }
}
```

Calling a Method from a method

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

• More on Classes and Methods