**Administrative**

**Today’s session**

Method

Method call

Method declaration

Method stack

Method overloading

Graphics

**Session Topics**

**Method**

● A **method** is a block of code for performing a task.

● A method may be defined by:

✓ Someone else and made available to an application via an imported class.

✓ The developer for an application-specific purpose.

● A developer-defined method is useful when there is a block of code to execute from two or more spots in an application.

**Method call**

● A method is called using:

✓ **<method-name>** if the method is defined in the same class.

✓ **<class-name>.<method-name>** if the method is defined as static in another class.

✓ **<object-name>.<method-name>** if the method is defined as non-static in another class.

**Method declaration**

● There are two types of methods since a method may or may not return a value when it completes:

✓ A **void method** runs but *does not* return a value.

✓ A **value method** runs and then returns a value.

● A void method has syntax:

public static void <method-name>(<parameter-list>)

{

<block>

}

**Where:**

**void** means th:e method does not return a value.

**<parameter-list>** is a list of zero or more inputs needed by the method.

● A value method has syntax:

public static <data-type> <method-name>(<parameter-list>)

{

<block>

return <expression>;

}

**Where:**

**<data-type>** is any primitive or class type.

**<parameter-list>** is a list of zero or more inputs needed by the method.

**return <expression>;** appears one or more times in the method. The data type of <expression> must match data type of **<data-type>**.

● A void method may be called wherever a statement may appear.

● A value method may be called wherever an expression may appear. The data type of the returned value must be compatible the expression context.

● Zero or more **parameters** may be defined after the method name and between the parentheses.

● Parameters are the inputs to a method.

● Each parameter is separated from the next one with a comma.

● A parameter list has syntax:

<data-type> <parameter-1>, <data-type> <parameter-2>, ….

● Whenever a method is called, there must be one **argument** provided for each parameter.

● Parameters and arguments must match in number, order, and data type.

● A parameter is also known as a **formal parameter**.

● An argument is also known as an **actual parameter**.

● See **Methods, fields, and, local variables** sample application on Blackboard.

**Method stack**

● A **stack data structure** is used to store a sequence of method calls.

● A stack is also known as a last-in-first-out (LIFO) data structure.

● Suppose there are three methods: **M1**, **M2**, and **M3**, and M1 calls M2 and M2 calls M3. When M1begins, its local variable information is placed on the stack. When M1 calls M2, M1 is suspended, and the M1 next statement pointer and M2 local variable information is placed on the stack. When M2 calls M3, M2 is suspended, and the M2 next statement pointer and M3 local variable information is placed on the stack. When M3 ends, its stack information is removed, and M2 is resumed with information from the stack. When M2 ends, its stack information is removed, and M1 is resumed with information from the stack.

● Because stack information is removed when a method ends, and stack information includes local variables, the values of local variables are lost.

● Data from methods may be retained by using **fields** instead of local variables.

● A field is a variable that is available throughout a class (it is global to the class).

● A field is declared near the top of a class, outside of any methods.

● See **Methods, fields, and, local variables** sample application on Blackboard.

**Method overloading**

● A **method signature** consists of:

✓ The method name.

✓ The data types of any parameters.

● A method signature does not include the return type, if any, of a method.

● Method signature example:

public void printDate(int year, String month, int day)

{

System.out.println("Date is " + month + " " + day + ", " + year);

}

The signature of this method is **printDate(int, String, int)**.

● Two or more methods within a class may have the same name, but not the same method signature.

● **Method overloading** occurs when two or more methods have the same name but different sets of parameter data types.

● The Java Virtual Machine determines which method version to use by matching the method call to one of the signatures for that method.

● Method overloading example:

public void setPhoneNumber(String number)

{

phoneNumber = "734-" + number;

}

public void setPhoneNumber(String areaCode, String number)

{

phoneNumber = areaCode + number;

}

The signature of the first method is **setPhoneNumber(String)**

The signature of the second method is **setPhoneNumber(String, String)**

● See **Methods, fields, and, local variables** sample application on Blackboard.

**Graphics**

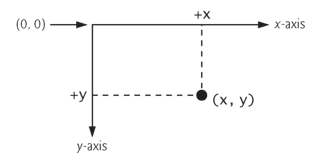
● Java enables the drawing of graphics.

● Drawing is performed within a panel (class JPanel).

● A panel is composed of pixels identified by a coordinate system.

● Each pixel is located by ordered pair (x-coordinate, y-coordinate).

● The coordinate system identifies the upper-left corner as (0, 0) and the lower-right corner as (width, height) where width and height are the extents of the panel.



● Here are some drawing methods:

| Method | Purpose |
| --- | --- |
| drawLine(x1, y1, x2, y2) | Draw a line from (x1, x2) to (y1, y2). |
| drawOval(x, y, width, height) | Draw an oval with upper left corner at (x, y) and given width and height. |
| drawRect(x, y, width, height) | Draw a rectangle with upper left corner at (x, y) and given width and height. |
| drawRoundRect(x, y, width, height, arcWidth, arcHeight) | Draw a rounded rectangle with upper left corner at (x, y), given width and height, and given arc width and height. |
| fillOval(x, y, width, height) | Draw a filled oval with upper left corner at (x, y) and given width and height. |
| fillRect(x, y, width, height) | Draw a filled rectangle with upper left corner at (x, y) and given width and height. |
| fillRoundRect(x, y, width, height, arcWidth, arcHeight) | Draw a filled rounded rectangle with upper left corner at (x, y), given width and height, and given arc width and height. |
| setColor(Color.<color>) | Change the drawing color to one from class Color. |

● Calls to these drawing methods are made from within method **paintComponent**.

● Subsequent calls to these drawing methods are triggered with method **repaint**.

● See **Drawing** sample application on Blackboard.