University at Buffalo
The State University of New York

cse@buffalo

CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce

343 Davis Hall

alphonce@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

**Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6*

ROADMAP

Last time

- Graphics
- Event handling

Today

- Primitives
- Control structures

Coming up

- Collections

ANNOUNCEMENT

LAPTOP

On Wednesday this week, bring a laptop with Eclipse and the WebCAT submitter installed to class.

We will be doing paired coding exercises in lecture.

If you do not have a laptop you can bring, you will be paired up with a student who has one.

REVIEW

Live coding result

```
package graphics;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

public class Application {
    public Application() {
        JFrame window;
        window  = new JFrame("Our very first graphical program");
        window.setVisible(true);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton b;
        b = new JButton("Click me");
        window.add(b);
        ActionListener x;
        x = new EventHandler();
        b.addActionListener(x);
    }
}
```

Live coding result

```
package graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class EventHandler implements ActionListener {

    public EventHandler() {
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Oh stop that!");
    }

}
```

MOVING ON

ActionListener objects can serve as event handlers for JButtons.

An ActionListener object must be associated with a JButton to play this role:

```
JButton b = new JButton("Click me!");
ActionListener e = new EventHandler();
b.addActionListener(e);
```

A JButton is a component which can react to mouse clicks.

Events

Clicks on buttons, mouse movements, etc. are all considered events.

A program can react to events by setting up event handlers.

An event handler defines what should happen when a particular event occurs.

The component which gives rise to an event is decoupled from the part of the code that handles the event.

This is called the *observer pattern*.

General form:

http://www.ibm.com/developerworks/java/tutorials/j-patterns/j-patterns.html
http://www.oodesign.com/observer-pattern.html
http://en.wikipedia.org/wiki/Observer_pattern

Observer pattern in Java

An observer is called a listener in Java

Button clicks are “ActionEvents”.

Handlers for ActionEvents are ActionListeners.

An event-generator can have many listeners

Use “addActionListener” method to register a listener with a component

PRIMITIVES

(and odds and ends)

Brief detour: primitives

To this point we have seen only so-called reference types, types whose values are accessed via a reference.

Reference types:

- classes
- interfaces

Java also has an inventory of so-called primitive types.

- so-called because their values are atomic (they have no accessible internal structure)

primitives

The value of a primitive type is not an object:

- primitives have no instance variables
- methods cannot be called on primitives

The value of a primitive type is stored directly in a variable.

Primitive values can be expressed using literals (see next two slides).

the boolean type has two values, `true` and `false`

boolean operators: && (and), || (or) and ! (not)

P	Q	P && Q	P || Q	!Q
true	true	true	true	false
true	false	false	true	true
false	true	false	true	-
false	false	false	false	-

examples:

```
boolean x;                  // declaration
x = true;                   // assignment
boolean y = false;          // combined
boolean z = x && y;         // using && operator
```

primitive type: int

the int type includes integral values in a given range (we'll return to this in a later lecture): 0, +1, -1, +2, -2, …

int operators:

+, integer addition	(operator type is int × int → int)
–, integer subtraction	(operator type is int × int → int)
*, integer multiplication	(operator type is int × int → int)
/, integer division (quotient)	(operator type is int × int → int)
%, integer remainder	(operator type is int × int → int)

examples:

```
int x = 5;
int y = 3;
int q = x / y;      // q has value 1
int r = x % y;      // r has value 2
                    // note: x = q * y + r
```

Odds and ends review

The next several slides review some odds and ends, some of which were discussed in lab 5 as well.

import

An import is used to allow unqualified use of a name which would otherwise need to be fully qualified

form: import <fully qualified name> ;

examples:

```
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

These examples allow the names GridLayout, JButton, JFrame and Jlabel to be used without full qualification. This improves both the writability and readability of code.

java.lang package

The java.lang package is special in that its elements are all imported by default.

java.lang.System

java.lang.String

java.lang.String

String is a class.

String is special – we can create instances with a special syntax: a sequence of characters enclosed in double quotes:

- "This is a String"
- "So it this"

String objects are immutable.

- the contents of a String cannot be changed

String concatenation

‘+’ is the name of the String concatenation operator

‘+’ is a binary operator, meaning it takes two arguments (also called operands)

‘+’ is an infix operator, meaning it is written between its two arguments

“Hi” + “there” is an expression whose value is a new String object “Hithere”.

“The answer is ”+17 is an expression whose value is “The answer is 17”. The int expression 17 is converted to a textual equivalent, the String “17” (which consists of the two characters ‘1’ and ‘7’)

static members: methods and variables

static methods

- a static method is invoked on a class, rather than an object
- a static method has no access to instance variables
- its invocation record has no 'this'

static variables

- a static variable is accessed via a class, rather than an object
- a static variable is often declared 'public'
- a static variable is often given a name of all upper-case letters, as in java.awt.Color.RED

main method

```
public static void main(String[] args) {
        …
}
```

main method is standard entry point for a Java program

main method is invoked by Java runtime system

static

- reserved word
- indicates member is associated with CLASS not INSTANCE

parameter of main

- square brackets are special syntax used with arrays
  - we will discuss arrays at start of CSE116
- the parameter ‘args’ is initialized with ‘command line arguments’ – arguments given on the command line when the program is run

System.out.println

The System class is defined in the java.lang package.

The name ‘System’ can therefore be used in an unqualified way, even without an explicit import directive.

‘out’ is a public and static variable of the System class, whose type is PrintStream

The ‘PrintStream’ class defines a static method named ‘println’

‘println’ accepts an argument of any type, and prints a textual representation of the argument on the console (see the output in the console view if running in Eclipse, or the terminal window from which the program was started if running outside of Eclipse).