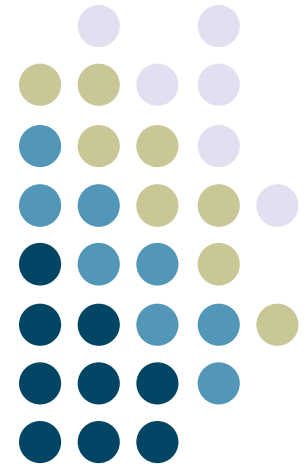


Intro to Java 2



**Georgia
Tech**



John Stasko

CS 6452

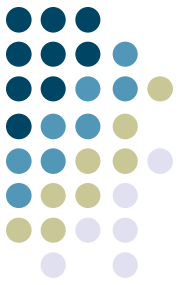
Prototyping Interactive Systems

Learning Objectives

- Control flow
 - Conditionals
 - Iteration
- Random numbers
- Arrays
- Java API



Some Leftovers



```
i++;    // Similar to i = i + 1;
```

```
i += 12;    // Same as i = i + 12;
```

Some Leftovers

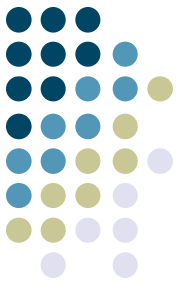


Block statement

```
{  
    // Statements here  
}
```

can go anywhere a single statement can go

Control Flow



- How execution flows through a program
 - Conditionals
 - Iteration

Conditionals



```
if (total > 20)
    sum = sum + total;
```

```
if (choice == x1) {
    value = value * 10;
    j = k / i;
}
```

```
if (!done && (total < 100))
    choice = scan.nextLine();
```

! – not
&& - and
|| - or

Conditionals



```
if (total < 20)
    sum = sum + total;
else if (total < 50) {
    sum = sum + 100;
    total = 0;
}
else if (total < 100)
    sum = 1000;
else {
    a = 1;
    b = 2;
}
```

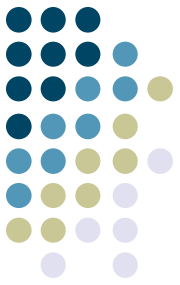
Multiple conditions

Iteration

- Multiple constructs
 - while
 - do
 - for



while



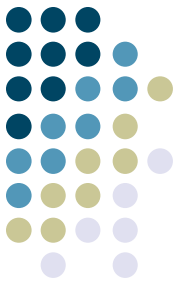
```
while (count < total) {  
    sum = sum + count;  
    count = count + 1;  
}
```

Condition checked first
Body might never be done

`continue` – jump back up to check condition

`break` – exit the loop and move to next statement

Find the Average



```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int i, count=0, sum=0;
    System.out.println("Enter numbers, -1 to quit");

    while (true) {
        i = scan.nextInt();
        if (i == -1)
            break;
        else {
            sum = sum + i;
            count++;
        }
    }
    System.out.println("Ave is "+sum/(double)count);
}
```

Not the best code

do



```
do {  
    i = scan.nextInt();  
    System.out.println(i);  
} while (i != 0);
```

Body done at least once

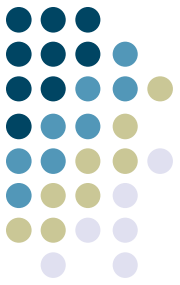
for

```
for (int i=0; i<max; i++) {  
    j = j + 1;  
    System.out.println(i);  
}
```

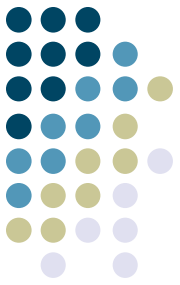
Body might never be done

Equivalent to

```
i = 0;  
while (i < max) {  
    j = j + 1;  
    System.out.println(i);  
    i = i + 1;  
}
```

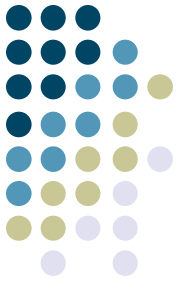


When to Use



- Use `for` when your bounds are known
- Use `while` when unsure bounds
 - Maybe done never
- Use `do` when unsure bounds
 - Always done once

Palindromes



```
public static void main (String[] args)
{
    String str, another = "y";
    int left, right;

    Scanner scan = new Scanner (System.in);
    while (another.equalsIgnoreCase("y")) // allows y or Y
    {
        System.out.println ("Enter a potential palindrome:");
        str = scan.nextLine();

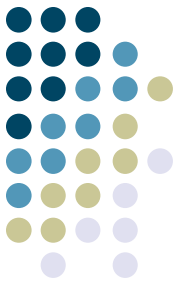
        left = 0;
        right = str.length() - 1;

        while (str.charAt(left) == str.charAt(right) && left < right)
        {
            left++;
            right--;
        }
        System.out.println();

        if (left < right)
            System.out.println ("That string is NOT a palindrome.");
        else
            System.out.println ("That string IS a palindrome.");

        System.out.println();
        System.out.print ("Test another palindrome (y/n)? ");
        another = scan.nextLine();
    }
}
```

Random Numbers



```
import java.util.Random;

int i;

Random gen = new Random();
i = gen.nextInt();
```

Calls

```
float nextFloat()
    // 0.0 <= x < 1.0
int nextInt()
    // all +, -, 0
int nextInt(int num)
    // 0 <= x < num
```

Java API



docs.oracle.com/javase/8/docs/api/

Java™ Platform Standard Ed. 8

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

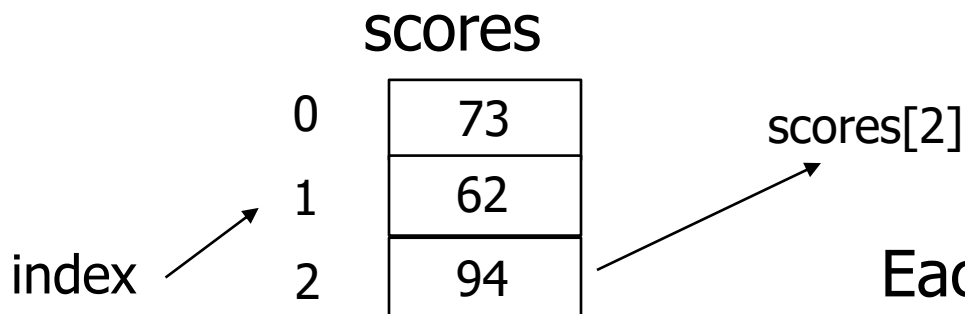
Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.

Arrays



- Lists of values
- Positions (index) are numbered
- Homogeneous type
- Uses [] notation



Each of these spots can be used anywhere an integer can

Arrays

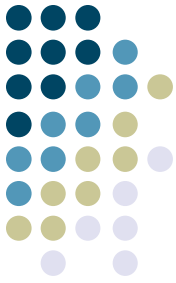


- Array is an object
 - Implications?
- Declaration

```
int[] scores = new int[3];
```

```
for (i=0; i<length; i++)  
    exam[i] = i * 10;
```

Counting Letters



```
import java.util.Scanner;

public class LetterCount
{
    public static void main (String[] args)
    {
        final int NUMCHARS = 26;

        Scanner scan = new Scanner (System.in);

        int[] upper = new int[NUMCHARS];
        int[] lower = new int[NUMCHARS];

        char current;    // the current character
                        //   being processed
        int other = 0;   // counter for
                        //   non-alphabetic

        System.out.println ("Enter a sentence:");
        String line = scan.nextLine();

        // Count the number of each letter occurrence
        for (int ch = 0; ch < line.length(); ch++)
        {
            current = line.charAt(ch);
            if (current >= 'A' && current <= 'Z')
                upper[current-'A']++;
            else
                if (current >= 'a' && current <= 'z')
                    lower[current-'a']++;
                else
                    other++;
        }

        // Print the results
        System.out.println ();
        for (int letter=0; letter < upper.length; letter++)
        {
            System.out.print ( (char) (letter + 'A') );
            System.out.print (": " + upper[letter]);
            System.out.print ("\t\t" + (char) (letter + 'a') );
            System.out.println (": " + lower[letter]);
        }

        System.out.println ();
        System.out.println ("Non-alphabetic characters: " +
                            other);
    }
}
```

Learning Objectives



- Control flow
 - Conditionals
 - Iteration
- Random numbers
- Arrays
- Java API

Next Time

- Classes and objects
 - Instance data
 - Methods
 - Visibility
 - Inheritance

