CS 180 Problem Solving and Object Oriented Programming

Fall 2011

http://www.cs.purdue.edu/homes/apm/courses/CS180Fall2011/

Notes for Week 7:

Oct 3-7, 2011

This Week:

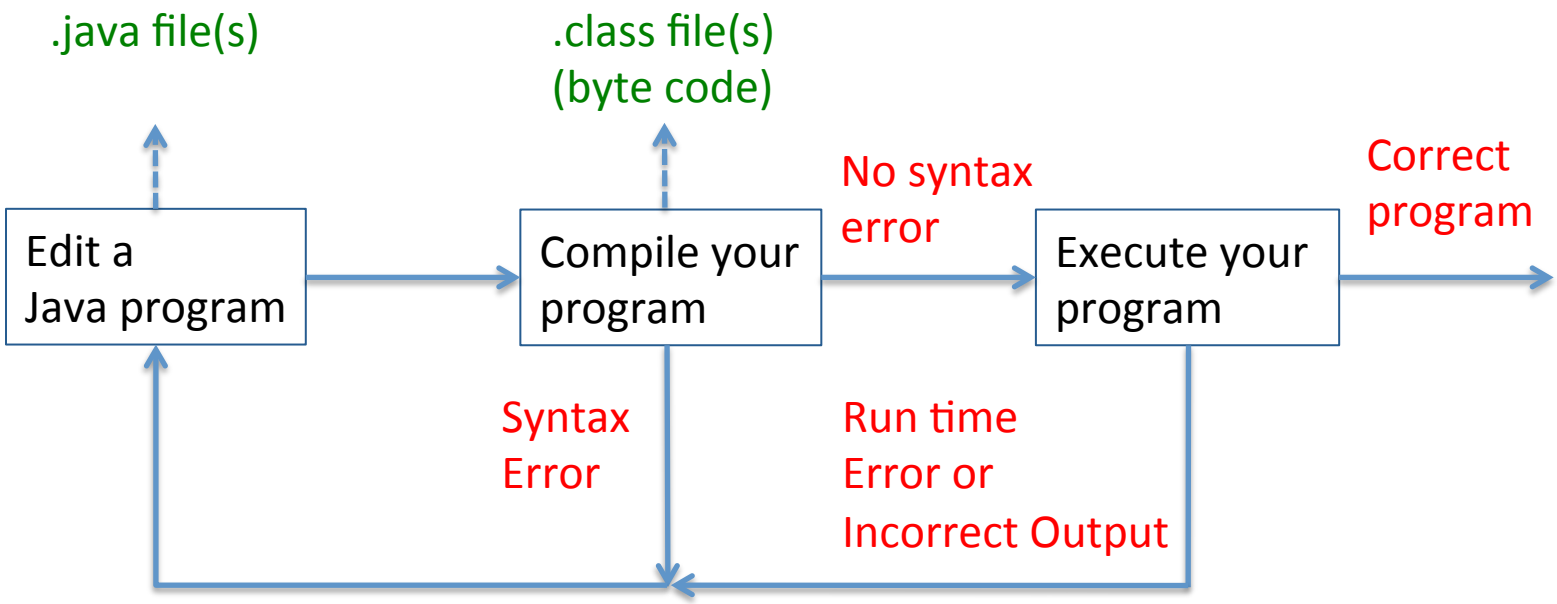
10/4

1. Review
2. Arrays
3. Demo
4. Multi-dimensional arrays
5. Visual depiction of arrays

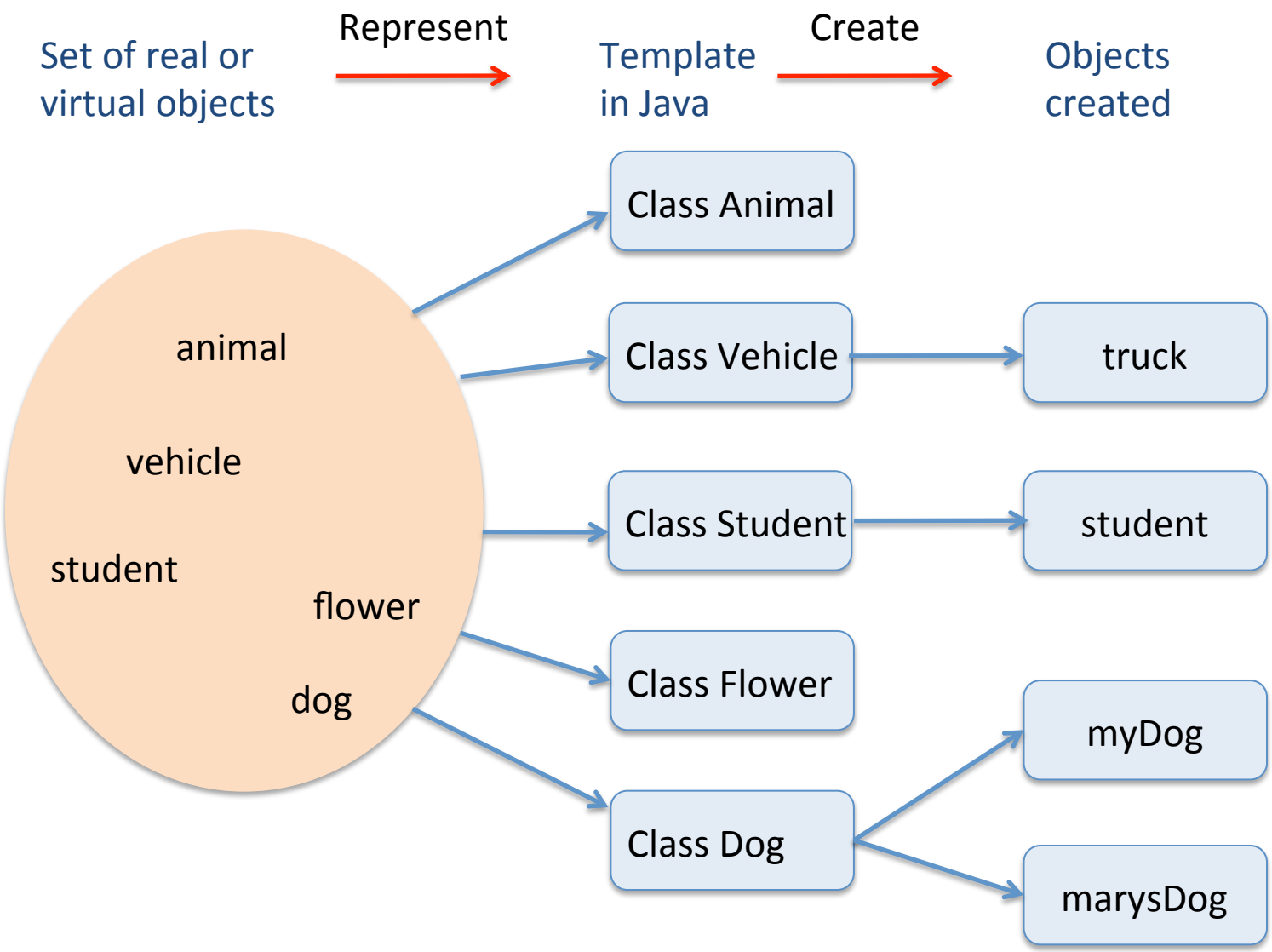
Aditya Mathur
Department of Computer Science
Purdue University
West Lafayette, IN, USA

Review

The edit, compile, execute cycle



Classes and Objects



©Aditya Mathur. CS 180. Fall 2011. Week 7

String

Is a sequence of zero or more Unicode characters.

Examples:

"Greetings!"
"Your total tax is:"
"Please enter the price:"
" "
""

Declaration:

```
String name="Beatles"; // name is an object of type String
String store="Macy's"; // store is an object of type String
```

String

Expressions:

```
String firstName, middleInitial, lastName;

String fullName;

fullName=firstName+middleInitial+lastName;

String message="Please enter the price:";
```

Strings: Other operations

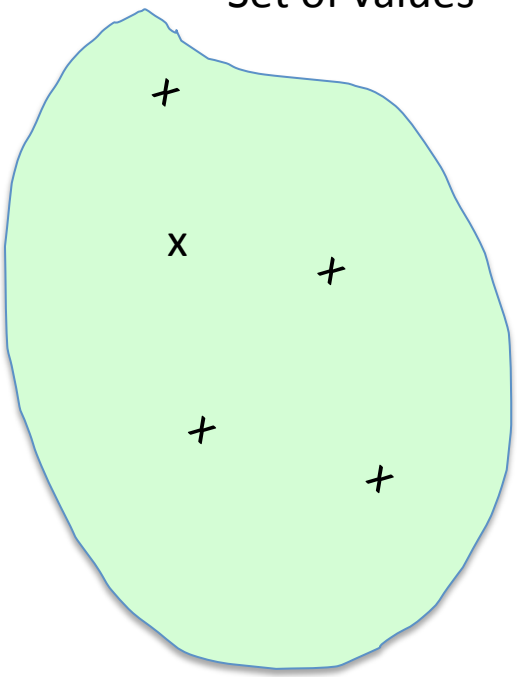
You may apply a variety of operations to strings. Examples follow.

Statement	Operation used
String commend="Bently,"+ " good girl!";	Catenation
char firstChar=commend.charAt(0);	Character extraction
movieName.equals("Fugitive")	Comparison
String.valueOf(29)	Conversion to String
commend.charAt(5)	Extract character at position 5 from string commend
commend.replace("Bently", "Ziggy");	Replace "Bently" by "Ziggy"
commend.substring(startIndex, endIndex);	Extract substring of characters starting from position startIndex until and including position endIndex-1.

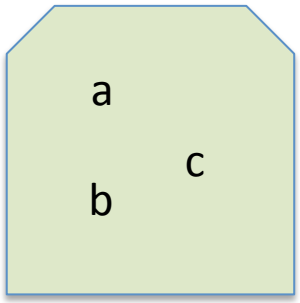
Types

Types

Set of values

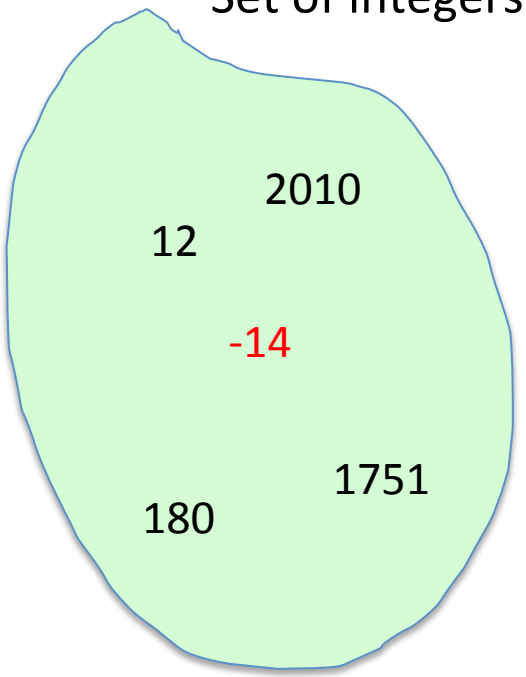


Set of Operations

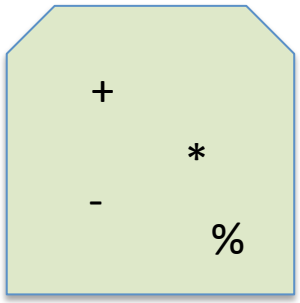


Primitive types: short, int, long

Set of integers



Set of Operations



short: 2 bytes
int: 4 bytes
long: 8 bytes

Integer.MAX_VALUE: 2^{31} -1

Integer.MIN_VALUE: -2^{31}

Long.MAX_VALUE: 2^{63} -1

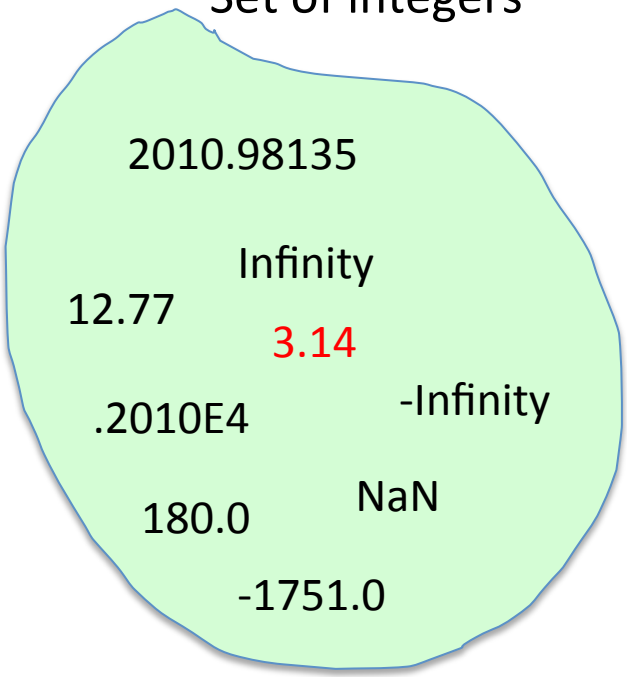
Long.MIN_VALUE: -2^{63}

Primitive types: short, int, long: Examples

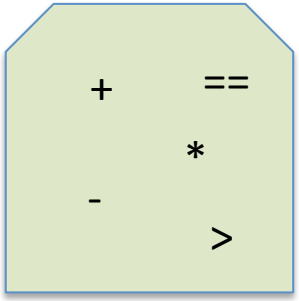
Real world entity or expression	Type	Possible name in Java
Population of a country	int	countryPopulation
Age of a patient (in years)	short	patientAge
Number of different ways to arrange 15 books in a bookshelf	long	bookArrangementCount
Difference between two integers	int or long	diff
Number of web sites	long	numberOfWebSites

Primitive types: float, double

Set of integers



Set of Operations (sample)



float: 4 bytes
double: 8 bytes

Float.MAX_VALUE: 3.40282347e+38f Float.MIN_VALUE: 1.40239846e-45f

Double.MAX_VALUE: 1.79769313486231570e+308

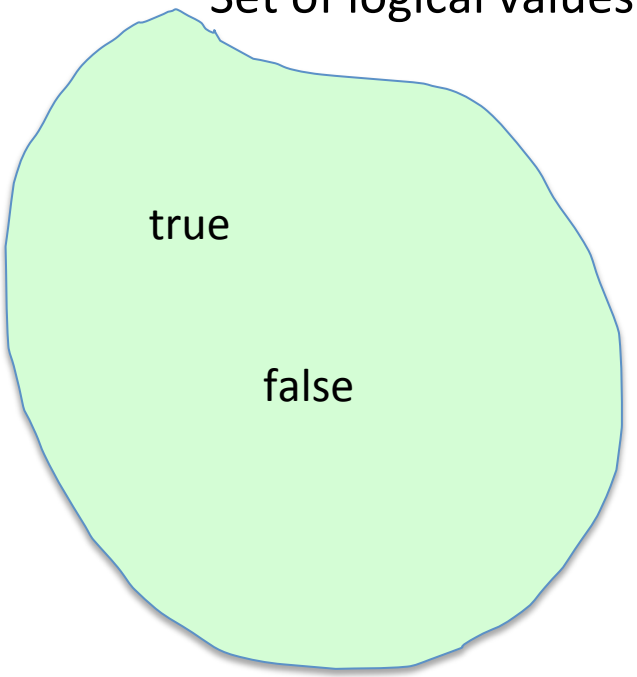
Double.MIN_VALUE: 4.94065645841246544e-324

Primitive types: float, double: Examples

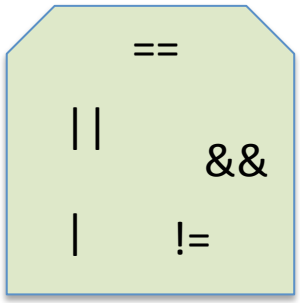
Real world entity or expression	Type	Possible name in a Java program
Height of a person	float	height
Voting percentage	float	votePercent
Wavelength of green light	double	wavelengthLIght
Price of a ticket	float	ticketPrice
π	double	pi (Note: PI is a constant in Java)

Primitive types: boolean

Set of logical values



Set of Operations (sample)



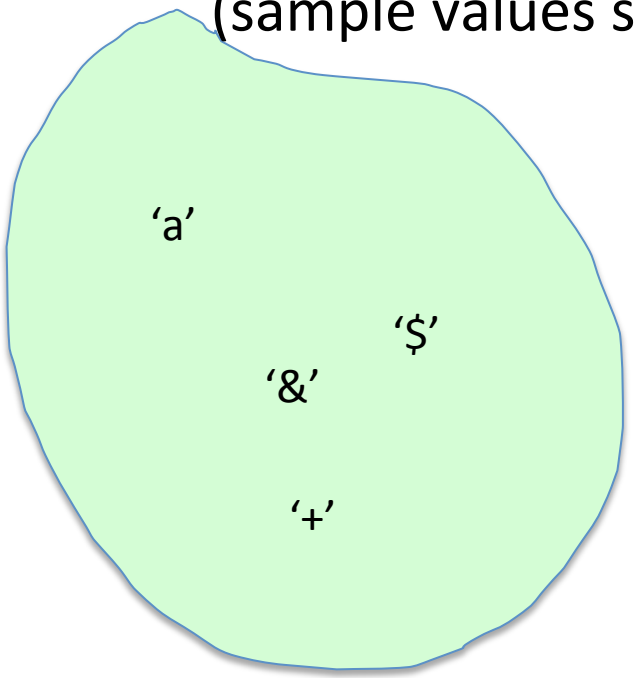
boolean: 1 bit; size not defined

Primitive types: boolean: Examples

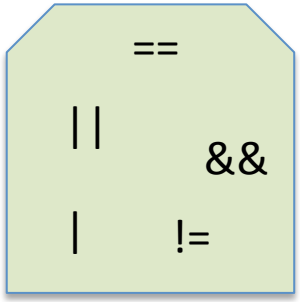
Real world entity or expression	Type	Possible name in a Java program
Value of x<y;	boolean	result
she/he drives a car	boolean	canDriveCar
Class ended	boolean	classEnded

Primitive types: char

Set of characters
(sample values shown)



Set of Operations
(sample)



char: 2 bytes, unicode character

Primitive types: char: Examples

Real world entity or expression	Type	Possible name in a Java program
Middle initial	char	middleInitial
Letter of the alphabet	char	letter
US currency sign	char	usCurrency

Names

Used to denote classes, objects, data

Contain characters; must start with a letter, or a $ sign or an underscore.

Examples: height, area1, Dog, $great

Length unlimited, case sensitive.
Dog and dog are different names.

Convention: All class names begin with an uppercase letter; all other names begin with a lower case letter.

Constants

A constant is something that cannot change during program execution.

Examples:

Integer constants: 0, 1, -1, +24, 29, 300009998, O14, 0x1B

Floating point constants: 0.0, -2.345e28, -0.000976512

Boolean constants: true, false

Character constants: ‘ ‘, ‘a’, ‘A’, ‘$’

String constants: “”, “ “, “Hi!”, “Alice in Wonderland”

Named Constants

A constant can be named and the name used instead of the constant itself.

Examples:

```
final float pi=3.14159;

final boolean  dogsExist=true;
```

Variables

A variable is something whose value may change during program execution.

Every variable has a name and a type.

Every variable must be declared before it is used.

Declarations

```
int age;

float  height, area;

String name;

boolean  lightsOn;

int x=1, y=0;

String firstName="Harry";
```

Simple expressions

Expressions are used to compute “something”.

```
float  x, y, z;

x*y+z; // Arithmetic expression, results in float value

x<y;  // Boolean expression, results in boolean value

String firstName=“Mary”, lastName= “Jones”;

firstName+” “+lastName; // Results in a string
```

Assignment statement

An assignment statement allows assigning the value of an expression to a variable.

```
float p=x*y+z; // p gets the value of x*y+z

boolean q=x<y;  // q gets the value of x<y

String firstName="Mary", lastName= "Jones";

String name= firstName+" "+lastName;
```

Handling characters

```
char first='a';
char second='b';
```

Compare the values of two character variables:

Yes → `if(first==second)`
NO → `if(first.equals(second))`

Extract a character from a string:

```
String course="CS180";
char c=course.charAt(pos); // Extracts the  character at position pos
```

Mixing Strings and characters

```
char first='a';
String s1="b";
```

The following are correct:

```
String s2=first+s1;
System.out.println(first+s1);
```

Following are not correct:

```
if (s1==first)
if(s1.equals(first))
if(first.equals(s1))
```

Readings and Exercises for Week 7

Readings:

Chapter 7: 7.1, 7.2, 7.3, 7.4

Self help exercises:

7.3, 7.4, 7.5, 7.7 [Optional]

Arrays

Arrays: What are these?

A homogeneous collection of variables or objects. All elements of an array are of the same type.

Examples:

Array of integers where each integer represents the age of a patient.

Array of cars where each element of the array denotes a specific car.

Array of flowers where each element of the array denotes a flower.

Arrays: When to use?

When there is a need to retain data in memory for efficient processing. Data is created once and used many times.

Examples:

Array of flights: Search for a specific flight

Array of cars: Search for a specific car, or find the average price.

Array of laptops: find the cheapest laptop

Declaring an Array

Indicates age is an array variable

```
int [] age;
/* age  refers to an array of integers;
   e.g. age of people in a database*/

double [] weight;
/* weight refers to an array of doubles;
   e.g. weights of items shipped*/

String [] name;
/* name refers to an array of elements
   each of type String; e.g., names of
   people in a database*/
```

Declaring an Array of objects

```
Bird [] bird;
/* Bird refers to a class; bird is an array
   where each element is an object of
   type Bird. */
```


```
Aircraft [] fleet;
/* Aircraft refers to a class; fleet is an
   array where each element is an
   object of type Aircraft. */
```

Creating an Array

```
int [] age=new int[10];
/* age  is an array of 10 elements each of type int*/

double [] weight=new double[100];
/* weight is an array of 100 elements each of type  double*/

String [] name=new String[15];
/* name is an array of 15 elements each of type  String*/

Aircraft [] fleet=new Aircraft[150];
/* fleet is an array of 150 elements each of type  Aircraft*/
```

Single dimensional array: Example 1

```
int [] cloudyDays=new int [12] // average cloudy days/month
```

Anchorage, Alaska

0	1	2	3	4	5	6	7	8	9	10	11	← Index
[19	18	18	18	20	20	22	21	21	21	20	21]	← Data

cloudyDays[0] cloudyDays[5] cloudyDays[11]

Single dimensional array: Example 2: Declaration

```
Car [] carStock=new Car [5] // Cars at a dealership

        public class Car{
            String make; // e.g., Ford
            String model; // e.g., Fusion
            int year;   // e.g., 2011
            long msrp; // e.g., US$23,145
            Picture carPic;
        }
```

Single dimensional array: Example 2: Visual

Car [] carStock=new Car[5] // Cars at a dealership

CS180 Dealership in West Lafayette

0 1 2 3 4 ← Index

[ ]

{Ford, fusion, 2011, 23150 }

....

{Porsche, Boxster Spyder, 72000, }

}

← Object Reference

carStock[0] carStock[4]

Accessing an array element: general syntax

Name of the array

Index, starts from 0, must be an int

name [expression]

Note: First element of an array is located at index 0.

Examples:

```
age[i]; // 0<=i<10

fleet[k]; // 0<=k<150

weight[i+j]; // 0<=(i+j)<100
```

Accessing an array element

```
int [] age=new int[10];
int p= age[5]; /* p gets element at index 5 of array age*/

double [] weight=new double[100];
double hisWeight=weight[i]; /* hisWeight gets the element of
    weight at index i*/

Aircraft [] fleet=new Aircraft[150];
Aircraft rental=fleet[k]/* rental gets the element of fleet
    located at index k */
```

Assigning a value to an array element

```
int [] age=new int[10];
age[i]=15; /* (i+1)th element of age becomes 15. */

int [] age={-15, 20, 30}  // All elements initialized

double [] weight=new double[100];
weight[k]=weight[k]-10; /* (k+1)th element of the weight is
    decremented by 10.*/

Aircraft [] fleet=new Aircraft[150];
fleet[i]=new Aircraft(); /* (i+1)th element of fleet gets a new
    Aircraft */
```

Iterating through elements of an array

In many problems we need to iterate through all or a few of the elements of an array.

Such an iteration is accomplished by using a for or a while loop.

Announcements

1. Feast with Faculty: Today at 6:30pm. Ford Dining Hall.

2. Next week:

   . Class does not meet Monday and Wednesday.
   . No lab sessions..
   . Recitations WILL MEET

3. Exam 1 grades should be on the blackboard by Sunday afternoon.

Commonly used syntax of the for statement

Executed once prior to entering the loop

Checked each time prior to entering the loop body; loop terminates if condition is false

Executed at the end of each iteration

```
for( init; condition; increment){
        Loop body; i.e., a sequence of statements.
    }
```

Problem 1: Initialize an array of elements

```
// Initialize an array of n elements to random numbers
// between 0 and 1.
final int n=100;// Array size
double [] rnum=new double [n]; // Create array to hold numbers
for( int i=0; i<n; i++){
    rnum[i]=Math.random() // Initialize the ith element
} // End of for
```

Problem 2: Initialize an array (read from console)

```
Scanner s=new Scanner(System.in);
int [] a=new int [10];
for( i=0; i<10; i++){
        a[i]=s.nextInt(); // Sets a[i] to number entered.
    } // End of for
```

Problem 3: Find average of numbers in an array

```
final int n=100;// Array size
double [] rnum=new double [n]; // Create array to hold numbers
for( int i=0; i<n; i++){
    rnum[i]=Math.random() // Initialize the ith element
} // End of for
double sum=0.0; // Initialize sum.
for( int i=0; i<n; i++){
    sum-sum+rnum[i]; // Add the ith element to sum
} // End of for
double average=sum/n;
```

Problem 4: Count positive (including 0) and negatives.

```
int sum=0; int [] a={-2, 12, 30, -4, 90};
int pos=0, neg=0; // Initialize counts.
for(int  i=0; i<a.length; i++){
        if(a[i]>=0){
            pos=pos+1; // Increment positive count
        }else{
            neg=neg+1; // Increment negative count
            }
    }// End of for
```

i++ is equivalent to i=i+1;

Problem 5: Search in an array: Step 1 initialize

```
String [] wList={"Emilio", "Semion", ”Ziggy"};
Scanner s=new Scanner (System.in);
String wanted=s.next();  // Get name to be searched
boolean found=false; // Not yet found
```

Problem 5: Search in an array: Step 2 set up loop

```
boolean found=false; // Not yet found
int next=0; // Points to the first element in wanted list
while(???) {
    if(wanted.equals(wList[nextItem])) // Compare
        found=true; // If equal then we have found the wanted!
    else
        next=next+1; // else point to the next item on the list
}//End of loop
```

Problem 5: Search in an array: Step 3 complete loop condition

```
int next=0; // Points to the first element in wanted list
boolean found=false; // Not yet found
while(next<wList.length&&!found) {
    if(wanted.equals(wList[next])){ // Compare
      found=true; // If equal then we have found the wanted!
     }else{
     next=next+1; // else point to the next item on the list
     }
}//End of loop
```

Problem 5: Search in an array: Display output

```
if(found)
        System.out.println(wanted+  " exists in database");
else
        System.out.println(wanted+  " not in   database")
} // End of program
```

Problem 6: Sort an array in ascending order

Example:

Input array of integers: [5 4 -1 2]

Sorted array of integers: [-1 2 4 5]

Problem 6: Sort algorithm: Bubble sort (Compare and exchange)

Pass 1

Step	1	2	3	End of pass 1
	5	4	4	4
	4	5	-1	-1
	-1	-1	5	2
	2	2	2	5

Total 3 steps in Pass 1

Pass 2

Step	1	2	End of pass 2
	4	-1	-1
	-1	4	2
	2	2	4
	5	5	5

Total 2 steps in Pass 2

Problem 6: Sort algorithm: Bubble sort

Pass 3

Step 1 → End of pass 2

-1	-1
2	2
4	4
5	5

Total 1 step in Pass 3

Problem 6: Sort algorithm: Bubble sort: Analysis

Given: Array of n elements

Number of passes needed:

Number of steps (comparisons) in

Pass 1:

Pass 2:

.

.

Pass i:

Total number of steps:

Problem 6: Sort algorithm: Bubble sort: Warning

Bubble sort is not a recommended algorithm for large arrays (e.g. an array of, say, 100 elements).

Better algorithms:

Insertion sort
Merge sort
Quick sort

Challenge: Code the bubble sort algorithm (on your own).

Problem 7: Plant biology: Statement

A plant biology lab maintains a collection of leaves.

The lab director wishes to create a simple database that contains several entries. Each entry is a plant name, type of its leaf, and a brief description of the leaf in the collection. Example:

WhitePine	Needle	NA
Heather	Compound	NA

Write a program that reads data from a file as mentioned above and allows a user to search for a tree in the database.

Problem: Understanding

A plant biology lab maintains a collection of leaves.

From where can my program read the leaf data? If it is in a file, then what is the name of that file?

The lab director wishes to create a simple database that contains several entries.

How much data is available?

Write a program that reads data from a file as mentioned above and allows a user to search for a tree in the database.

In what form will the user input a request?

Java program for plant biology lab.

Incremental stepwise development!

Tasks

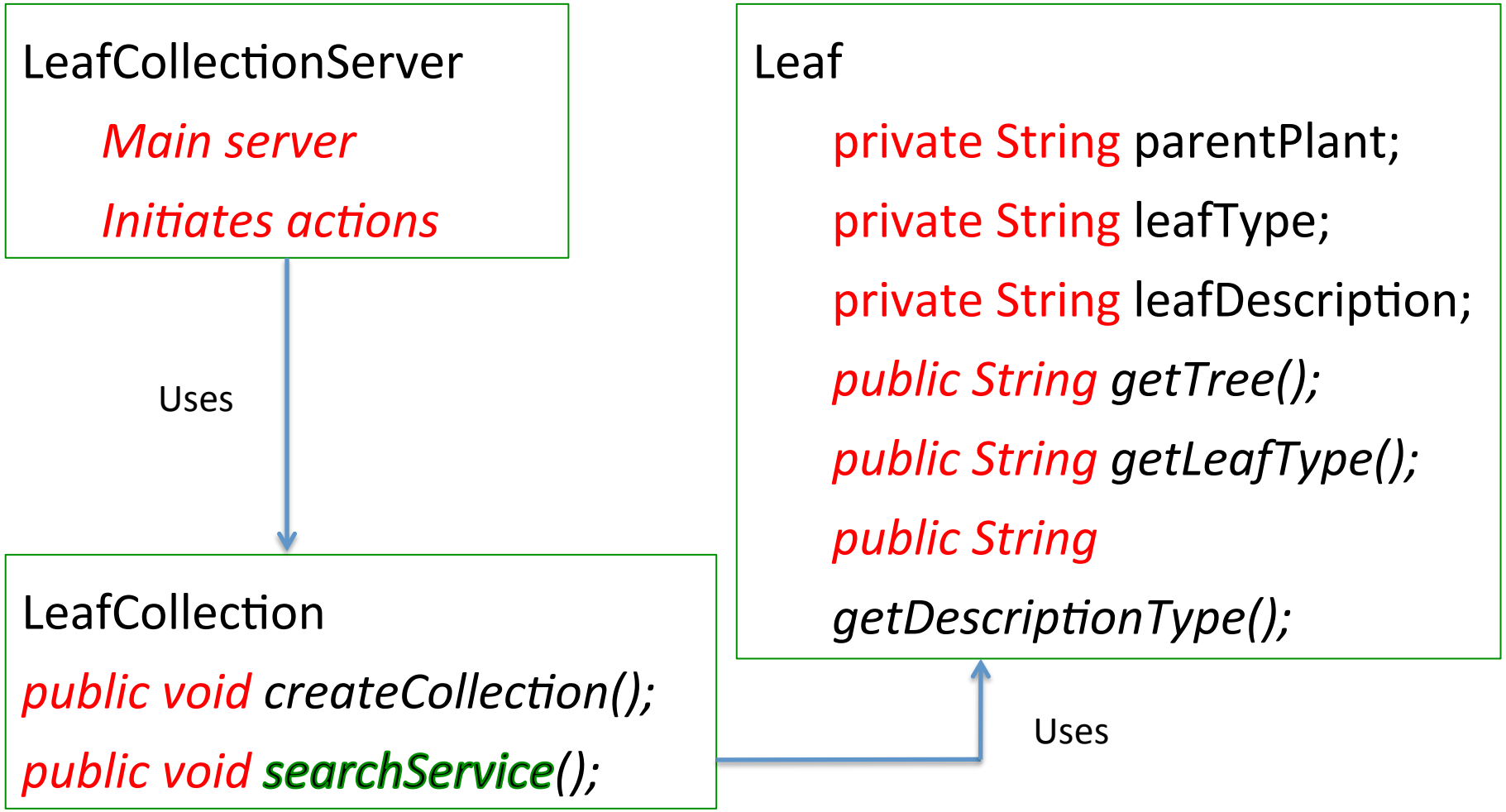
1. Read plant collection data from a file
2. Provide search service to the user
3. Exit with a bye bye message when done.

Let us assume….

1. LeafCollection class is available
2. It provides the following methods:
   - *createCollection()*
   - *searchService()*
3. Leaf class is available. It provides the following methods:
   - *public void getTree();*
   - *public void getLeafType();*
   - *public void getDescriptionType();*

Why did we make the above assumptions?

Overall design: Classes



LeafCollectionServer

Back to the Java program. Version 1.0.

LeafCollection: Design

- This is a class.
- Provides two methods:
  - *createCollection()*
  - *searchService()*
- But before we can write these methods we must have a way to read from a file and save in memory all the data about the collection!

Question: How should we store leaf collection data?

- *Recall: For each leaf, we have its parent tree, its type, and its description.*

LeafCollection: Design Question

Question: How should we store leaf collection data?

- *Recall: For each leaf, we have its parent tree, its type, and its description.*

Answer: ??

LeafCollection

Back to the Java program. Version 2.0.

Leaf: Design: Attributes

Recall, for each leaf, we have the following available data:

Name of parent tree
Name of the leaf
Description

Leaf: Design: Methods

Recall, for each leaf, we have the following available data:

```
public String getLeafName();
public String getTreeName();
public String getDsecription();
```

Leaf class

Back to the Java program. Version 3.0.

Arrays: Multidimensional

Example 1

MEAN MONTHLY CLOUDY DAYS IN ARIZONA

		0	1	2	3	4	5	6	7	8	9	10	11
		J	F	M	A	M	J	J	A	S	O	N	D
0	FLAGSTAFF	12	11	12	9	7	4	9	8	5	7	8	11
1	PHOENIX	10	9	8	6	3	2	4	4	3	4	6	9
2	TUCSON	10	9	9	6	4	3	9	7	4	5	6	10
3	WINSLOW	12	10	9	7	5	4	8	6	4	6	8	10
4	YUMA	9	6	6	4	2	1	3	3	2	3	5	8

rows columns

```
int [][] cloudyDays=new int  [5][12]
```

What is the value of cloudyDays[1,8]?

Declaration

Cars [] inventory=new Car [3][];

3 rows and undefined number of columns

Each row represents make of a car and column represents models

		0	1	2
0	Chevy	Avalanche	Traverse	
1	Honda	Accord	Fit	Civic
2	Toyota	Camry	Corolla	Rav4

Arrays: Typical errors

- Index out of bounds [Run time error]

```
int [] a=new int [10];

a[i]=x; // i is greater than 9 or less than 0
```

- Element not initialized [Compile time error]

```
String [] name;

name[i]="Bob"; // element not initialized
```

Week 7: October 3-7, 2011
Hope you enjoyed this week!

Questions?

Contact your recitation instructor. Make full use of our office hours.