To read a text file, we will use a Scanner

Scanner sc = new Scanner(System.in);

String phrase = “I am happy it is Friday”;

Scanner sc = new Scanner(phrase);

sc.next()- to grab one word at a time from it

Scanner sc = new Scanner(someFileObject);

What is someFileOject

File class – represents a program’s connection to a file on the hard drive

File f = new File(stringThatIsThePathToTheFile);

Suppose we want to read the file stored on c:\temp\story.txt.

/home/temp/story.txt

Scanner fsc = new Scanner(new File(“c:\temp\\story.txt.”));

While (fsc.hasNextLine( )){

Line = fsc.nextLine( ).trim( );

System.out.println(line.length( ));

}

fsc.close( );

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(new File(filename))));

Pw.println(“happy Friday”);

Pw.print

Pw.printf

Pw.close();

Array- collection of data where all the data is stored in consecutively numbered slots in memory

Indexes start at 0

Int[ ] numbers = new int[10];

Numbers[0] = 17;

Numbers.length;

For (int I = 0; I < numbers.length; i++){

Numbers[i] = 7;

}

Int[ ] numbers = {7, 3, 4};

Numbers have 3 slots and they have values 7, 3, 4.

Iterator approach to navigating an array

Only when you want to use the value, not set it

For(int num : numbers){

System.out.println(num);

}

ArrayList

Java.util.ArrayList

ArrayList<Integer> numbers = new ArrayList<Integer>();

Numbers.add(val)

Numbers.add(pos, val) – insert

Numbers.size()--# of vaues in ArrayList

Get(index)—returns the value of a location index

Clear()- removes the items

indexOf(val)—returns the index where val was found or -1 if not found

lastIndexOf(val)- returns the last place it was found

remove(index)—removes value at index location

remove(objectValue)—remove all instances of particular object

set(index, value)—sets value at position index to value

class – data type that has both data(attributes) and functions(methods) inside it

encapsulation – placing data alongside functions that will use that data inside one structure

two broad categories of classes

library- it consists entirely of static functions- functions that are called from the class name

don’t declare variable/ create objects with these

math is a library class

blueprint- designed to create objects from – to declare variables of

provides the design/ the instructions for building something with that data and those functions

usually will have multiple objects that you build according to the blueprint

most of the functions will be non-static (can’t call them from name of the class)

class BluePrint{

declare your data—most of which will be declared private and non-static

declare your constructors- special functions that create the objects of the class

declare your functions most of which will be public

}

Information hiding

All data should be private so that outside the class we can’t directly set it or get it

Access their value through public get (accessor) and set(mutator) functions

Constructor

A function

Does not have a return type(not even void)

Almost always public

They have the same name as the name of the class

There are usually at least two(function overloading- different parameter lists)

Default constructor – takes in nothing to initialize the values of the variables

Non-default constructor- takes in values and uses them to initialize the private variables

Should use the set functions you’ve written to set the values of the variables

Inheritance

A class can descend from or inherit from another class

The descendants inherits the data members and functions of the parents

Every class descends from java.lang.Object

11 functions that all classes inherit

@Override

Public String toString(){

Return some string representation

}

Whenever we try to treat an object like a string

For example when we try to print this toString() will automatically be called

Model view controller(MVC)

Model classes just hold and manage data(employee)

View classes show that data(employeeWriter)

Controller classes coordinate the interplay b/t them (main app)