Foo aFoo //declaring Foo variable; aFoo is a reference

aFoo = new Foo(); // creating Foo object

**composition** is building a new class using instance variables of previously defined class

**Instance variable : define data within the object**

**define operation to be used by object**

Private int number; reference type like other class

**Constructor, do not return, exact name as class name**

public String toString(){

StringBuilder S = new StringBuilder();

S.append("Width: " + width);

return S.toString();

MyRectangle R1, R2, R3;

R1 = new MyRectangle(100, 50, 80, 20);

System.out.println("R1: " + area(R1));

Public **Circle**(double aRadius)

Radius = aRaduis

**Accessor, getter** public int getSize()

**Mutator, setter** b.append(“father”);

class names: System, JFrame, ArrayList,

method name: println, close, add

**statement**

variable declaration statements: int count;

assignment statement: count = 2;

**reference types**

store value is a reference to an object that is stored else where

String s = new String(“ boy…”);

String s = “”;

Stored value is a reference (location)

object is stored else where (string)

Call method: (**instance method**)

ClassName.methodName(parameter\_list)

ClassName.objectname.methodName(parameter\_list)

Public static void methodName(parameter)

**encapsulation and data abstraction**

-operations on the data are considered to be part of the data type; no need to know how data is represented

-allow **instance variable** to be separated or hidden from the user of a class

- **data abstraction** enable user do not require direct knowledge of these variables in order to use a class

**Polymorphism**

-operations used with a variable are based on the class of the object being accessed

-parent type and sub-type objects can be accessed in a consistent way

String[] info = user[i].split(",");

ArrayList<String> A = new ArrayList<String>();

A.add(1,"Zany"); | A.remove(0); | A.getsize();

**Binary search, recursion**

public static int binarySearch(int[] array, int first, int last, int value)

int middle;

if (first > last)

return -1;

if (array[middle] == value)

return middle;

else if (array[middle] < value)

return binarySearch(array, middle + 1, last, value);

else

return binarySearch(array, first, middle - 1, value);

**Recursion**

A java method can call any other public java method

Main() is just a method itself, and we have called other methods from it

public static int factorial(int n)

if(n <= 1) //base case

return 1;

else // recursive case

return n \* factorial(n - 1);

2

public static void reverse(int[] data)

recursive\_reverse(data, 0, data.length - 1);

public static void recursive\_reverse(int[] data, int firstIndex, int lastIndex)

{

if(lastIndex <= firstIndex)

return;

int x = data[firstIndex];

data[firstIndex] = data[lastIndex];

data[lastIndex] = x;

recursive\_reverse(data, firstInde+1, lastIndex-1);

**This** reference

public int counter;

public aClass(){}

public int aMethod(){

int counter;

counter = 5; //method variable

this counter = 6; //instance variable

}

public class anotherClass{

aClass c = new aClass();

println(c.counter); //instance variable

Wrapper class, wrap objects around primitive class, thus making compatible with other class

Integer i = new Integer(5);

Integer.parseInt(); is a static method that enables us to convert from a String to an int

**Exception**

**Try**, **Catch**(IOException e), **finally**

If catch (Exception e) is the first, NO OTHER catches in the block could ever execute

**Checked Exceptions**  // not handle in method

public static void main(String args[]) throws IOException

IOException or FileNotFoundException

**Unchecked Exeptions**

Method not required to explicitly “throw” these These include RunTimeException

ArrayIndexOutOfBoundsException

NumberFormatException

Try 第一行错了 后面都不会执行

**abstract method** in superclass, expects to be overridden in a subclass.

AccessSpecifier abstract ReturnType MethodName(ParameterList);

Subclasses of an abstract class must implement all abstract methods, or they too must be declared abstract

public abstract class Grocery

{ private String name;

public Grocery(String aName)

{

name = aName; }

public abstract double cost();

public String toString()

{ return name;

public BulkItem extends Grocery

{

private double quantity;

private double costPer;

public BulkItem(String aName, double q, double c)

{

super(aName);

quantity = q;

costPer = c;

}

public String toString()

{

String s = super.toString() + " Quantity: " + quantity;

}

public double cost()

{ return quantity \* costPer;

Super class references can access subclass objects

The only methods that are callable are those that were initially defined in the super class

the Car class (subclass) inherits the attributes and methods from the Vehicle class (superclass)

method initially defined within subclass cannot be accessed via a superclass reference

if (A[i] instanceof Fish)

Fish F = (Fish) A[i];

System.out.println(F.getWaterType() + "fun!");

**Interface** must be public

All methods in an interface are implicitly public

Provide an implementation of the display method

state a list of method, may also contain static variables

but no instance variables are allowed.

// 1) Static methods // 2) Default methods

//abstract super-class with no instance data

public interface Laughable

{ public static String chuckle = "Hardy Hardy";

public void laugh();

default public void wacky()

{ System.out.println("default method");

public static void weasel()

{ System.out.println("static method");

public class Person implements Laughable, Booable

Add hoc polymorphism / Overloading: methods within the same class (or in a common hierarchy) share the same name but have different method signature (parameters), compile-time concept

True (run time) polymorphism / Overriding: superclass is redefined in a subclass with an identical method signature

Static variable

Variables that are associated with the class itself rather than individual object

Access through the class using ClassName.variableName

Or through the object using objectName.variableName

静态变量被所有的对象所共享，在内存中只有一个副本，它当且仅当在类初次加载时会被初始化。而非静态变量是对象所拥有的，在创建对象的时候被初始化，存在多个副本，各个对象拥有的副本互不影响。

protected declaration is between public and private

protect data and methods are directly accessible in the base class and in any subclasses and in the package

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | alpha | beta | alphasub | gamma |
| modifier | y | y | y | y |
| protected | y | y | y | n |
| no modifier | y | y | n | n |
| private | y | n | n | n |

