**class**

class are blueprints of data

public class Foo{}

**object**

instance of class contain specified data and methods

java object are instance of classes, need to be constructed

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

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

两字符串的地址是否相同，也就是是否是同一个字符串的引用

== shows use that it is the same object

.equals() show that the value are in some way the same

(different object)

**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**

StringBuilder b1 = new StringBuilder(“hhhhhhhh”);

Public **Circle**(double aRadius)

Radius = aRaduis

**Accessor, getter**

public int getSize()

public String toString()

**Mutator, setter**

b.append(“father”);

public boolean addDisk(Disk aDisk)

public void setRadius(set aNewRadius)

radius = aNewRadius

**other identifiers must:**

begin with a-z or A-Z, or (\_),

follow by zero or more letter, digits(0-9), (\_), $

keywords:

if, else, class, int

predefine identifiers

class names: System, JFrame, ArrayList,

method name: println, close, add

**literals:**

integer 1

string “yes”

character ‘5’

**statement**

variable declaration statements

int count;

**assignment statement**

count = 2;

**primitive types:** boolean

byte < int < long< float< double

float point number is double by default

you must cast to less precise float pi = (float) 3.14;

double = int 必须范围大的在前面

Float num = Float.parseFloat(str);

**Convert string to numbers**

int x = (int) 2.7; => 2

integer division && floating point division

int x = 5 / 2; x is 2

double y = 5 / 2; y is 2.0 double z = 5.0 / 2; z is 2.5

prefix and postfix x++ ====== x ++y ====== (y+1)

**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)

**random number**

Random rand = new Random();

double hh1 = (rand.nextDouble()); // 0.0~1.0

int hh = (rand.nextInt(2)); // 0~1

**Scanner**

Scanner s = new Scanner(System.in);

val = s.nextInt(); name = s.nextLine();

**delimiter**, character or set of characters that distinguish one token form another Scanner class use white space nextInt()

**compiler** **errors** are problems with the syntax or structure of a program. these prevent the program from being compiled properly. forget semicolon. logic errors are problems with what the code actually does.

**Logic errors** are problems with what the code actually does. else is always associated with unassociated if

if(1 + 1 == 2)

System.out.println(“yes!”);

if( 2 + 2 == 5)

System.out.println("shit");

else

System.out.println(“hh");

用switch 的话记得每个do后面加break

还有default

Math.pow(x,2)就是平方 Math.sqrt(4)开方

Call method: (**instance method**)

ClassName.methodName(parameter\_list)

ClassName.objectname.methodName(parameter\_list)

System.out.println(“shit”);

method calls

System.out.println(“surprise!”);

Static method

Public static void methodName(parameter)

**ARRAY**

double[] user; //declare array variable (reference)

user = new double[arr\_size]; // create array object

user[5] = 123.0; // index array to access locations

user.length int[] user = new int[6];

string = null; // array of reference object int = 0;

String[] S = new String[5] // create five locations

import java.util.Random; “%.2f”, sales

Import java.io.\* // file class

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

File inputFile = new File(“aText.txt”)

Scanner inputFileScanner = new Scanner(inputFile);

PrintWriter outputFileWriter(new File(“output.txt”)

While(inputFileScanner.hasNextLine())

System.out

inputFileScanner.close()

public class MyRectangle

{

// Declare instance variables here.

private int startX = 0;

public MyRectangle(int x, int y, int w, int h)

{

startX = x;

}

public **int** area()

{

int area = width \* height;

return area;

}

// Note how a StringBuilder is

// utilized, since (as we discussed in lecture)

// it can be modified

// without having to create a new object each

// time (unlike a String).

public String toString()

{

StringBuilder S = new StringBuilder();

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

S.append(" Height: " + height);

S.append(" X: " + startX);

S.append(" Y: " + startY);

return S.toString();

}

public boolean isInside(int x, int y)

{

if (x >= startX && x <= (startX + width))

{

if (y >= startY && y <= (startY + height))

return true;

else, return false;

}

else, return false;

}

}

public static void noOpSwap(int one, int two)

{

System.out.println("One: " + one + " Two: " + two);

int temp = one;

one = two;

two = temp;

System.out.println("One: " + one + " Two: " + two);

}

public class Lab05{

public static void main(String [] args){

MyRectangle R1, R2, R3;

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

// In Java, when Objects are printed (as shown below), the toString() method is implicitly called. Thus the statements below will call toString() for each of the three MyRectangle objects

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

int x1 = 120, y1 = 70; R1.testInside(x1, y1);

testInside(R1, x2, y2); R1.setSize(120, 240);

**OOP: share variable**

**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

**inheritance**

-properties of a data type can be passed down to a sub-type

**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

public class pracCode2

public static void main(String [] args)

TVClass myTV = new TVClass("Sony");

myTV.setResolution(TVClass.HD);

myTV.setSize(40);

myTV.type = "LCD";

Note that instance variable type is public while the others are private.The convention is to make instance variables private. However, in the main program the "type" instance variable is accessed directly. In orderto allow this, the variable must be declared to be public.

public class TVClass

{ private String brand;

private String resolution;

private int size;

public String type;

public TVClass(String br)

{

brand = new String(br);

}

public **void** setResolution(String res)

{

resolution = res;

}

public String toString()

{

StringBuilder b = new StringBuilder();

b.append("Brand: " + brand + "\n");

b.append("Size: " + size + "\n");

b.append("Type: " + type + "\n");

b.append("Resolution: " + resolution + "\n");

return b.toString();

} }

四舍五入 double value = 23423.455;

double result = (int)(value \* 100 + 0.5) / 100.0; 23423.46

public static int getInteger(Scanner s, int lowerBound)

import java.util.\*;

public class review

{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

int num1 = getInteger(s, 0);

int num2 = getInteger(s, num1);

System.out.println("your numbers are " + num1 + " and " + num2);

}

public static int getInteger(Scanner s, int lowerBound)

{

System.out.println("Enter an integer greater than" + lowerBound);

int number = s.nextInt();

while(number < lowerBound)

{

System.out.println("Enter an integer greater than" + lowerBound);

number = s.nextInt();

s.nextLine();

}

return number;

}

}

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

String first = info[0];