CLASSES AND OBJECTS

- Class defines the shape and behavior of an object
- Any concept we wish to represent is encapsulated in a class.
- Java class includes instance variables and methods.
- Class is a template for an object.
- A Java program consists of one or more classes

- A class is an abstract description of objects

```
class Car {
        ...description of a car goes here...
}
```

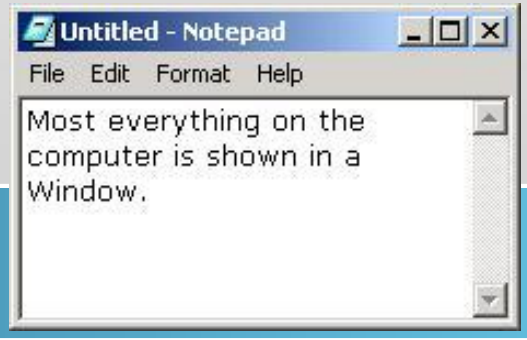
- some objects of class:



- Here is another example of a class:

class Window { ... }

- Some examples of Windows:



Basic Terminology

- A *class* defines a kind of objects:
  - ✓specifies the kinds of *attributes* (*data*) an object can have.
  - ✓provides *methods* specifying the *actions* an object can take.
- An object is an *instance* of the class.
- Person is a class
  - ✓Alice and Bob are objects of the Person class.

What does a class have?

- *Members* of a class*:*
  - ✓*Attributes (instance variables)*

    For each instance of the class (object), values of attributes can vary.
  - ✓*Methods*

- Person class
  - ✓Attributes: name, address, phone number
  - ✓Methods: change address, change phone number
- Alice object
  - Name is Alice, address is …
- Bob object
  - Name is Bob, address is …

```
class classname
{           type instance-variable1;
            // ...
           type instance-variableN;

           type methodname1(parameter-list)
           {        // body of method
           }
           // ...
           type methodnameN(parameter-list)
           {        // body of method
           }
}
```

```
class Box
 {
   double width;
   double height;
   double depth;
}
```

```
class Box
 {
   double width;
   double height;
   double depth;
}
```

```
Box      mybox = new Box();          // create a Box object
```

```
mybox.width = 100;
```

```
class Box
{       double width;
        double height;
        double depth;
}
class BoxDemo
{         public static void main(String args[])
        {       Box mybox = new Box();
                double vol;
                mybox.width = 10;
                mybox.height = 20;
                mybox.depth = 15;
               vol = mybox.width * mybox.height * mybox.depth;
               System.out.println("Volume is " + vol);
  } }
```

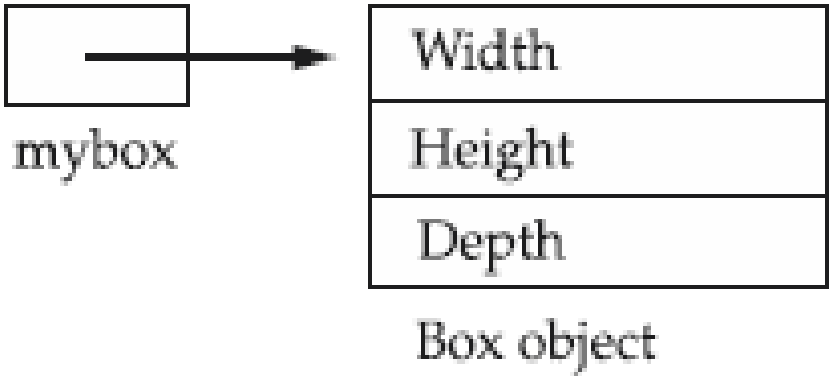
```
class BoxDemo2
{       public static void main(String args[])
        {       Box mybox1 = new Box();
                Box mybox2 = new Box();
                double vol;
                mybox1.width = 10;
                mybox1.height = 20;
                mybox1.depth = 15;
                mybox2.width = 3;
                mybox2.height = 6;
                mybox2.depth = 9;
                vol = mybox1.width * mybox1.height * mybox1.depth;
                System.out.println("Volume is " + vol);
                vol = mybox2.width * mybox2.height * mybox2.depth;
               System.out.println("Volume is " + vol);
  } }
```

Declaring Objects

```
Box mybox = new Box();
```

This statement combines the two steps.

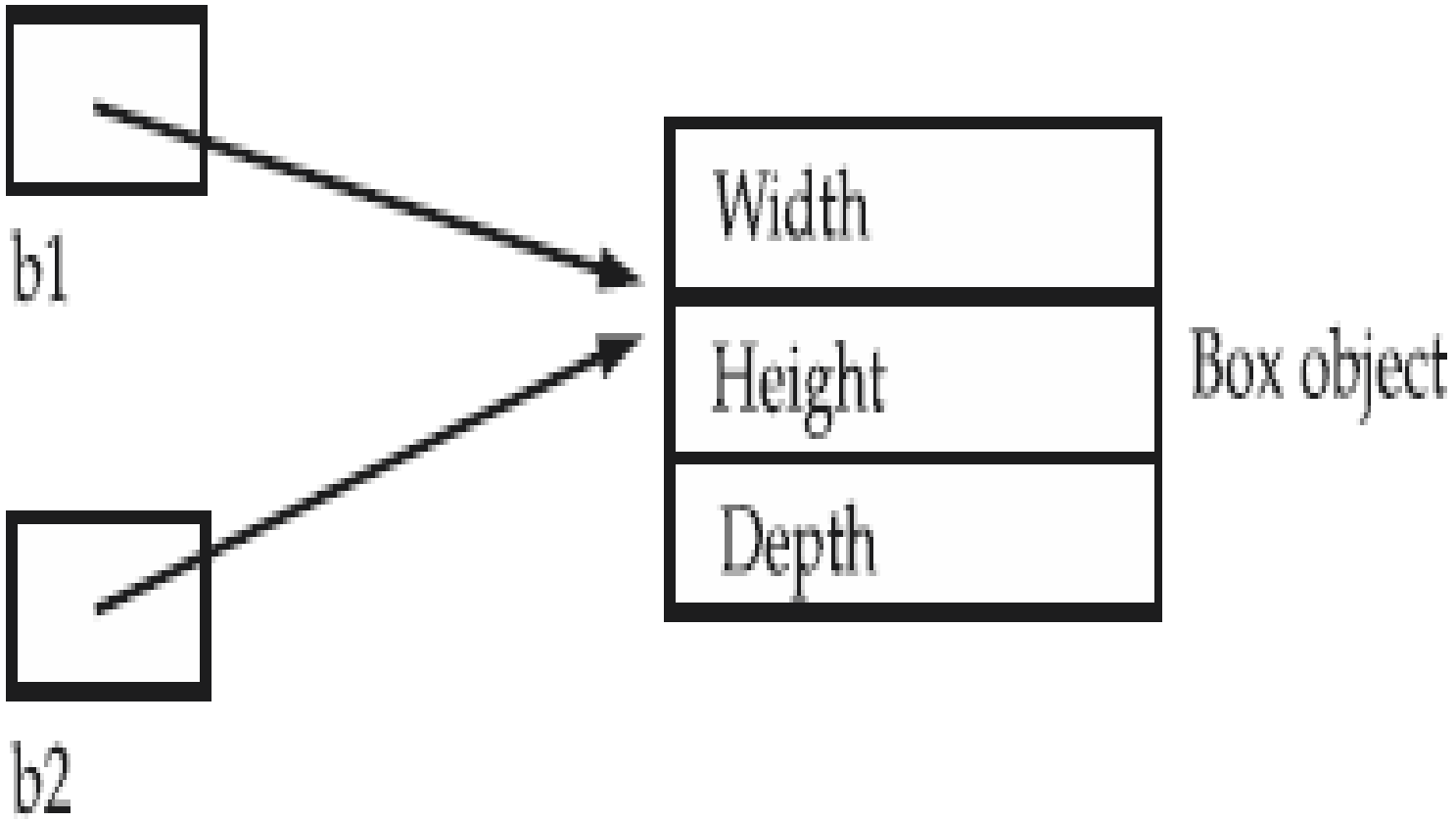
```
Box    mybox;            // declare reference to object
mybox = new Box();    // allocate a Box object
```

Statement	Effect
Box mybox;	
mybox = new Box();	

Declaring an object of type Box

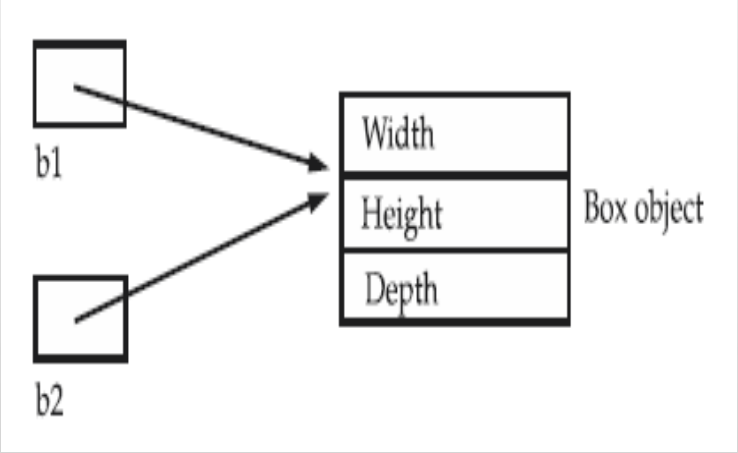
Assigning Object Reference Variables

```
Box b1 = new Box();
Box b2 = b1;
```


b1
b2
Width
Height
Depth
Box object

```
Box b1 = new Box();
Box b2 = b1;
// ...
b1 = null;
```

b1 has been set to null, but b2 still points to the original object.



Introducing Methods

```
type name(parameter-list)
 {
      // body of method
 }
```

Adding a Method to the Box Class

```
class Box
 {
        double width;
        double height;
        double depth;

        void volume()
        {       System.out.print("Volume is ");
                System.out.println(width * height * depth);
        }
}
```

```
class BoxDemo3
{       public static void main(String args[])
    {       Box mybox1 = new Box();
            Box mybox2 = new Box();

            mybox1.width = 10;
            mybox1.height = 20;
            mybox1.depth = 15;

            mybox2.width = 3;
            mybox2.height = 6;
            mybox2.depth = 9;

            mybox1.volume();
            mybox2.volume();
  } }
```

Returning value

```
class Box
{
        double width;
        double height;
        double depth;

        double volume()
        {
                return width * height * depth;
        }
}
```

```
class BoxDemo4
{    public static void main(String args[])
    {       Box mybox1 = new Box();
            Box mybox2 = new Box();
            double vol;
            mybox1.width = 10;    mybox1.height = 20;     mybox1.depth = 15;
            mybox2.width = 3;     mybox2.height = 6;      mybox2.depth = 9;

            vol = mybox1.volume();
            System.out.println("Volume is " + vol);
            vol = mybox2.volume();
            System.out.println("Volume is " + vol);
    } }
```

```
// parameterized method.
class Box
{       double width; double height; double depth;

        double volume()
      {       return width * height * depth;    }

       void setDim(double w, double h, double d)
     {
              width = w;
              height = h;
              depth = d;
     }
}
```

```
class BoxDemo5
{       public static void main(String args[])
        {
                Box mybox1 = new Box();
                Box mybox2 = new Box();
                double vol;
                mybox1.setDim(10, 20, 15);
                mybox2.setDim(3, 6, 9);

                vol = mybox1.volume();
                System.out.println("Volume is " + vol);

                vol = mybox2.volume();
                System.out.println("Volume is " + vol);
        }
```

```
class Rect
{
        int length, width,area;

        void setData(int l,int w)
        {
                length = l;
                width = w;
        }
        int  computeArea()
        {
                area = length * width;
                return area;
        }
        void disp()
        {
                System.out.println("area "+area);
        }

}
```

```
class prg6
{
        public static void main(String args[])
       {
                Rect r1 = new Rect();
                Rect r2 = new Rect();

               r1.setData(5,6);
               r2.setData(8,9);

               int  res1 = r1.computeArea();
               int res2 = r2.computeArea();

                System.out.println("area "+res1);
                System.out.println("area "+res2);
        }
}
```

Constructor

Constructor

- Constructor is a special type of method.
- Constructor has the same name as the class name.
- Constructor cannot return values.
- Constructor is normally used for initializing objects.
- gets invoked “automatically” at the time of object creation.
- A class can have more than one constructor.

```
/* Here, Box uses a constructor to initialize the dimensions of a box. */
class Box
{
   double width;
   double height;
   double depth;
   // This is the constructor for Box.
   Box()
    {          System.out.println("Constructing Box");
               width = 10;
               height = 10;
               depth = 10;
   }
   // compute and return volume
   double volume()
    {
               return width * height * depth;
   }
}
```

```
class BoxDemo6
 {
    public static void main(String args[])
    {
                // declare, allocate, and initialize Box objects
                Box mybox1 = new Box();
                Box mybox2 = new Box();
                double vol;
                // get volume of first box
                vol = mybox1.volume();
                System.out.println("Volume is " + vol);
                // get volume of second box
                vol = mybox2.volume();
                System.out.println("Volume is " + vol);
    }
}
```

Parameterized Constructors

```
/* Here, Box uses a parameterized constructor to initialize the dimensions of a box. */
class Box
{
    double width;
    double height;
    double depth;
    // This is the constructor for Box.
    Box(double w, double h, double d)
    {
            width = w;
            height = h;
            depth = d;
    }
    // compute and return volume
    double volume()
    {
            return width * height * depth;
    }
}
```

```
class BoxDemo7
{
    public static void main(String args[])
    {
                // declare, allocate, and initialize Box objects
                Box mybox1 = new Box(10, 20, 15);
                Box mybox2 = new Box(3, 6, 9);
                double vol;
                // get volume of first box
                vol = mybox1.volume();
                System.out.println("Volume is " + vol);
                // get volume of second box
                vol = mybox2.volume();
                System.out.println("Volume is " + vol);
    }
}
```

this keyword

```
// A redundant use of this.
Box(double w, double h, double d)
 {
   this.width = w;
   this.height = h;
   this.depth = d;
}
```

instance variable hiding

```
// Use this to resolve name-space collisions.
Box(double width, double height, double depth)
 {
   this.width = width;
   this.height = height;
   this.depth = depth;
}
```

A stack class

```
// This class defines an integer stack that can hold 10 values.
class Stack
 {
        int stck[] = new int[10];
        int tos;
        // Initial I ze top-of- stack
        Stack()
         {       tos = -1;
        }
        // Push an item onto the stack
        void push(int item)
         {
                if(tos==9)
                System.out.println("Stack is full.");
                else
                stck[++tos] = item;
        }
```

```
// Pop an item from the stack
int pop()
{
        if(tos < 0)
         {
                System.out.println("Stack underflow.");
                return 0;
        }
        else
        return stck[tos--];
        }
}
```

```
class TestStack
 {
    public static void main(String args[])
     {
                Stack mystack1 = new Stack();
                Stack mystack2 = new Stack();
                // push some numbers onto the stack
                for(int i=0; i<10; i++)    mystack1.push( i );
                for(int i=10; i<20; i++) mystack2.push( i );
                // pop those numbers off the stack
                System.out.println("Stack in mystack1:");
                for(int i=0; i<10; i++)
                System.out.println(mystack1.pop());
                System.out.println("Stack in mystack2:");
                for(int i=0; i<10; i++)
                System.out.println(mystack2.pop());
    }
}
```

Stack in mystack1:

9

8

7

6

5

4

3

2

1

0

Stack in mystack2:

19

18

17

16

15

14

13

12

11

10

OVERLOADING METHODS

```
class OverloadDemo
 {
    void test()
     {          System.out.println("No parameters");
    }
    // Overload test for one integer parameter.
    void test(int a)
    {          System.out.println("a: " + a);
    }
    // Overload test for two integer parameters.
    void test(int a, int b)
     {          System.out.println("a and b: " + a + " " + b);
    }
    // overload test for a double parameter
    double test(double a)
     {          System.out.println("double a: " + a);
                return a*a;
    }
}
```

```
class Overload
{
    public static void main(String args[])
    {
                OverloadDemo ob = new OverloadDemo();
                double result;
                // call all versions of test()
                ob.test();
                ob.test(10);
                ob.test(10, 20);
                result = ob.test(123.25);
                System.out.println("Result of ob.test(123.25): " + result);
    }
}
```

```
// Automatic type conversions apply to overloading.
class OverloadDemo
 {
    void test()
    {
                System.out.println("No parameters");
    }
    // Overload test for two integer parameters.
    void test(int a, int b)
    {
                System.out.println("a and b: " + a + " " + b);
    }
    // overload test for a double parameter
    void test(double a)
    {
                System.out.println("Inside test(double) a: " + a);
    }
}
```

```
class Overload
 {
   public static void main(String args[])
    {
            OverloadDemo ob = new OverloadDemo();
            int i = 88;
            ob.test();
            ob.test(10, 20);
            ob.test(i); // this will invoke test(double)
            ob.test(123.2); // this will invoke test(double)
   }
}
```

OVERLOADING CONSTRUCTORS

```
/* Here, Box defines three constructors to initialize the dimensions of a box various ways. */
class Box
{               double width;  double height; double depth;
                Box(double w, double h, double d)
                 {              width = w;   height = h;  depth = d;
                }
                Box()
                 {              width = -1; // use -1 to indicate
                                height = -1; // an uninitialized
                                depth = -1; // box
                }
                // constructor used when cube is created
                Box(double len)
                 {              width = height = depth = len;
                }
                // compute and return volume
                double volume()
                {              return width * height * depth;
                } }
```

```
class OverloadCons
 {
        public static void main(String args[])
        {       // create boxes using the various constructors
                Box mybox1 = new Box(10, 20, 15);
                Box mybox2 = new Box();
                Box mycube = new Box(7);
                double vol;
                // get volume of first box
                vol = mybox1.volume();
                System.out.println("Volume of mybox1 is " + vol);
                // get volume of second box
                vol = mybox2.volume();
                System.out.println("Volume of mybox2 is " + vol);
                // get volume of cube
                vol = mycube.volume();
                System.out.println("Volume of mycube is " + vol);
        }
}
```

Using object as a parameter

```
// Objects may be passed to methods.
class Test
{
        int a, b;
        Test(int i, int j)
        {
                a = i;
                b = j;
        }
        // return true if o is equal to the invoking object
        boolean equals(Test o)
        {
                if(o.a == a && o.b == b) return true;
                else return false;
        }
}
```

```
class PassOb
{
    public static void main(String args[])
    {
                Test ob1 = new Test(100, 22);
                Test ob2 = new Test(100, 22);
                Test ob3 = new Test(-1, -1);
                System.out.println("ob1 == ob2: " + ob1.equals(ob2));
                System.out.println("ob1 == ob3: " + ob1.equals(ob3));
    }
}
```

```
// Here, Box allows one object to initialize another.
class Box
 {
        double width;
        double height;
        double depth;
        // construct clone of an object
        Box(Box ob)
        {       // pass object to constructor
                width = ob.width;
                height = ob.height;
                depth = ob.depth;
        }
        Box(double w, double h, double d)
        {       width = w;
                height = h;
                depth = d;
        }
```

```
// constructor used when no dimensions specified
  Box()
  {
        width = -1; // use -1 to indicate
        height = -1; // an uninitialized
        depth = -1; // box
  }
  // constructor used when cube is created
  Box(double len)
  {
        width = height = depth = len;
  }
  // compute and return volume
  double volume()
  {
        return width * height * depth;
  }
}
```

```
class OverloadCons2
{
    public static void main(String args[])
    {
                // create boxes using the various constructors
                Box mybox1 = new Box(10, 20, 15);
                Box mybox2 = new Box();
                Box mycube = new Box(7);
                Box myclone = new Box(mybox1);
                double vol;
                // get volume of first box
                vol = mybox1.volume();
                System.out.println("Volume of mybox1 is " + vol);
      }
}
```

Argument Passing

```
// Simple types are passed by value.
class Test
{     void meth(int i, int j)
     {          i *= 2;
                j /= 2;
      }
}
class CallByValue
{
    public static void main(String args[])
    {
                Test ob = new Test();
                int a = 15, b = 20;
                System.out.println("a and b before call: " + a + " " + b);
                ob.meth(a, b);
                System.out.println("a and b after call: " +a + " " + b);
   } }
```

```
// Objects are passed by reference.
class Test
 {
   int a, b;
   Test(int i, int j)
   {
                a = i;
                b = j;
   }
   // pass an object
   void meth(Test o)
   {
                o.a *= 2;
                o.b /= 2;
   }
}
```

```
class CallByRef
 {
    public static void main(String args[])
     {
                Test ob = new Test(15, 20);
                System.out.println("ob.a and ob.b before call: " +ob.a + " " + ob.b);
                ob.meth(ob);
                System.out.println("ob.a and ob.b after call: " +ob.a + " " + ob.b);
    }
}
```

Returning objects

```
// Returning an object.
class Test
{
    int a;
    Test(int i)
     {
                a = i;
    }
    Test incrByTen()
    {
                Test temp = new Test(a+10);
                return temp;
    }
}
```

```
class RetOb
{
    public static void main(String args[])
    {
                Test ob1 = new Test(2);
                Test ob2;
                ob2 = ob1.incrByTen();
                System.out.println("ob1.a: " + ob1.a);
                System.out.println("ob2.a: " + ob2.a);
                ob2 = ob2.incrByTen();
                System.out.println("ob2.a after second increase: “+ ob2.a);
    }
}
```

ob1.a: 2

ob2.a: 12

ob2.a after second increase: 22

Recursion

```
// A simple example of recursion.
class Factorial
{
    // this is a recursive function
    int fact(int n)
    {
                int result;
                if(n==1) return 1;
                result = fact(n-1) * n;
                return result;
    }
}
```

```
class Recursion
 {
    public static void main(String args[])
    {
                Factorial f = new Factorial();
                System.out.println("Factorial of 3 is " + f.fact(3));
                System.out.println("Factorial of 4 is " + f.fact(4));
                System.out.println("Factorial of 5 is " + f.fact(5));
    }
}
```

```
// Another example that uses recursion.
class RecTest
{
    int values[];
    RecTest(int i)
    {
            values = new int[i];
    }

    // display array -- recursively
    void printArray(int i)
    {
            if(i==0) return;
            else printArray(i-1);
            System.out.println("[" + (i-1) + "] " + values[i-1]);
    }
}
```

```
class Recursion2
 {
    public static void main(String args[])
    {
                RecTest ob = new RecTest(10);
                int i;
                for(i=0; i<10; i++)
                 ob.values[i] = i;

                ob.printArray(10);
    }
}
```

[0] 0
[1] 1
[2] 2
[3] 3
[4] 4
[5] 5
[6] 6
[7] 7
[8] 8
[9] 9