Pemrograman Berorientasi Objek
Class 2

Adam Mukharil Bachtiar

Teknik Informatika UNIKOM

Konsep Dasar PBO

1. Deklarasi Objek
2. Penggambaran Objek
3. Kata kunci this
4. Kata kunci statis
5. Overloading Method
6. Array of Object
7. Parameter

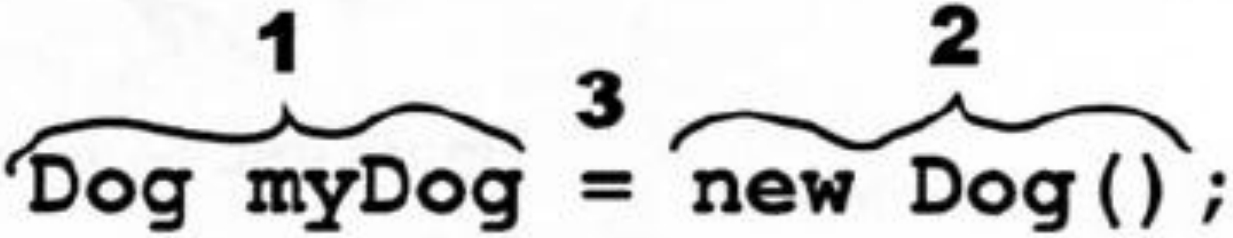
Deklarasi Objek C++

```
<nama_class> <nama_objek>;
```

MAHASISWA MHS1;

Deklarasi Objek Java

The 3 steps of object declaration, creation and assignment



Deklarasi Objek Java

1 Declare a reference variable

```
Dog myDog = new Dog();
```

Tells the JVM to allocate space for a reference variable, and names that variable *myDog*. The reference variable is, forever, of type Dog. In other words, a remote control that has buttons to control a Dog, but not a Cat or a Button or a Socket.

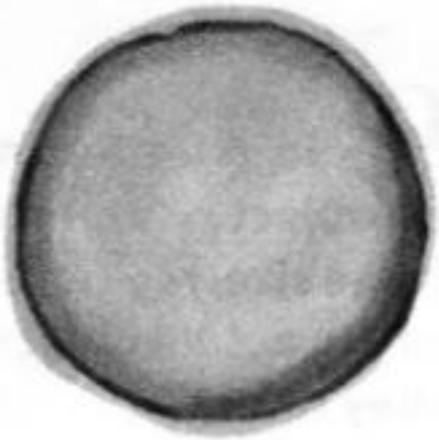
Dog

Deklarasi Objek Java

2 Create an object

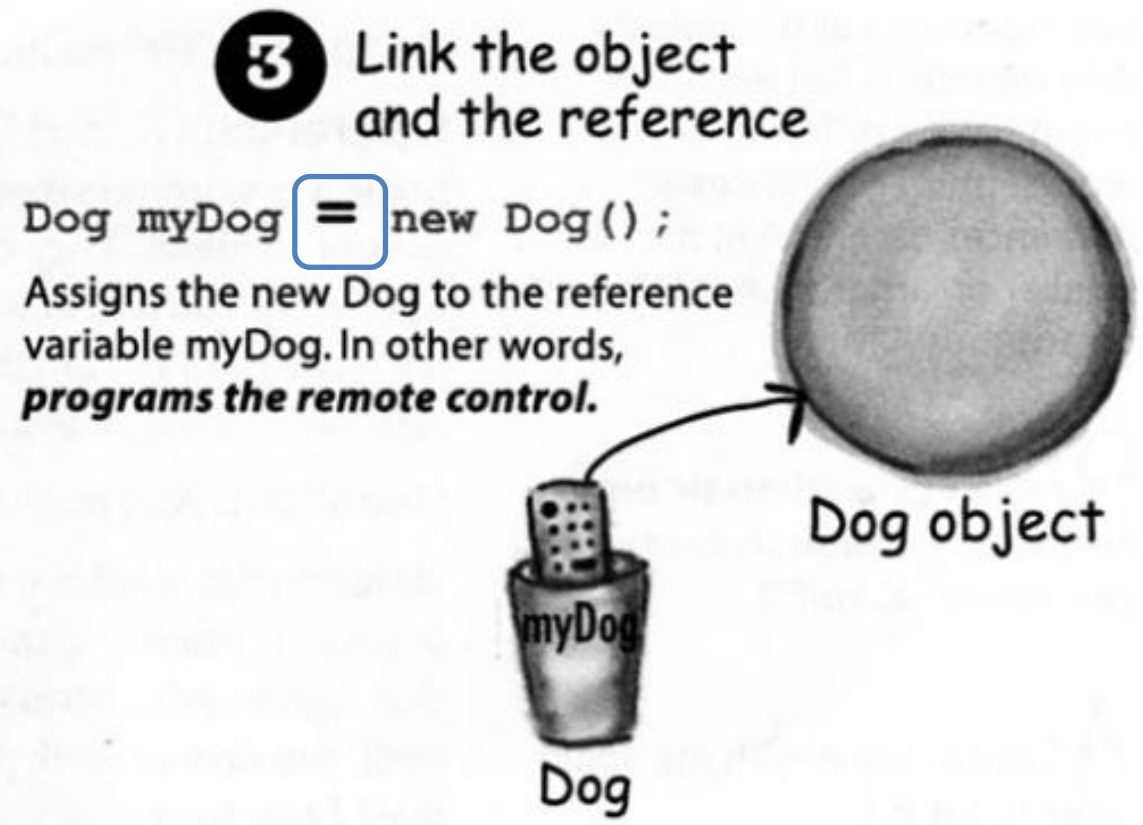
```
Dog myDog = new Dog();
```

Tells the JVM to allocate space for a new Dog object on the heap (we'll learn a lot more about that process, especially in chapter 9.)



Dog object

Deklarasi Objek Java



Penggambaran Objek

Kata Kunci This

```
Public void setUmur(int umur)
{
      umur = umur;
}
```

Is there something

Wrong???

Kata Kunci This

1. KATA KUNCI THIS DIBERIKAN PADA DEKLARASI INTANCE VARIABEL

2. PERINTAH INI AKAN MEREFERENSIKAN NILAI KEPADA INSTANCE VARIABLE DARI CLASS TEMPAT METHOD ITU BERADA

Kata Kunci This

FORMAT PLEASE!!!!

`this.nama_variabel`

Kata Kunci Statis

```
Public int setUmur(int umur)
{
      this.umur=umur;
}
```

Kata Kunci Statis

PERINTAH YANG DIGUNAKAN APABILA INGIN MENGGUNAKAN ATRIBUT ATAU METHOD YANG ADA DI SUATU CLASS TANPA MENGINSTANSIASI OBJEK YANG BERASAL DARI CLASS TERSEBUT.

Kata Kunci Statis

```
class Mahasiswa{
        static int jumlah;
        String nama;
        String nilai;
        public void setNama(String nama){
                this.nama=nama;
        }
        public static int getJumlah(){
                return jumlah;
        }
}
```

TAKE A LOOK AT THIS !!!

Kata Kunci Statis

WHERE'S THE OBJECT ?

```
Public class StaticTes{

  public static void main(String args[]){

    //cara pemanggilan static

    System.out.println("Jumlah: "+Mahasiswa.getJumlah());


    //cara pemanggilan non static

    Mahasiswa mhs1=new mahasiswa();

    mhs1.setNama("Adam");

}
```


Next : Overloading Function

Overloading Function

MEMBUAT FUNGSI DENGAN NAMA YANG SAMA TETAPI MEMILIKI PARAMETER YANG BERBEDA, BAIK SECARA JUMLAH MAUPUN PENAMAAN PARAMETERNYA.

Overloading Function

```
class Tes{
        private:
                char c;

        public:
                void cetak(){
                        cout<<"Hallo!";
                }

                void cetak( int x ){
                        int i;
                        for(i=0;i<n;i++)
                                cout<<"Hallo!"<<endl;
                }
};
```

IT MAKES DIFFERENCE!!!

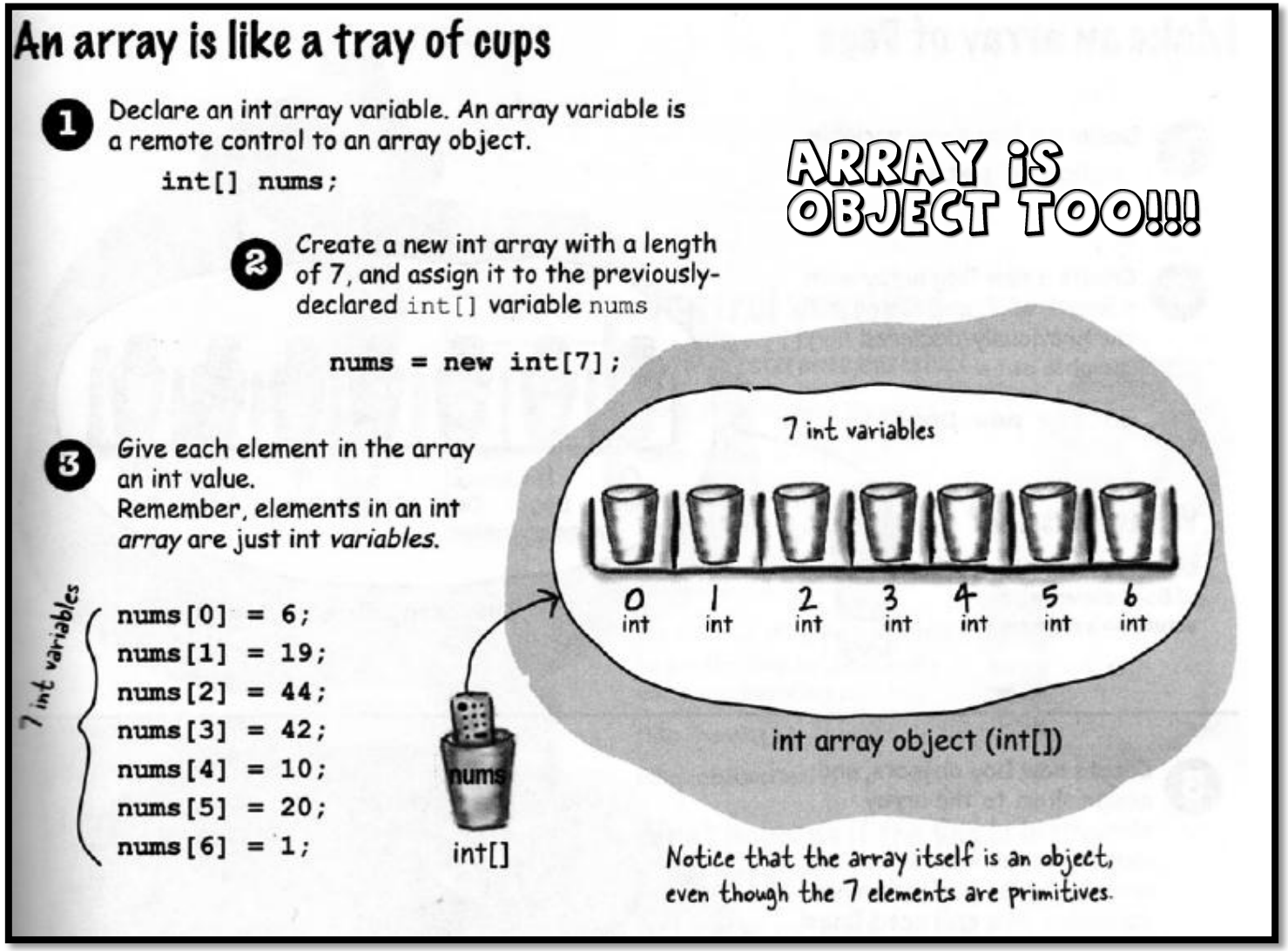
Overloading Function

```
Main()
{
        Tes a;
        a.cetak();
        a.cetak(5);
}
```

Keterangan:

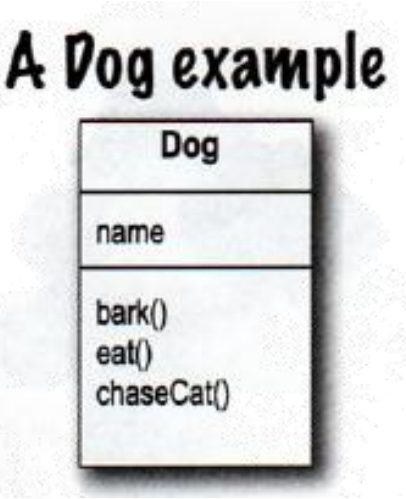
1. a.cetak(); akan menampilkan kata Hallo pada layar sebanyak 1 kali.
2. a.cetak(5); akan menampilkan kata Hallo pada layar sebanyak 5 kali.
3. Penamaan void yang ada pada kelas Tes sama tetapi dibedakan oleh jumlah parameter.

Array


An array is like a tray of cups
1
Declare an int array variable. An array variable is a remote control to an array object.
int[] nums;
ARRAY IS OBJECT TOO!!!
2
Create a new int array with a length of 7, and assign it to the previously-declared int[] variable nums
nums = new int[7];
3
Give each element in the array an int value.
Remember, elements in an int array are just int variables.
7 int variables
nums[0] = 6;
nums[1] = 19;
nums[2] = 44;
nums[3] = 42;
nums[4] = 10;
nums[5] = 20;
nums[6] = 1;
nums
int[]
7 int variables
0 int
1 int
2 int
3 int
4 int
5 int
6 int
int array object (int[])
Notice that the array itself is an object, even though the 7 elements are primitives.

HOW ABOUT ARRAY OF OBJECT???

Array of Object



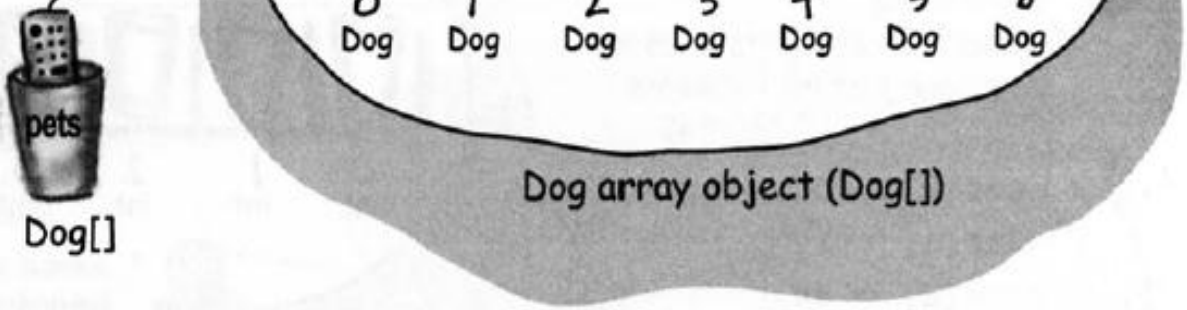
Make an array of Dogs

1. Declare a Dog array variable

```
Dog[] pets;
```

2. Create a new Dog array with a length of 7, and assign it to the previously-declared `Dog[]` variable `pets`

```
pets = new Dog[7];
```



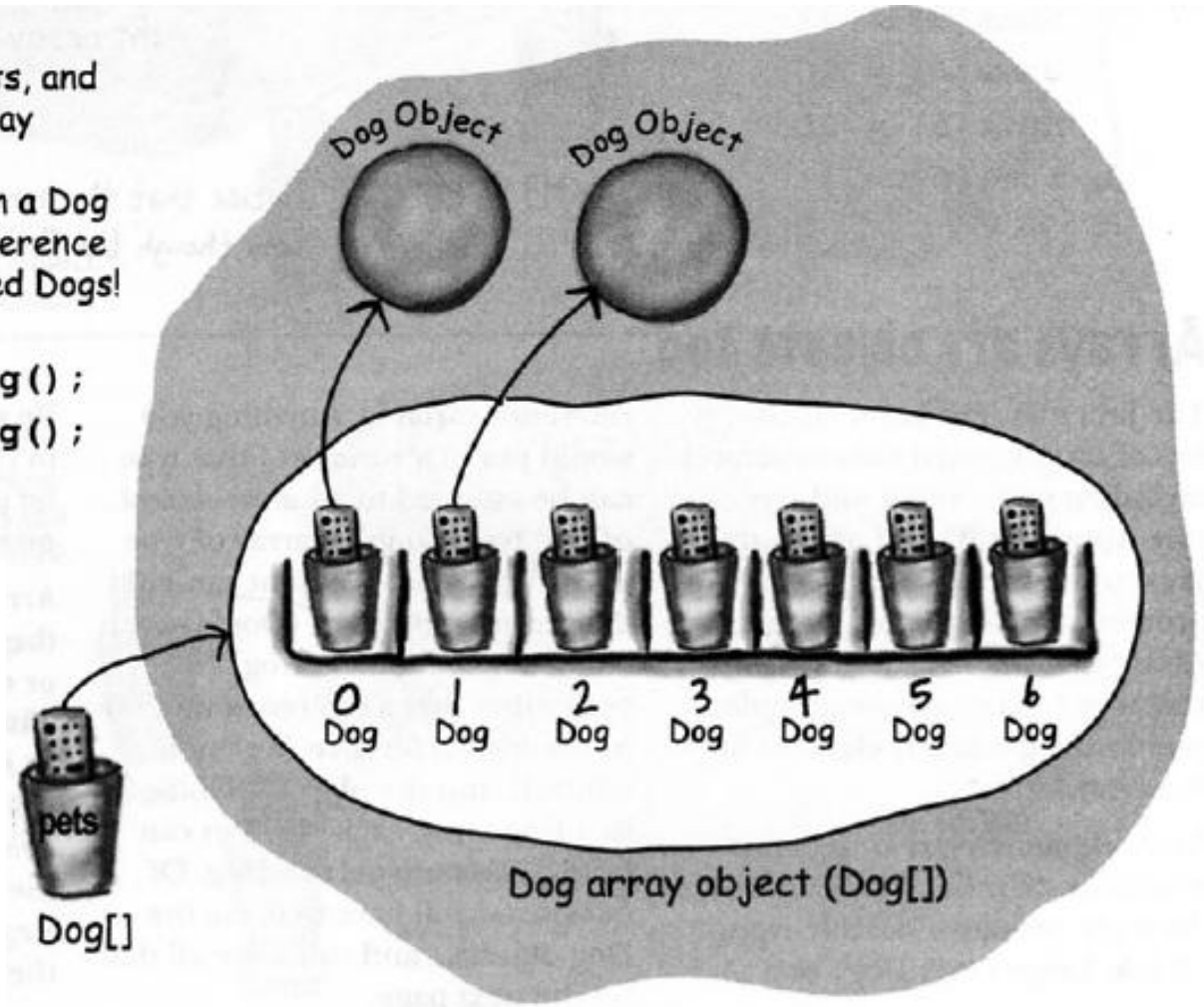
What's missing?

Dogs! We have an array of Dog *references*, but no actual Dog *objects*!

Array of Object

3 Create new Dog objects, and assign them to the array elements.
Remember, elements in a Dog *array* are just Dog reference *variables*. We still need Dogs!

```
pets[0] = new Dog();
pets[1] = new Dog();
```



Array of Object

What happens if the Dog is in a Dog array?

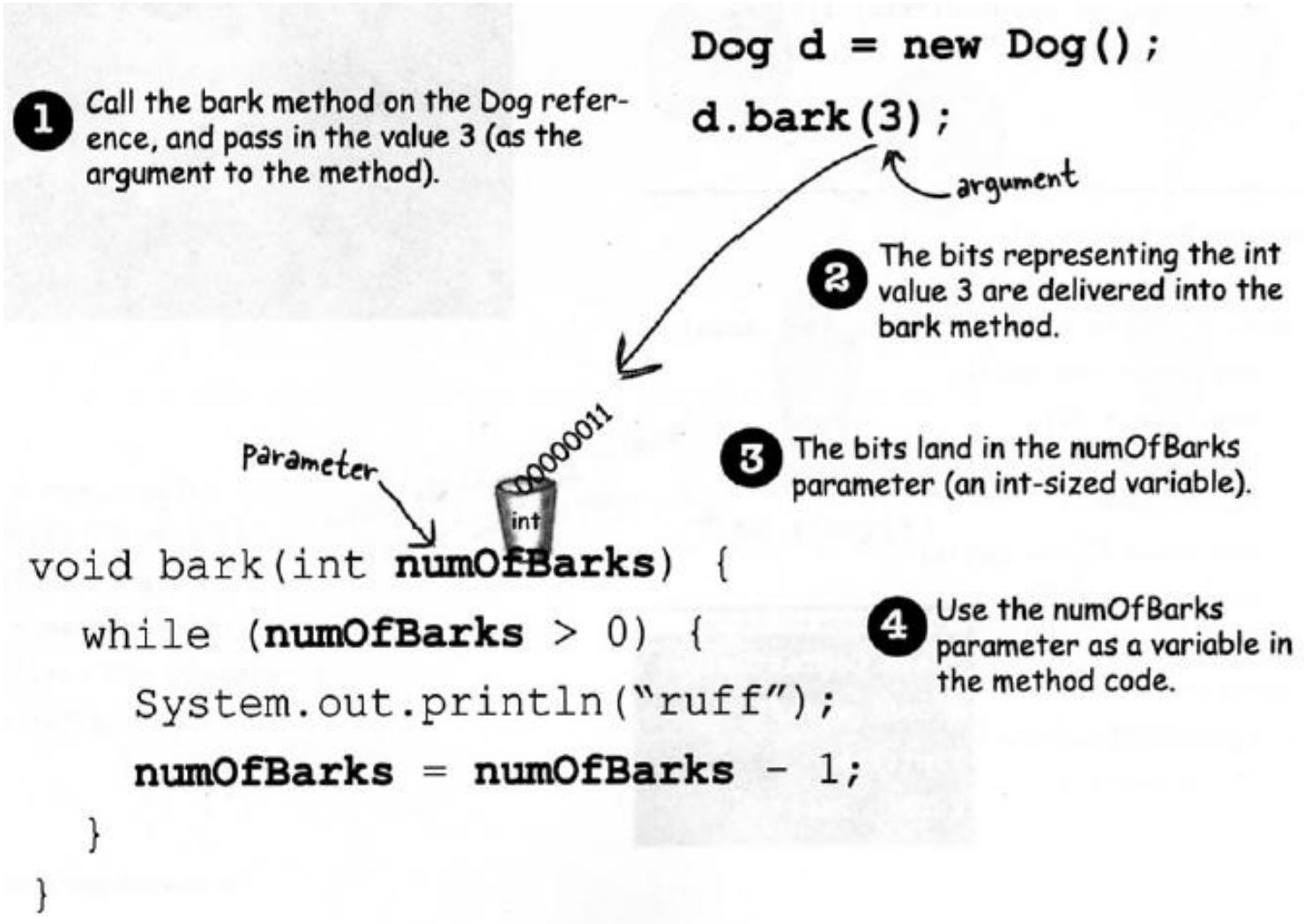
We know we can access the Dog's instance variables and methods using the dot operator, but *on what?*

When the Dog is in an array, we don't have an actual variable name (like *fido*). Instead we use array notation and push the remote control button (dot operator) on an object at a particular index (position) in the array:

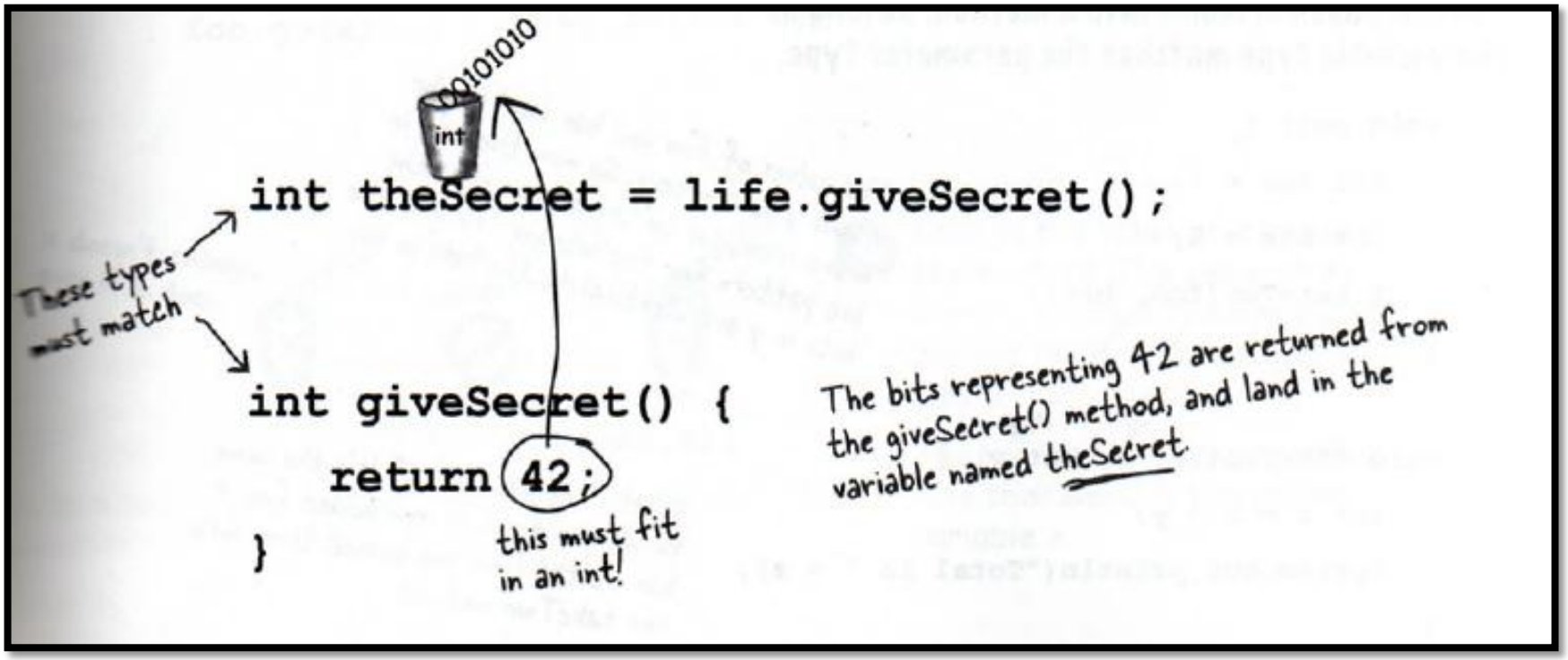
```
Dog[] myDogs = new Dog[3];
myDogs[0] = new Dog();
myDogs[0].name = "Fido";
myDogs[0].bark();
```

ENCAPSULATION IS NOT IMPLEMENTED HERE !!!

Parameter


Dog d = new Dog();
d.bark(3);
argument
1 Call the bark method on the Dog reference, and pass in the value 3 (as the argument to the method).
2 The bits representing the int value 3 are delivered into the bark method.
3 The bits land in the numOfBarks parameter (an int-sized variable).
Parameter
00000011
int
void bark(int numOfBarks) {
while (numOfBarks > 0) {
System.out.println("ruff");
numOfBarks = numOfBarks - 1;
}
}
4 Use the numOfBarks parameter as a variable in the method code.

Parameter

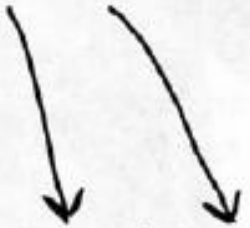

00101010
int
int theSecret = life.giveSecret();
These types must match
int giveSecret() {
return 42;
}
this must fit in an int!
The bits representing 42 are returned from the giveSecret() method, and land in the variable named theSecret.

Parameter

Calling a two-parameter method, and sending it two arguments.

```
void go() {
   TestStuff t = new TestStuff();
   t.takeTwo(12, 34);
}

void takeTwo(int x, int y) {
   int z = x + y;
   System.out.println("Total is " + z);
}
```



The arguments you pass land in the same order you passed them. First argument lands in the first parameter, second argument in the second parameter, and so on.

Parameter

Java is pass-by-value.

That means pass-by-copy.

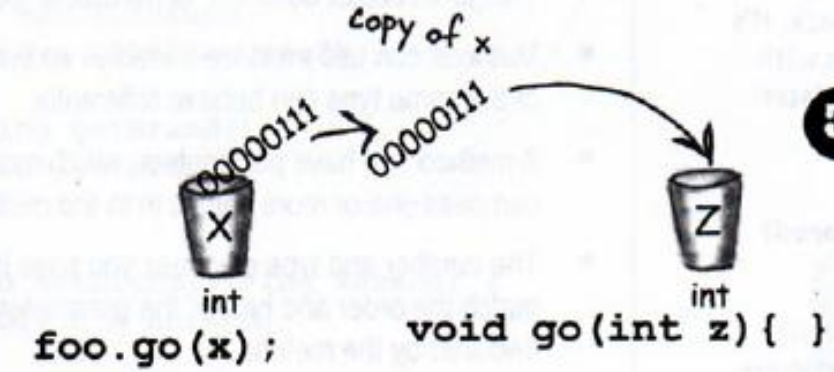
```
int x = 7;
```

1. Declare an int variable and assign it the value '7'. The bit pattern for 7 goes into the variable named x.

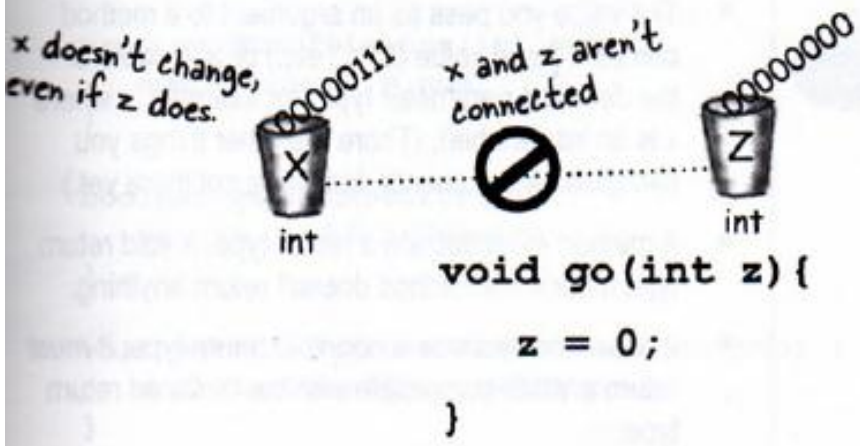
```
void go(int z){ }
```

2. Declare a method with an int parameter named z.

Parameter



3 Call the go() method, passing the variable x as the argument. The bits in x are copied, and the copy lands in z.



4 Change the value of z inside the method. The value of x doesn't change! The argument passed to the z parameter was only a copy of x.

The method can't change the bits that were in the calling variable x.

BRAIN STORMING

Be the Compiler 1

A

```
class Books {
  String title;
  String author;
}

class BooksTestDrive {
  public static void main(String [] args) {

    Books [] myBooks = new Books[3];
    int x = 0;
    myBooks[0].title = "The Grapes of Java";
    myBooks[1].title = "The Java Gatsby";
    myBooks[2].title = "The Java Cookbook";
    myBooks[0].author = "bob";
    myBooks[1].author = "sue";
    myBooks[2].author = "ian";

    while (x < 3) {
      System.out.print(myBooks[x].title);
      System.out.print(" by ");
      System.out.println(myBooks[x].author);
      x = x + 1;
    }
  }
}
```

B

```
class Hobbits {

  String name;

  public static void main(String [] args) {

    Hobbits [] h = new Hobbits[3];
    int z = 0;

    while (z < 4) {
      z = z + 1;
      h[z] = new Hobbits();
      h[z].name = "bilbo";
      if (z == 1) {
        h[z].name = "frodo";
      }
      if (z == 2) {
        h[z].name = "sam";
      }
      System.out.print(h[z].name + " is a ");
      System.out.println("good Hobbit name");
    }
  }
}
```

Be the Compiler 2

A

```
class XCopy {

  public static void main(String [] args) {

    int orig = 42;

    XCopy x = new XCopy();

    int y = x.go(orig);

    System.out.println(orig + " " + y);
  }

  int go(int arg) {

    arg = arg * 2;

    return arg;
  }
}
```

B

```
class Clock {
  String time;

  void setTime(String t) {
    time = t;
  }

  void getTime() {
    return time;
  }
}

class ClockTestDrive {
  public static void main(String [] args) {

    Clock c = new Clock();

    c.setTime("1245");
    String tod = c.getTime();
    System.out.println("time: " + tod);
  }
}
```

FINISH!!!