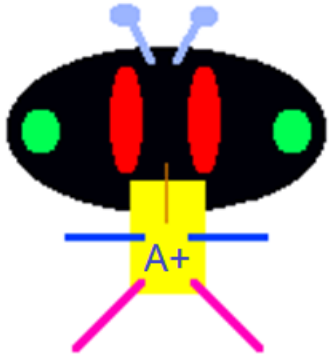
A+ Computer Science

Array of references

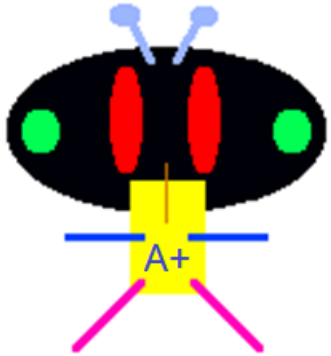
Objects and References

Monster




```
public class Monster
{
    // instance variables
    public Monster(){ code }
    public Monster( int ht ) { code }
    public Monster(int ht, int wt)
    { code }
    public Monster(int ht, int wt, int age)
    {  code  }
}
```

Monster



public Monster()

public Monster(int ht)

public Monster(int ht, int wt)

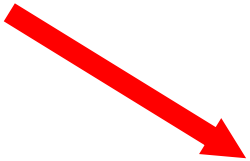
public Monster(int ht, int wt, int age)

The Monster constructor has been overloaded.

Monster Instantiation

Monster m = new Monster();

m

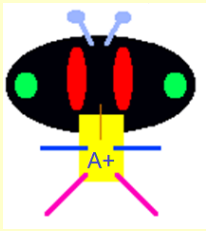


MONSTER

Properties

– height – 0 weight - 0 age - 0

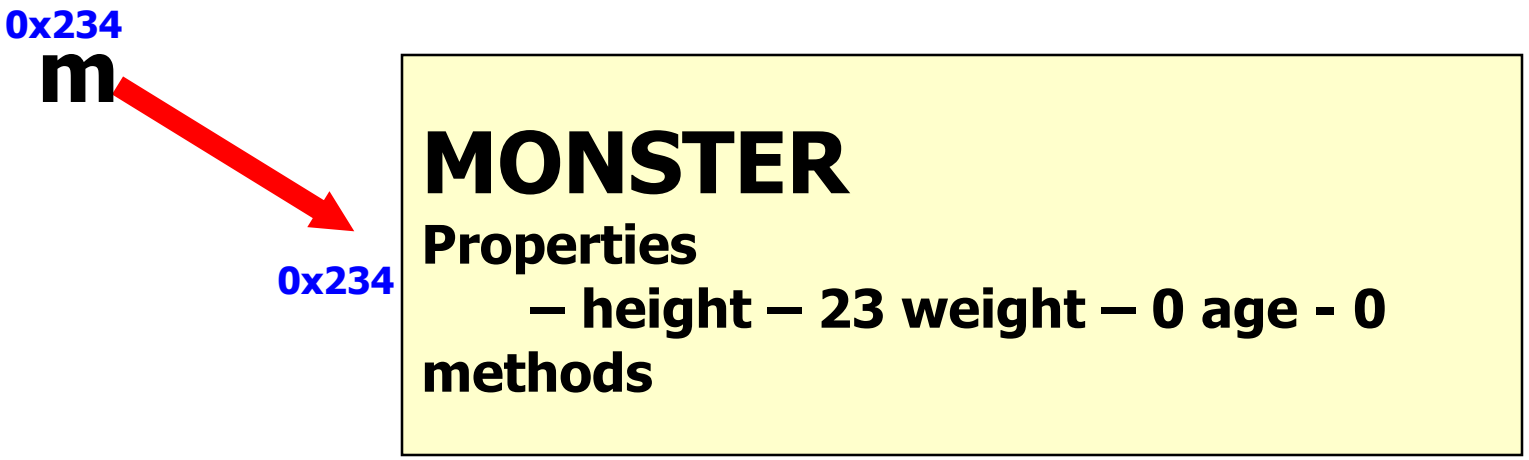
methods



m is a reference variable that refers to a Monster object.

Monster Instantiation

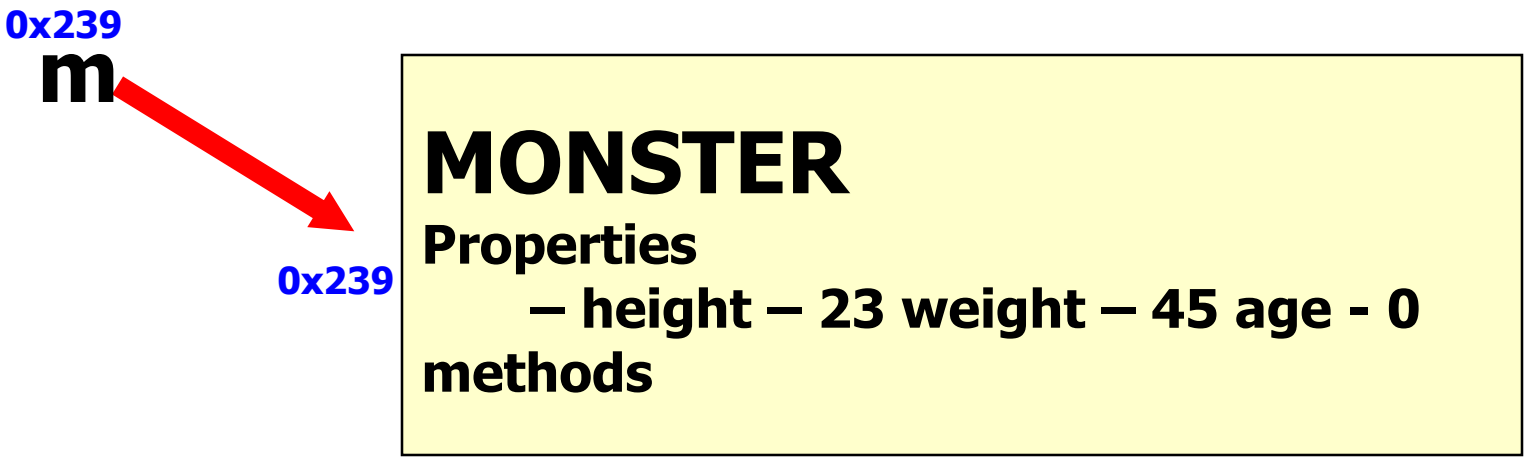
Monster m = new Monster(23);



m is a reference variable that refers to a Monster object.

Monster Instantiation

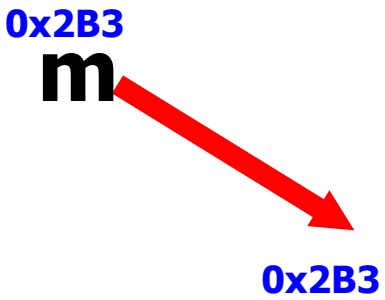
Monster m = new Monster(23, 45);



m is a reference variable that refers to a Monster object.

Monster Instantiation

Monster m = new Monster(23, 45, 11);



MONSTER

Properties

– height – 23 weight – 45 age - 11

methods

m is a reference variable that refers to a Monster object.

References

References

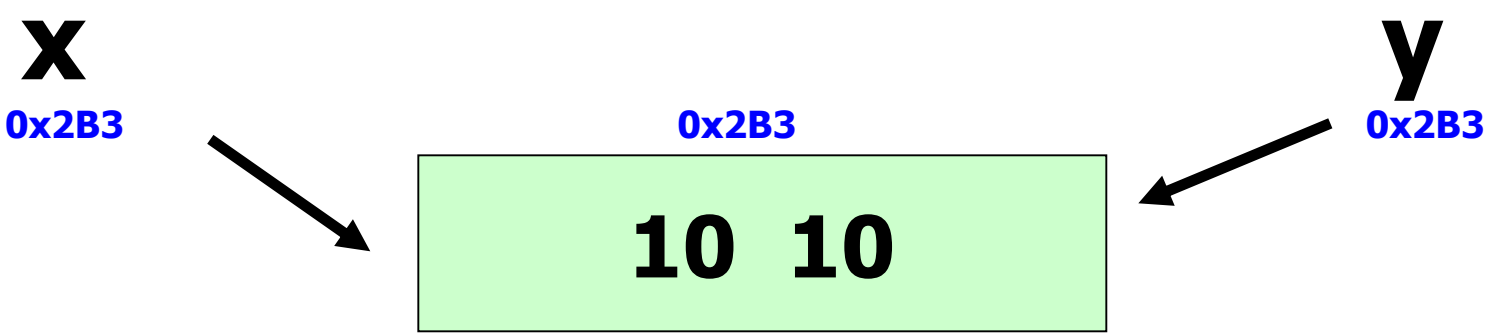
In Java, any variable that refers to an Object is a reference variable.

The variable stores the memory address of the actual Object.

References

```
Monster x = new Monster( 10, 10 );
Monster y = x;
```

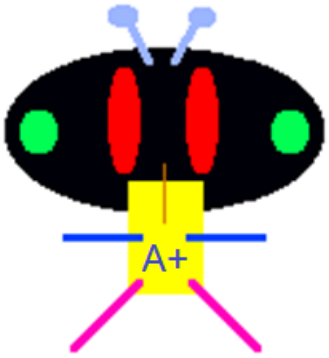
x and y store the same memory address.



References

```
Monster x = new Monster( 10, 10 );
Monster y = x;

System.out.println( x == y );
System.out.println( x.equals( y )  );
```



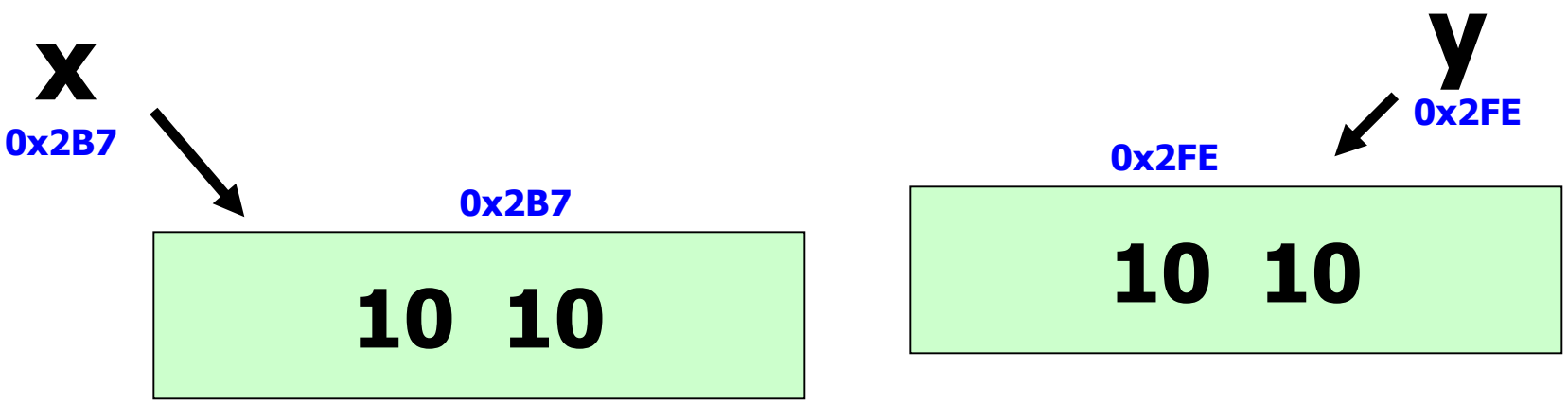
OUTPUT

true
true

References

```
Monster x = new Monster( 10, 10 );
Monster y = new Monster( 10, 10 );
```

x and y store different addresses.



References

```
Monster x = new Monster( 10, 10 );
Monster y = new Monster( 10, 10 );

System.out.println( x == y );
System.out.println( x.equals( y )  );
```

OUTPUT

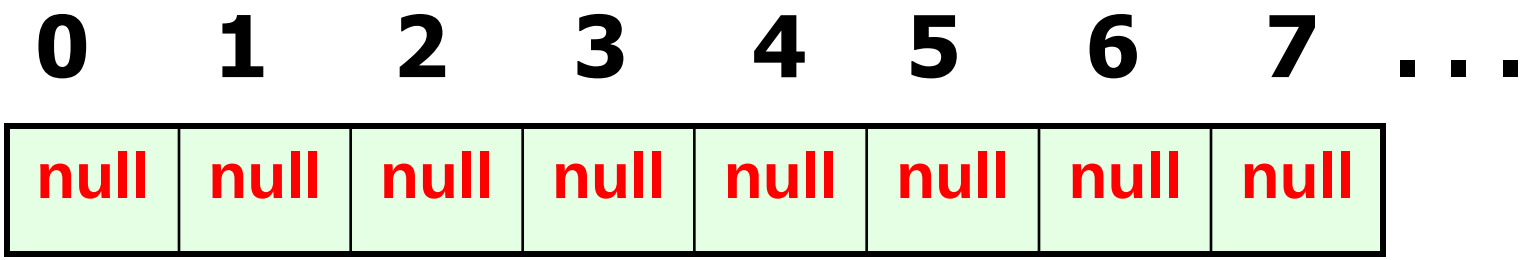
```
false
true
```

references.java

Array Of References

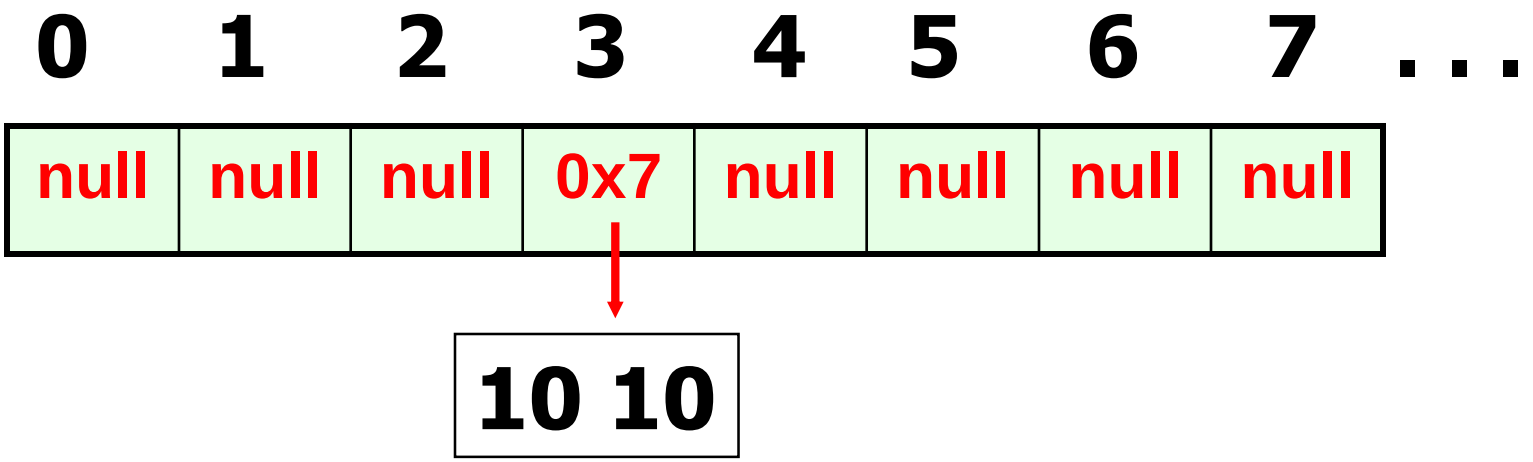
Array of References

```
Monster[] list = new Monster[50];
//all 50 spots are null
```



Array of References

list[3] = new Monster(10, 10);



Array of References

```
Monster[] list = new Monster[5];

out.println(list[0]);
out.println(list[1]);
out.println(list[2]);
out.println(list[3]);
out.println(list[4]);
```

OUTPUT

```
null
null
null
null
null
```

Array of References

```
Monster[] list = new Monster[5];
list[0] = new Monster();
list[1] = new Monster(33);
list[2] = new Monster(3,4,5);

out.println(list[0]);
out.println(list[1]);
out.println(list[2]);
out.println(list[3]);
```

OUTPUT

```
0 0 0
33 0 0
3 4 5
null
```

Array of References

```
Monster[]  list = new Monster[3];
list[0]=new Monster(4);
list[1]=new Monster(9);
list[2]=new Monster(1);

out.println( list[0] );
list[0].setSize(7);

out.println(list[0]);
out.println(list[2]);
```

OUTPUT

```
4
7
1
```

Array of References

```
list[0] . setSize(7);
```

0x242

What does this store?

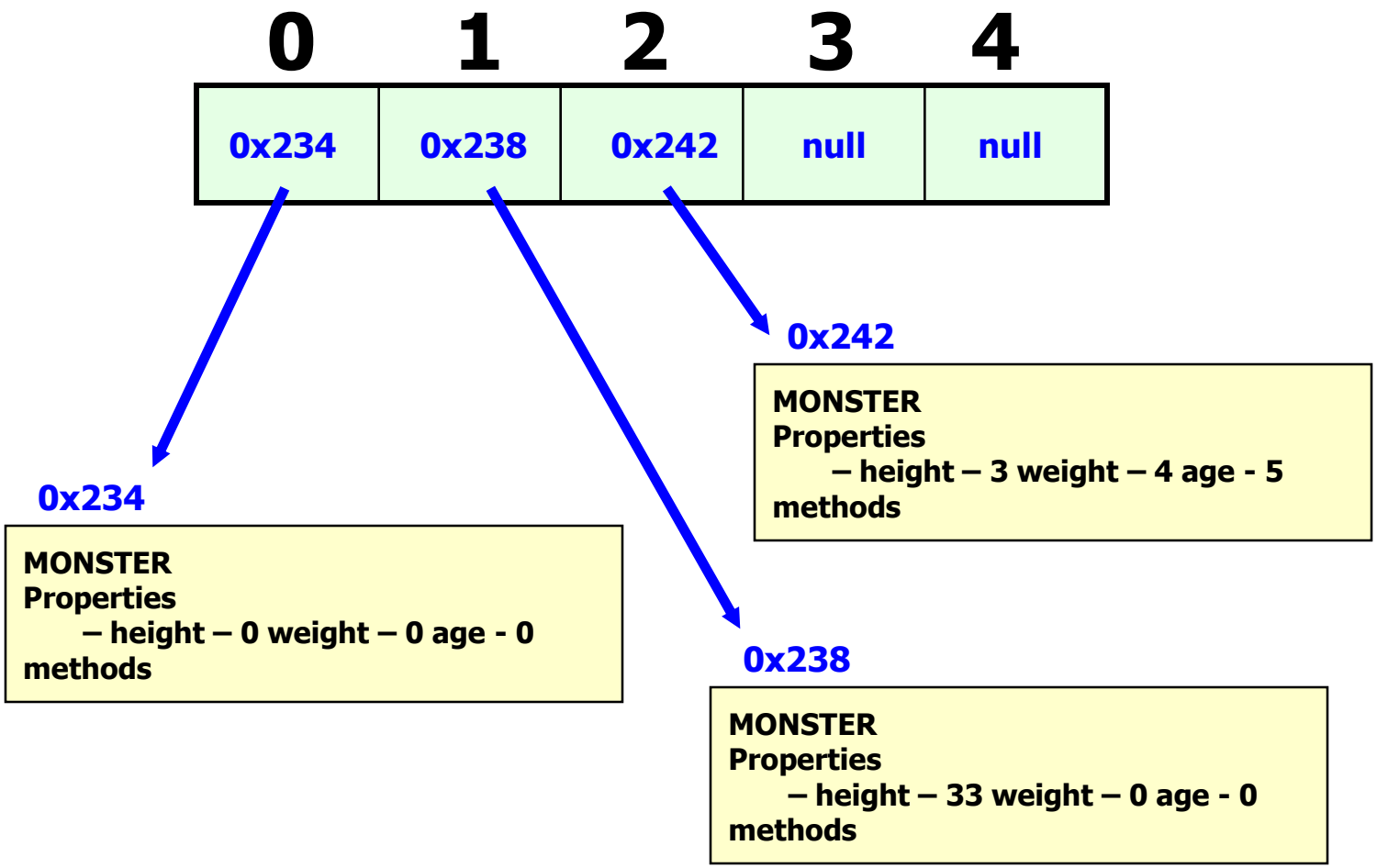
What does the . dot do?

0x242

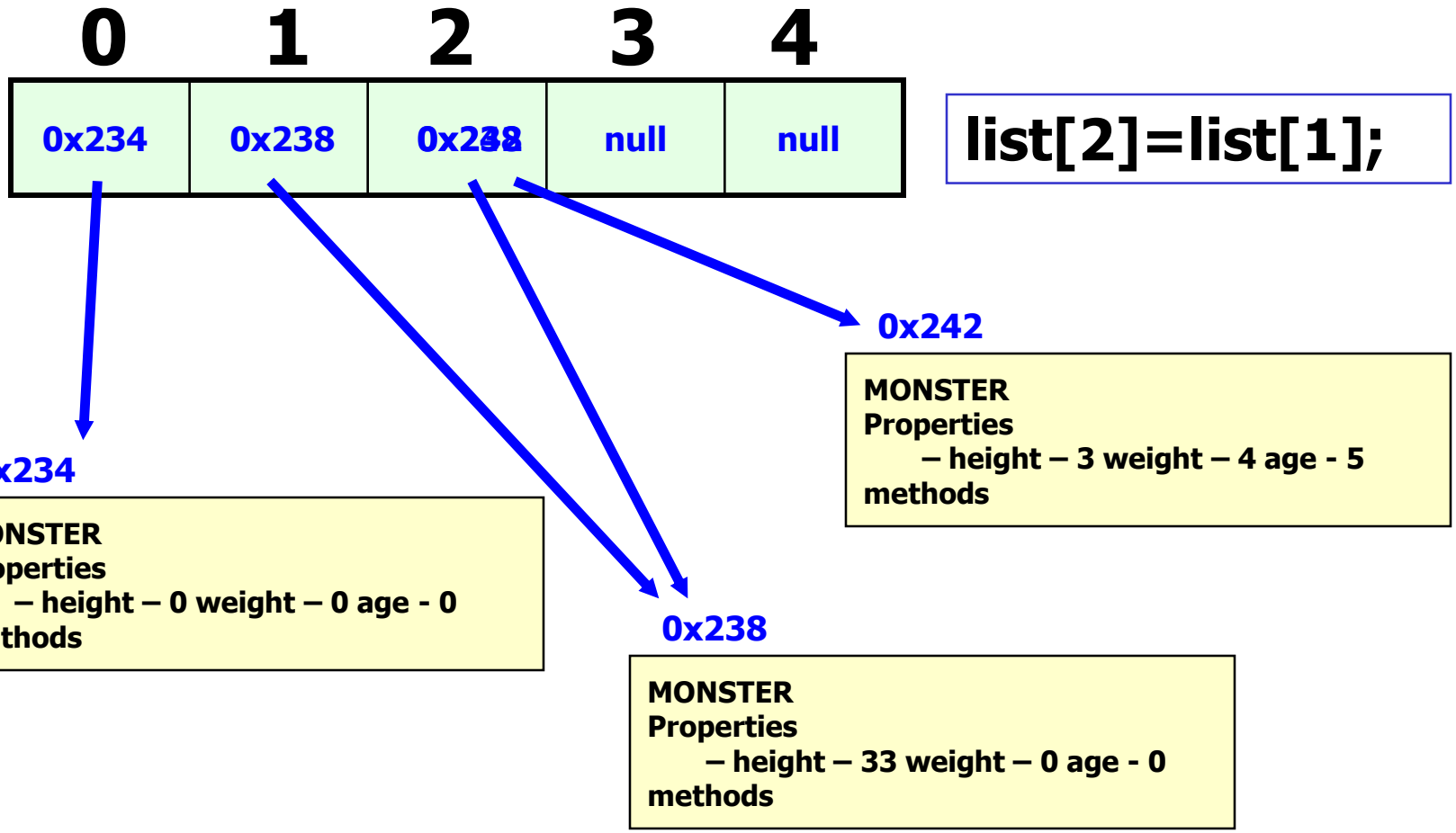
Monster

The . dot grants access to the Object at the stored address.

Array of References



Array of References



arrayofmonsters.java

Array of References

```
public class Herd
{
   private Creature[]  creatureList;

   public Herd()
   {
     //must size the array

   }

   //other constructors and methods
   //not shown
}
```

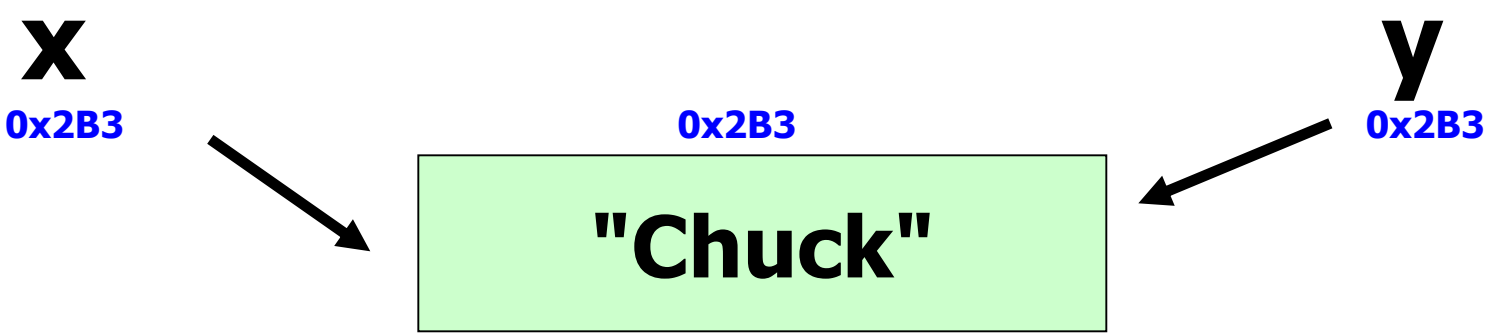
creature.java
herd.java
herdrunner.java

References

References

```
String x = new String("Chuck");
String y = x;
```

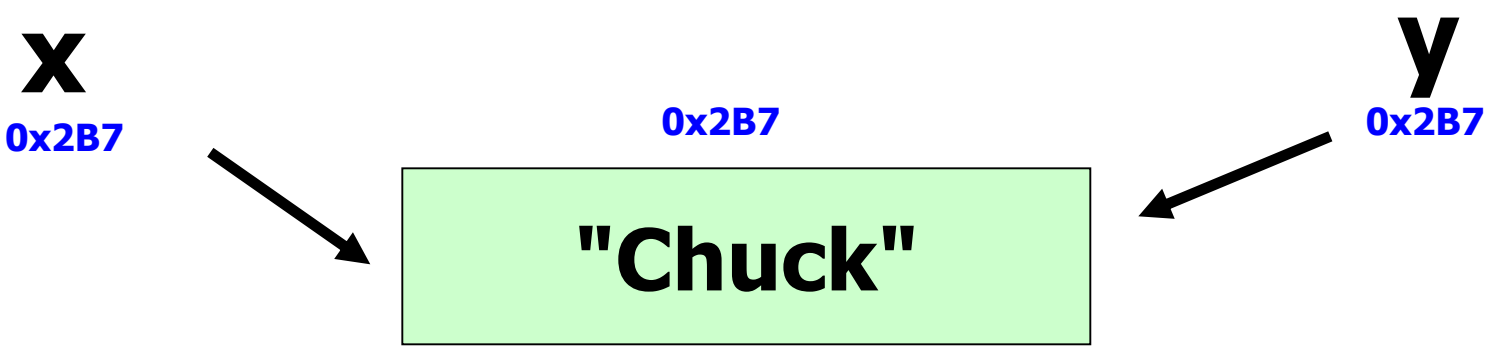
x and y store the same memory address.



References

```
String x = "Chuck";
String y = "Chuck";
```

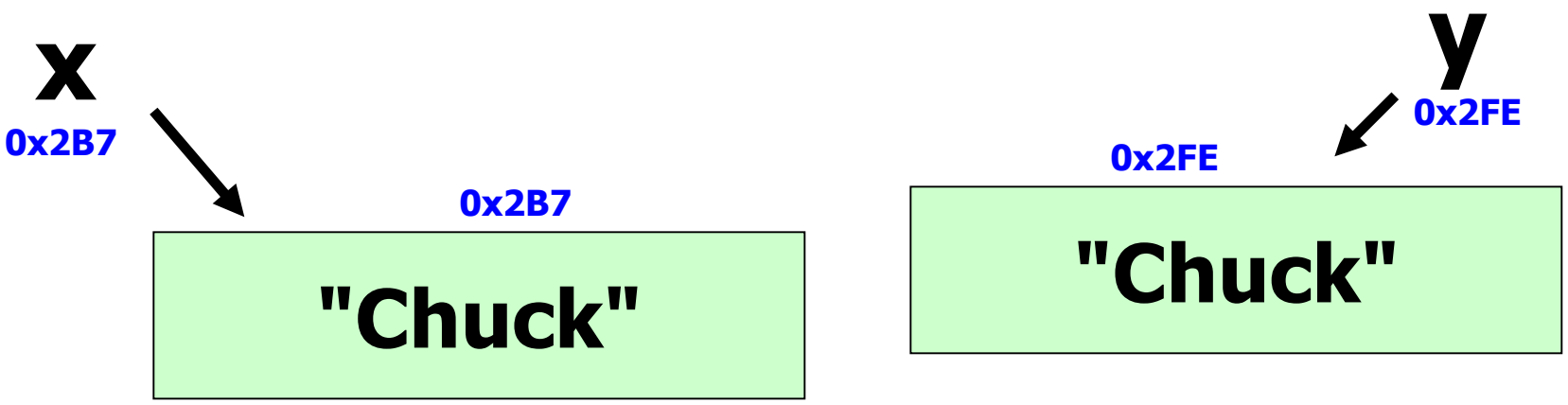
x and y store the same memory address.



References

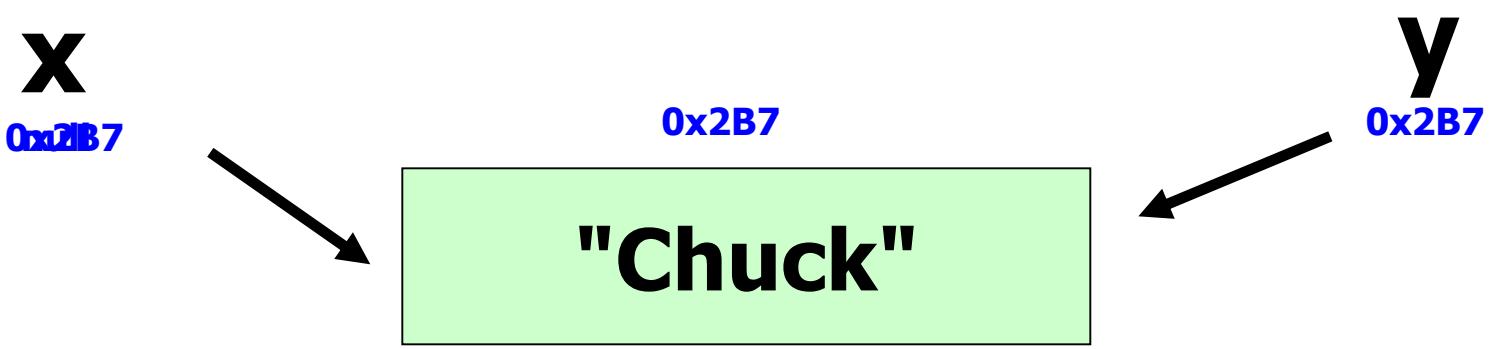
```
String x = new String("Chuck");
String y = new String("Chuck");
```

x and y store different memory addresses.



References

```
String x = "Chuck";
String y = "Chuck";
x = null;
```

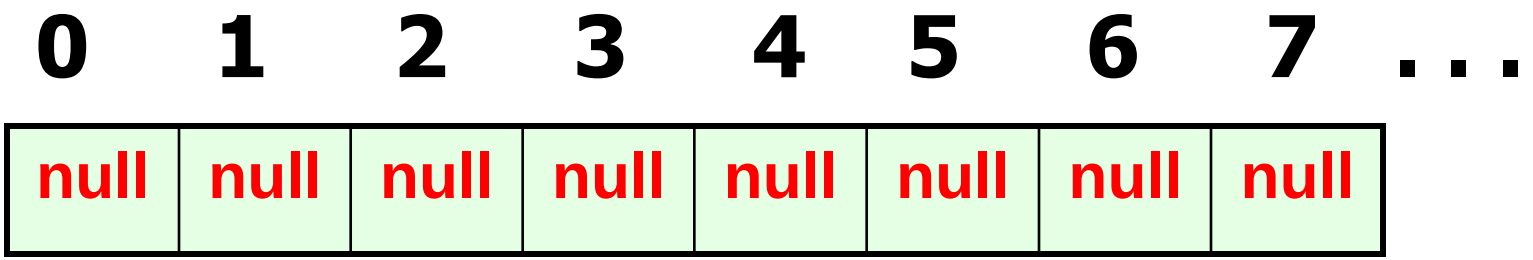


string_references.java

String Arrays

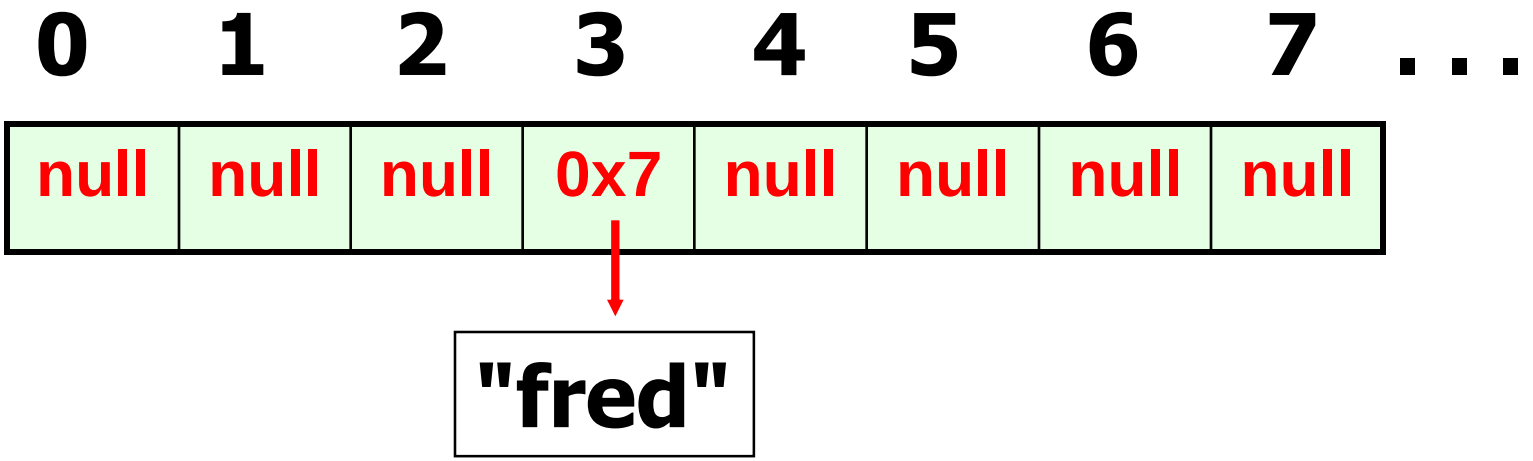
Array of References

```
String[] list = new String[50];
//all 50 spots are null
```



Array of References

list[3] = "fred";



Array of References

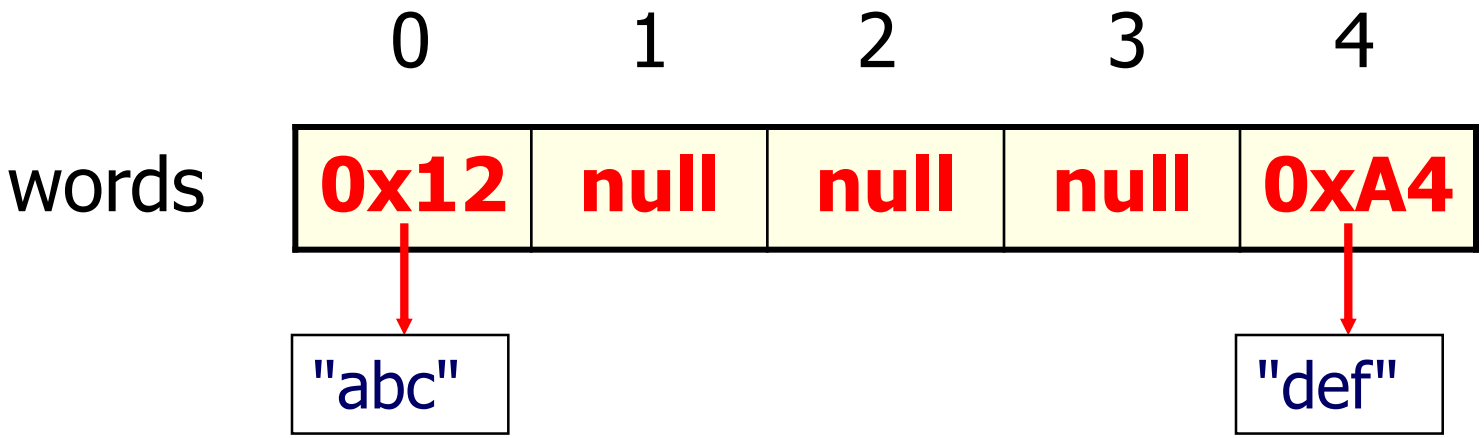
```
String[] words = new String[5];
words[0] = "abc";
words[4] = "def";
out.println(words[0]);
out.println(words[4]);
out.println(words[1]);
```

OUTPUT

```
a
bc
def
null
```

Array of References

```
String[] words = new String[5];
words[0] = "abc";
words[4] = "def";
```



Array of References

```
String s  = "one two four five";

String[] words = s.split(" ");

out.println(words[0]);
out.println(words[1]);
out.println(words[3]);
```

OUTPUT

```
one
two
five
```

stringray.java
splitone.java

Array of References

```
String s  = "10?25?109?1?23?18";
String[] nums = s.split("\\?");


int sum = 0;
for(String num : nums )
  sum += Integer.parseInt(num);
System.out.println( sum );
```

OUTPUT

186

splittwo.java
splitthree.java
splitfour.java

arrayofstrings.java

Work on Programs!

Crank Some Code!

A+ Computer Science

Array of references