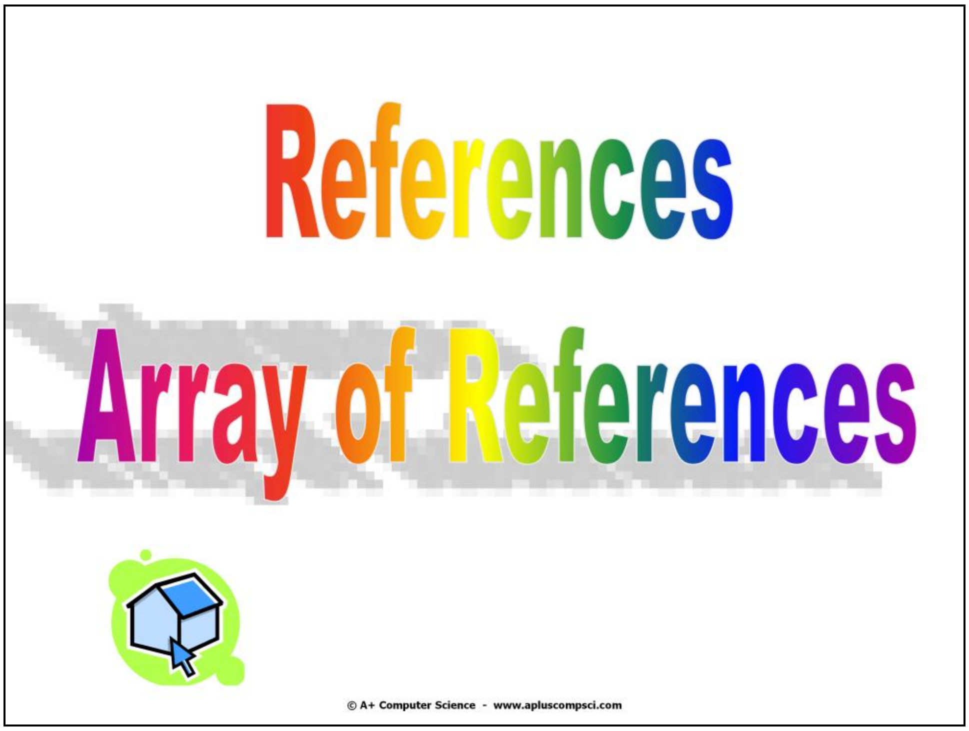
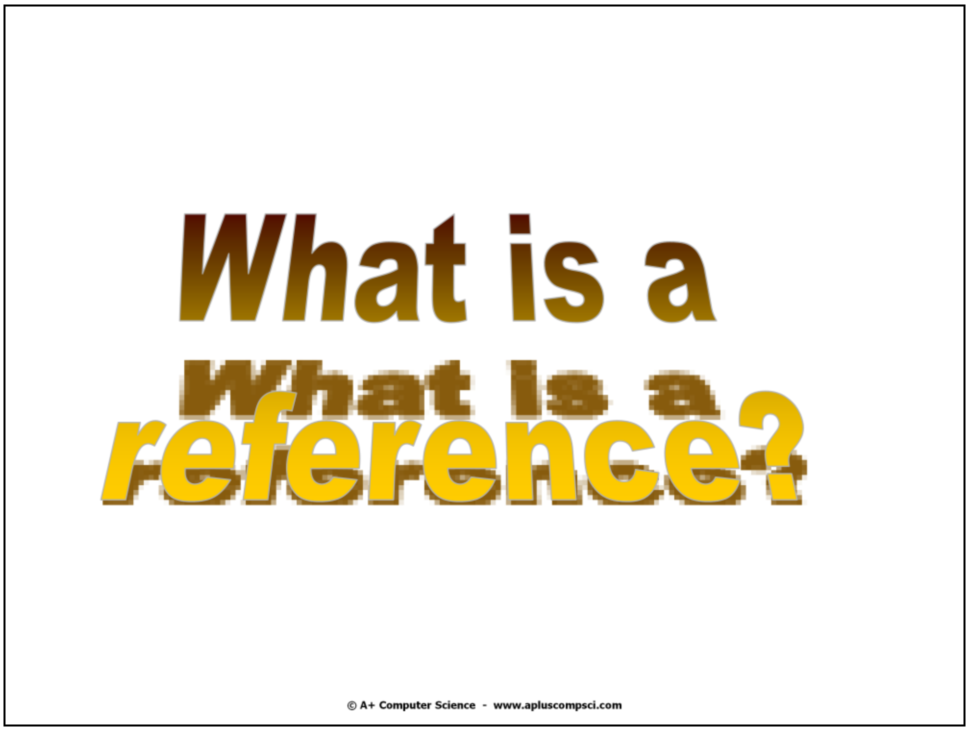
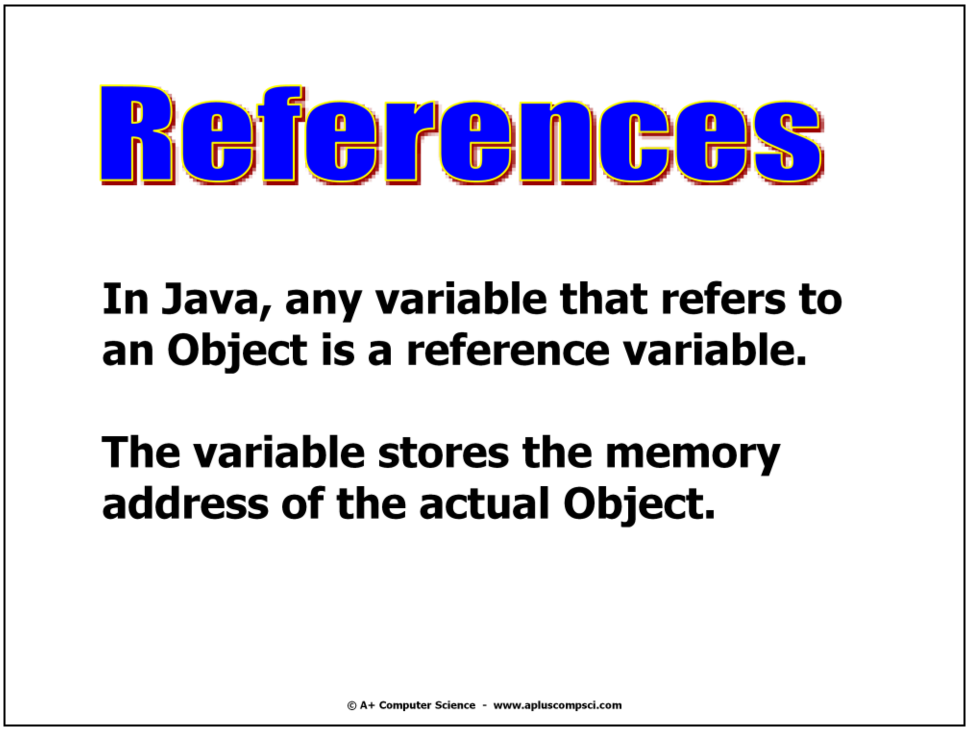


References

Array of References



What is a reference?



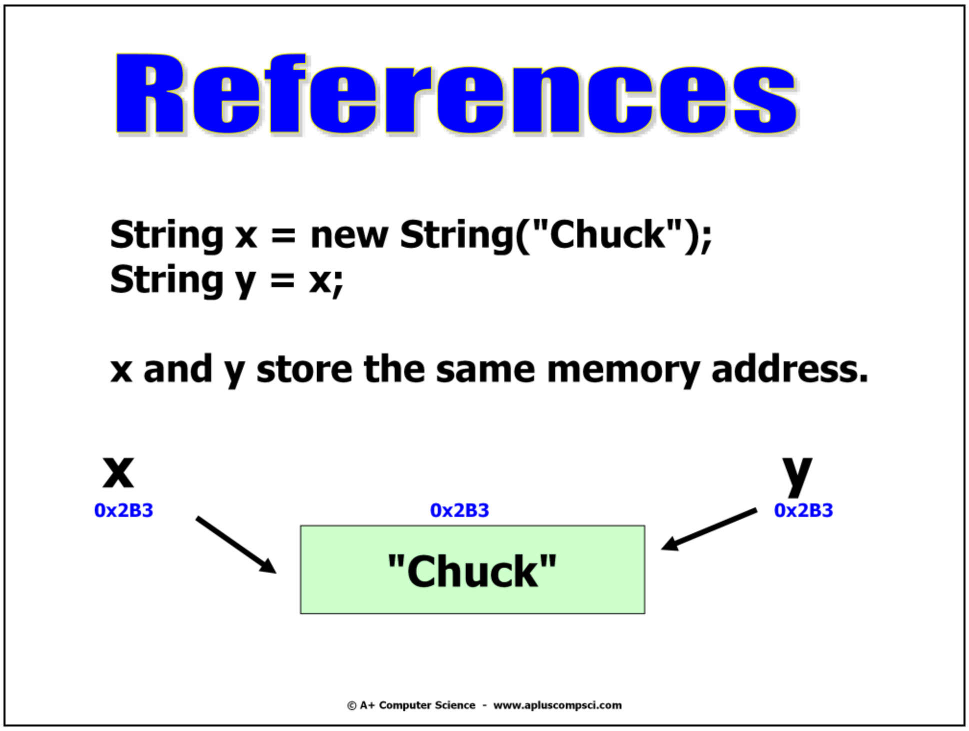
References

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

The variable stores the memory address of the actual Object.

© A+ Computer Science - www.apluscompsci.com

All variables in Java that refer to Objects are called references. Reference variables store the location / memory address of the actual Object. For most situations, the value stored in a reference is a memory address.



References

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

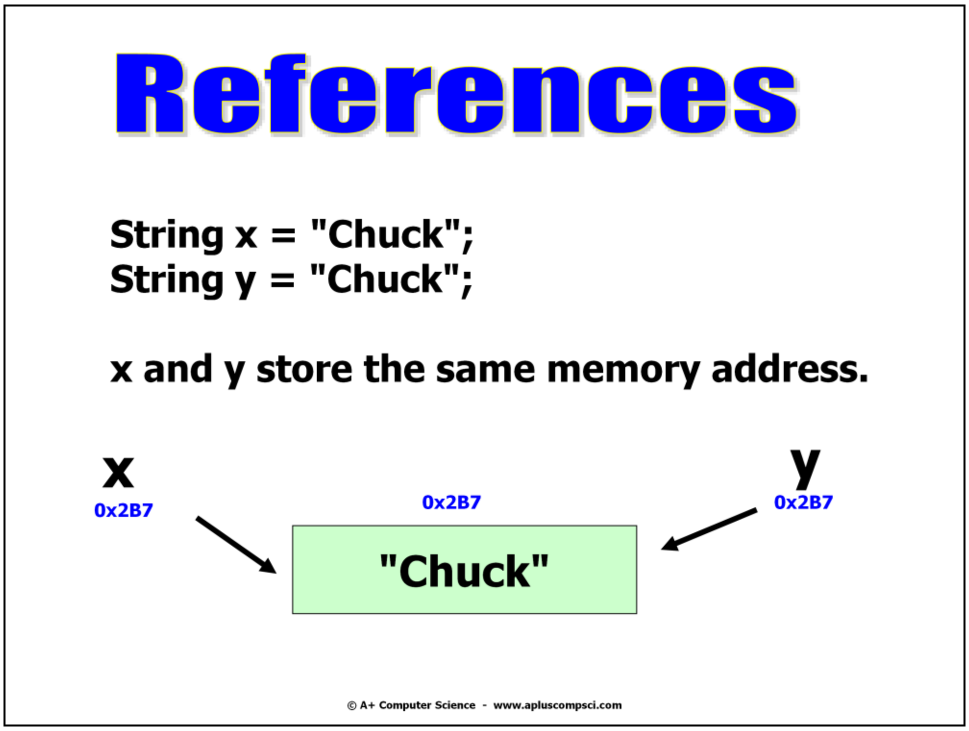
x and y store the same memory address.

x
0x2B3
0x2B3
"Chuck"
y
0x2B3
© A+ Computer Science - www.apluscompsci.com

In this example, x and y both the store the location / address of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

For this example, `x==y` is true. `x==y` compares the values stored in x and y. x and y both store the same location / address.

For this example, `x.equals(y)` is true. `x.equals(y)` compares the contents of the Objects referred to by x and y. `Chuck` is being compare to `Chuck`.



References

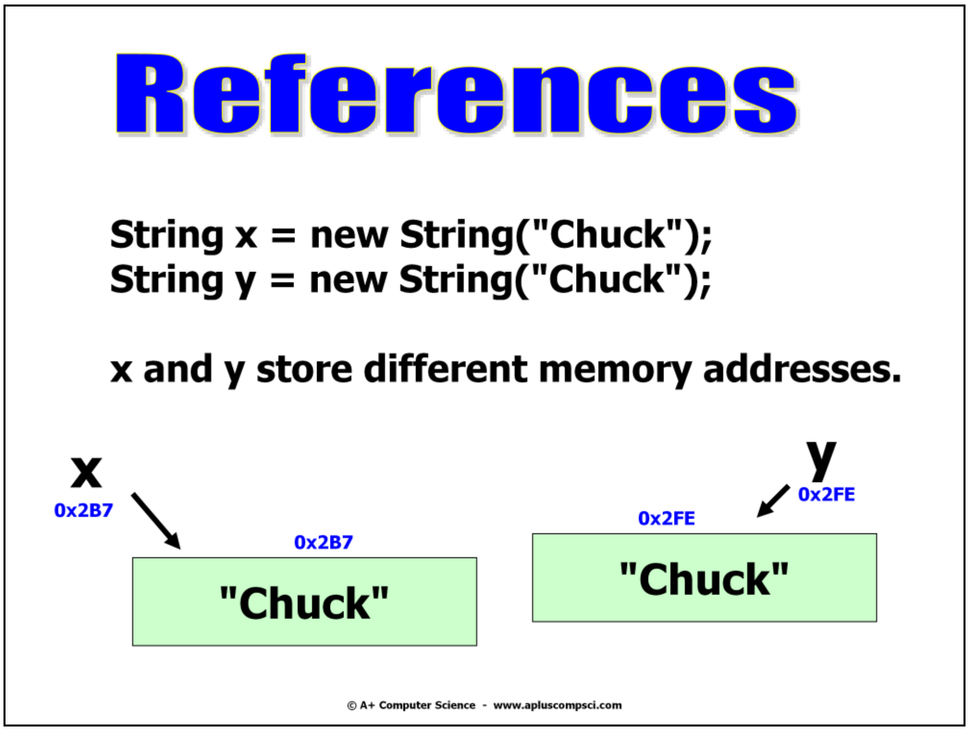
```
String x = "Chuck";
String y = "Chuck";
```

x and y store the same memory address.

In this example, x and y both the store the location of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

For this example, `x==y` is true. `x==y` compares the values stored in x and y. x and y both store the same location / address.

For this example, `x.equals(y)` is true. `x.equals(y)` compares the contents of the Objects referred to by x and y. `Chuck` is being compare to `Chuck`.



References

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

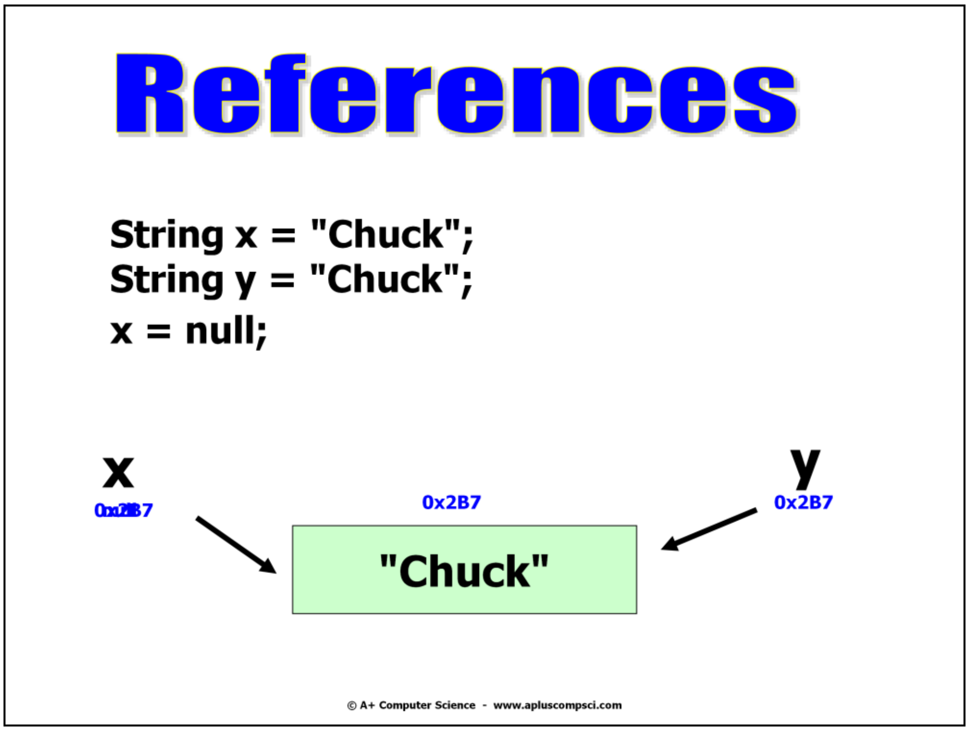
x and y store different memory addresses.

x
0x2B7
0x2B7
"Chuck"
y
0x2FE
0x2FE
"Chuck"

In this example, x stores the location / address of a String Object that stores the value `Chuck`. y also stores the location of a different String Object that stores the value `Chuck`. x and y do not store the same location / address.

For this example, `x==y` is false. x and y do not store the same location / address.

For this example, `x.equals(y)` is true.



References

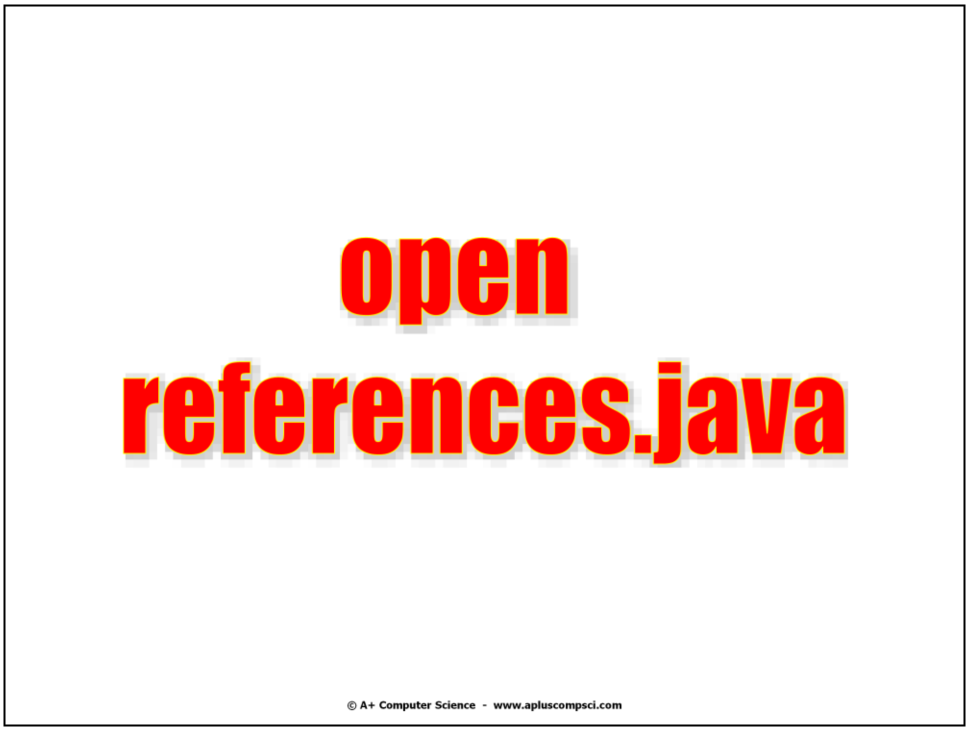
```
String x = "Chuck";
String y = "Chuck";
x = null;
```

In this example, x and y both the store the location / address of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

At the start, `x==y` is true.

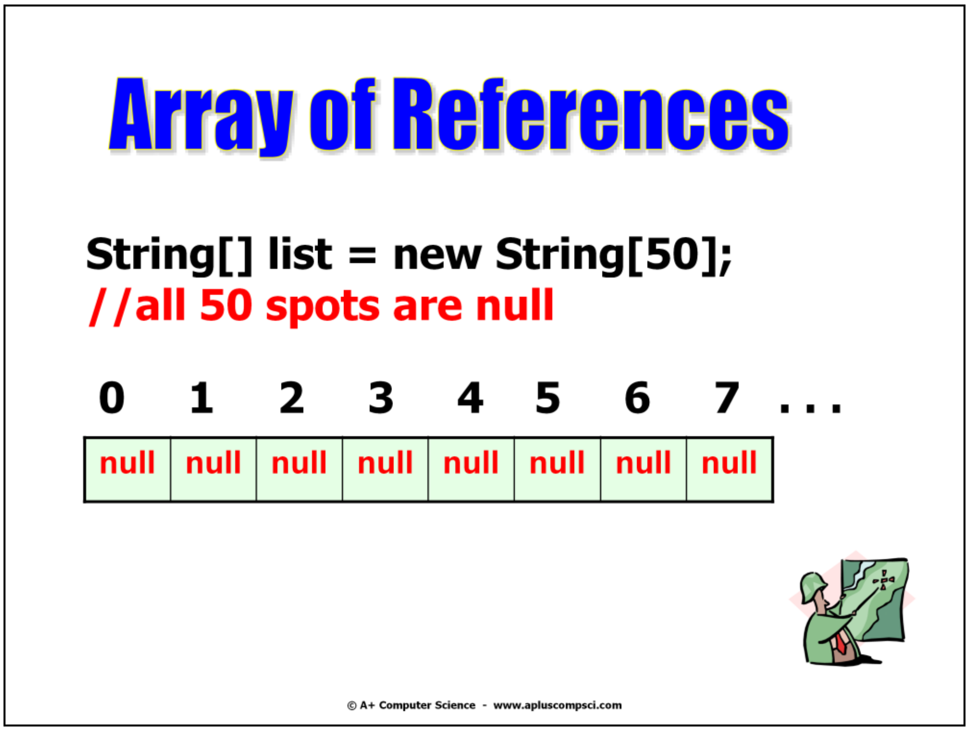
x is then referred to null. x now stores null. y was in no way changed. y still stores the address of `Chuck`.

After changing the value of x, `x==y` is false.



open references.java

Array of References



Array of References

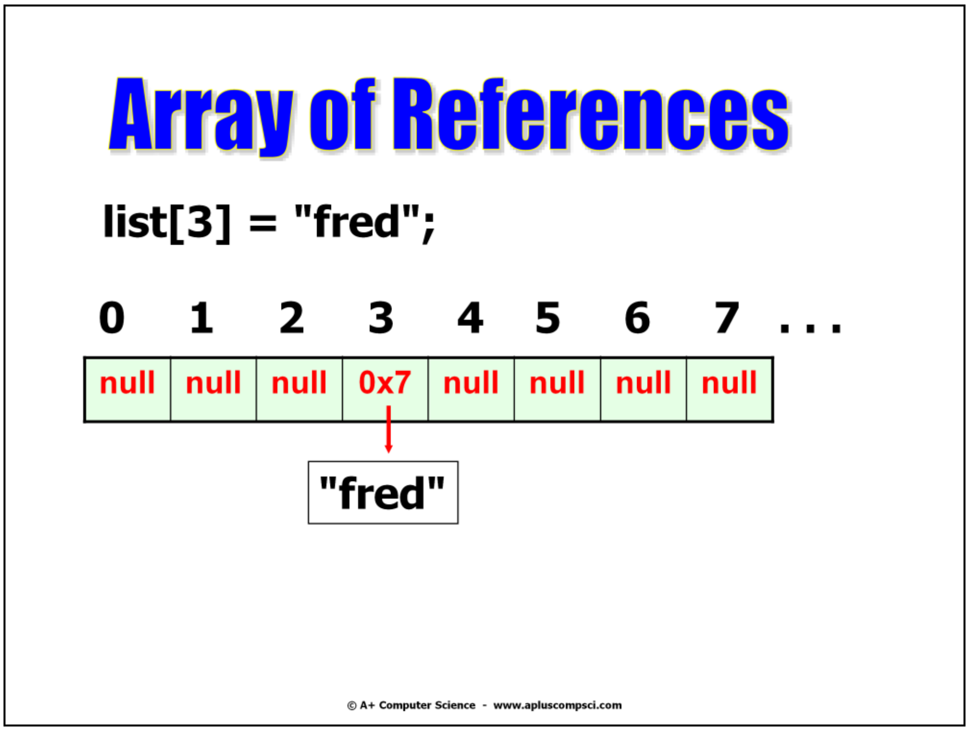
```
String[] list = new String[50];
//all 50 spots are null
```

0	1	2	3	4	5	6	7	. . .
null	null	null	null	null	null	null	null	

© A+ Computer Science - www.apluscompsci.com

In this example, list is an array of String references. list does not store Strings. list stores the locations of String Objects and in most cases list stores the actual memory address of String Objects.

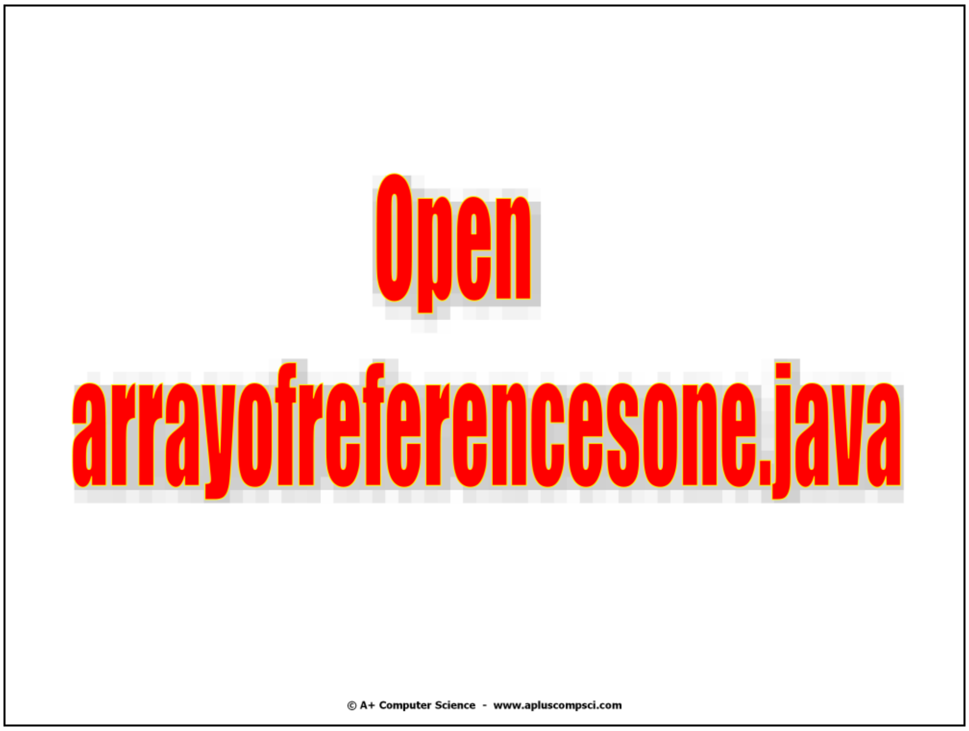
When instantiated, list would store null in all spots.



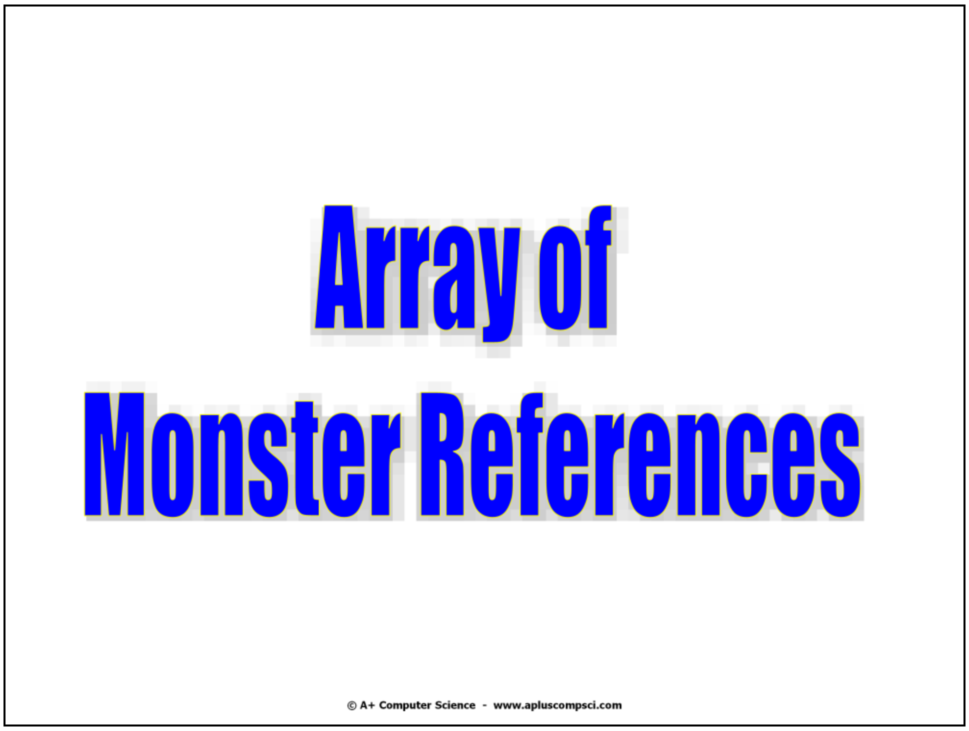
Array of References

list[3] = "fred";

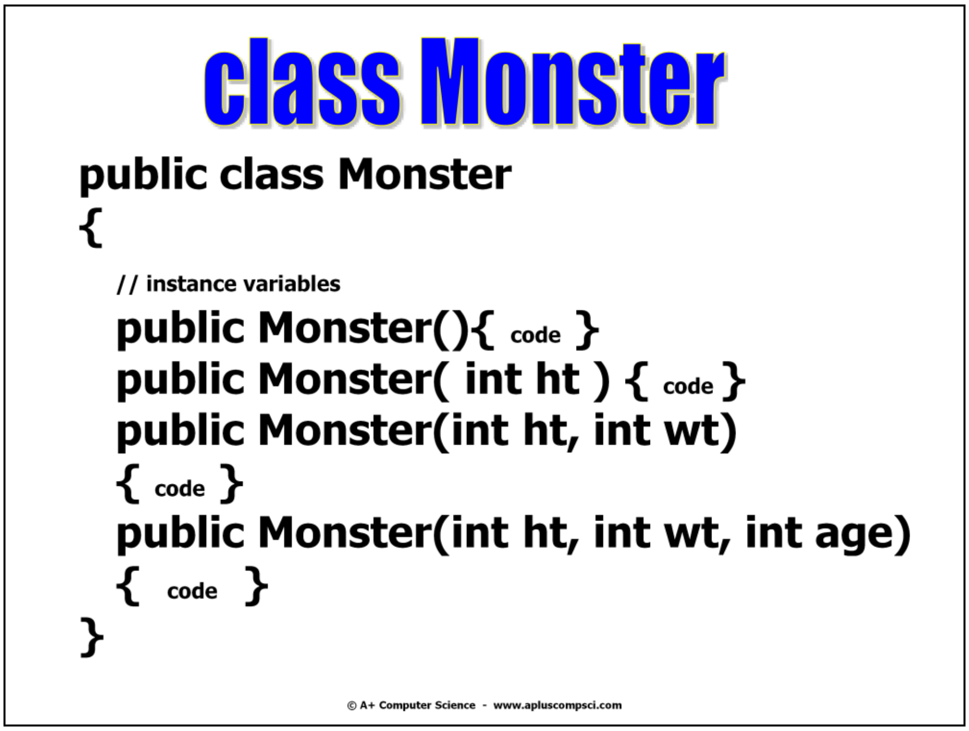
`list[3] = "fred"` assigns the location / address of `"fred"` to spot 3 in the array. All other spots in the array are still null.



Open arrayofreferencesone.java

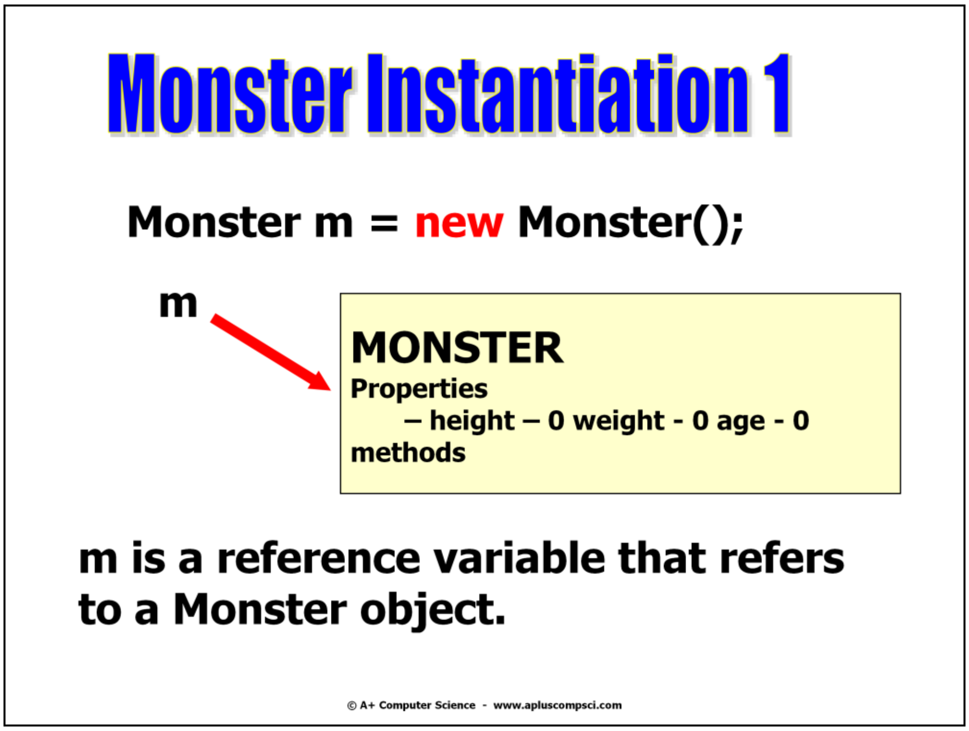


Array of Monster References



class 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  }
}
```



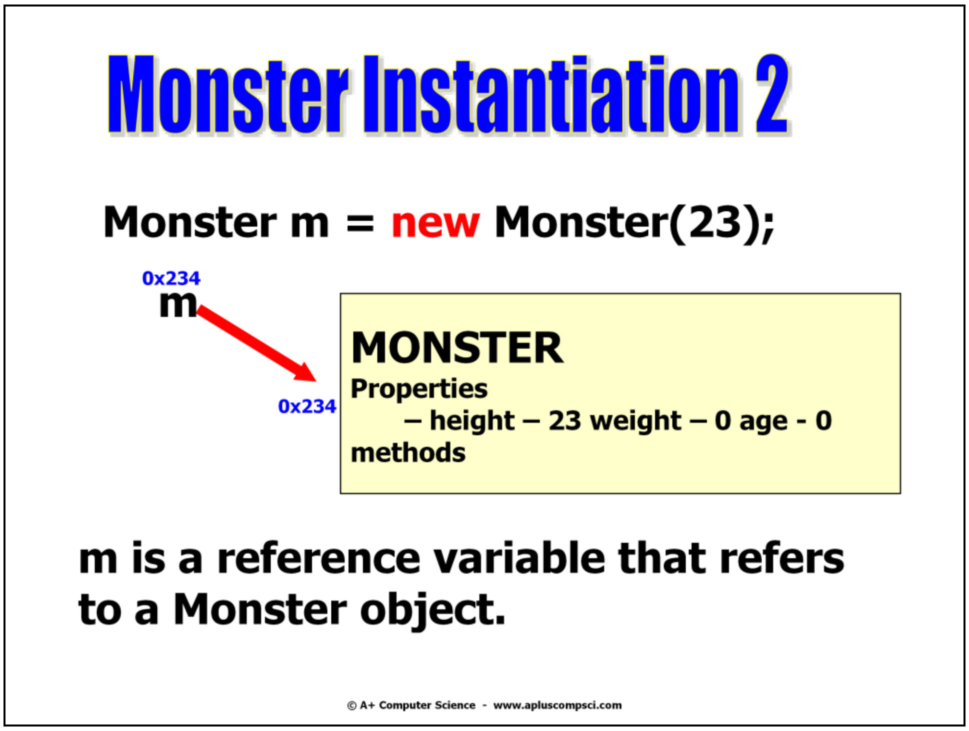
Monster Instantiation 1

```
Monster m = new Monster();
```

m

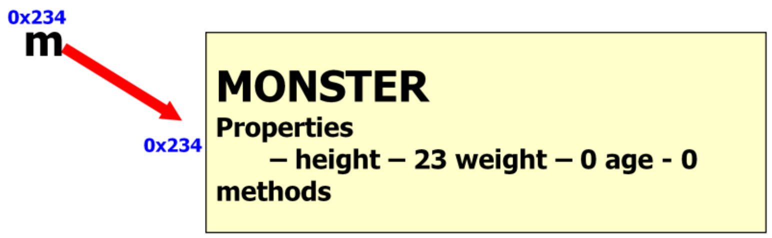
MONSTER
Properties
– height – 0 weight - 0 age - 0
methods

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

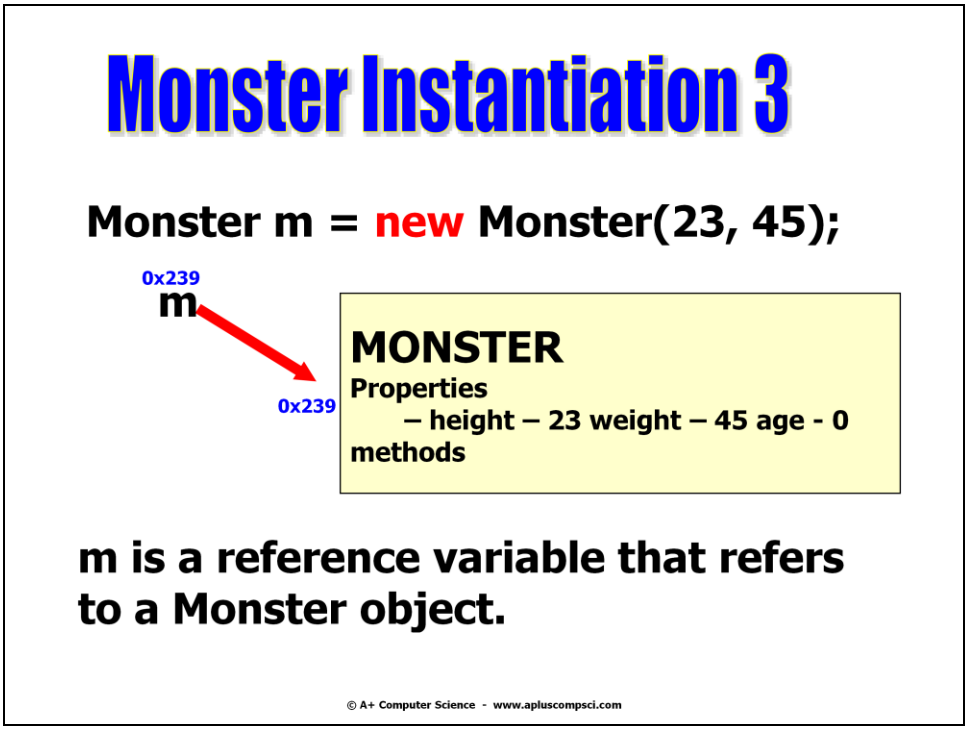


Monster Instantiation 2

```
Monster m = new Monster(23);
```



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



Monster Instantiation 3

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

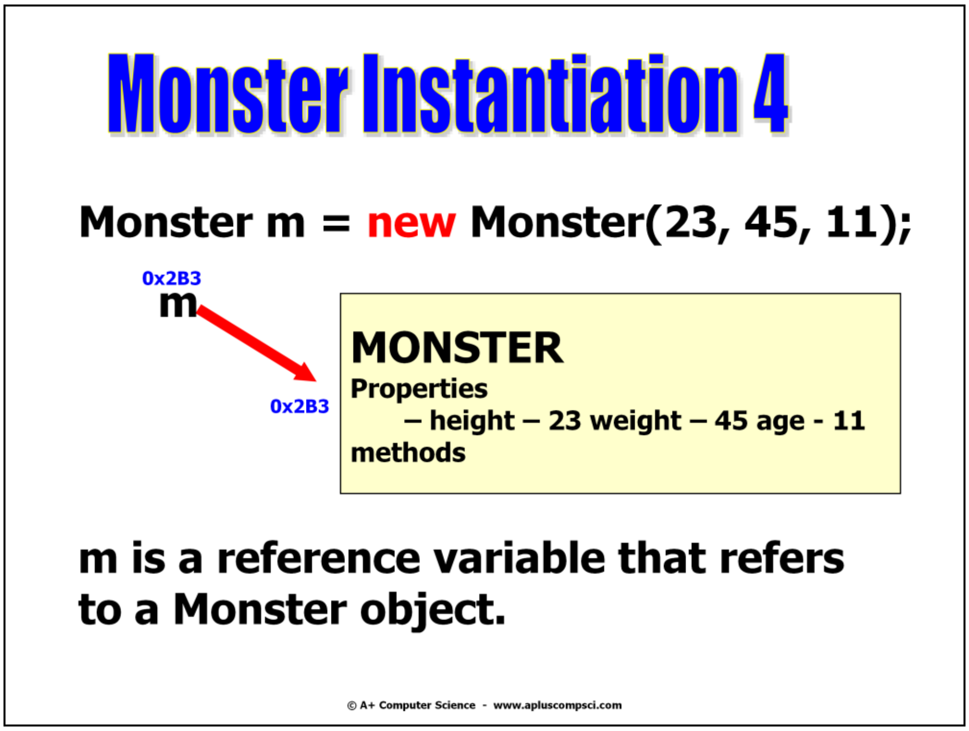
0x239
m

0x239

MONSTER
Properties
– height – 23 weight – 45 age - 0
methods

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

© A+ Computer Science - www.apluscompsci.com

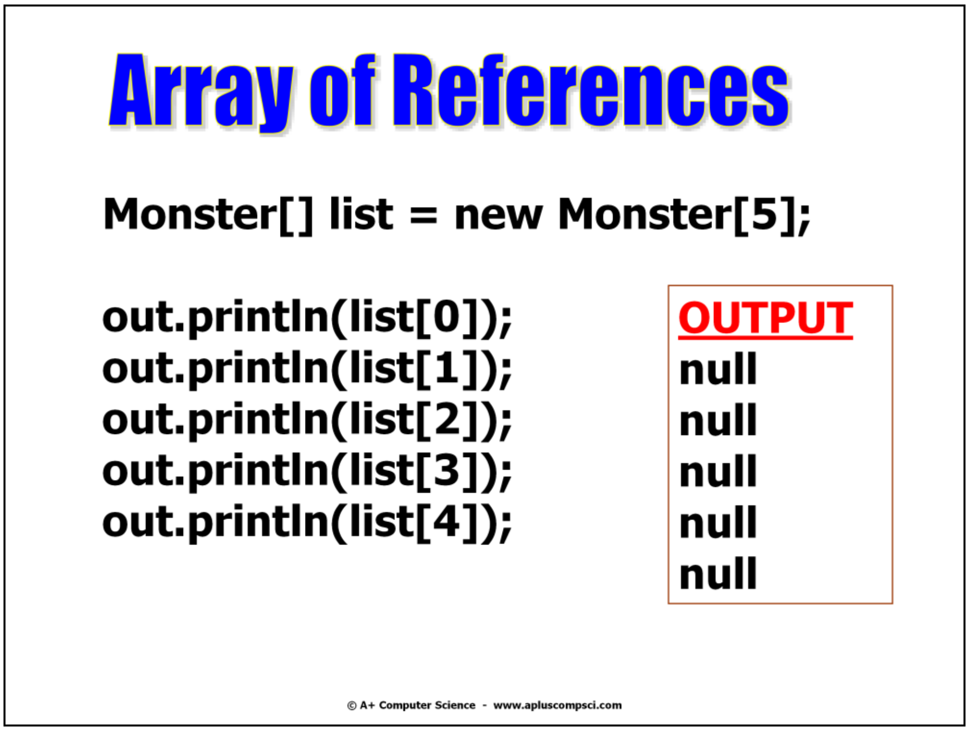


Monster Instantiation 4

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

0x2B3
m
0x2B3
MONSTER
Properties
– height – 23 weight – 45 age - 11
methods

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



Array of References

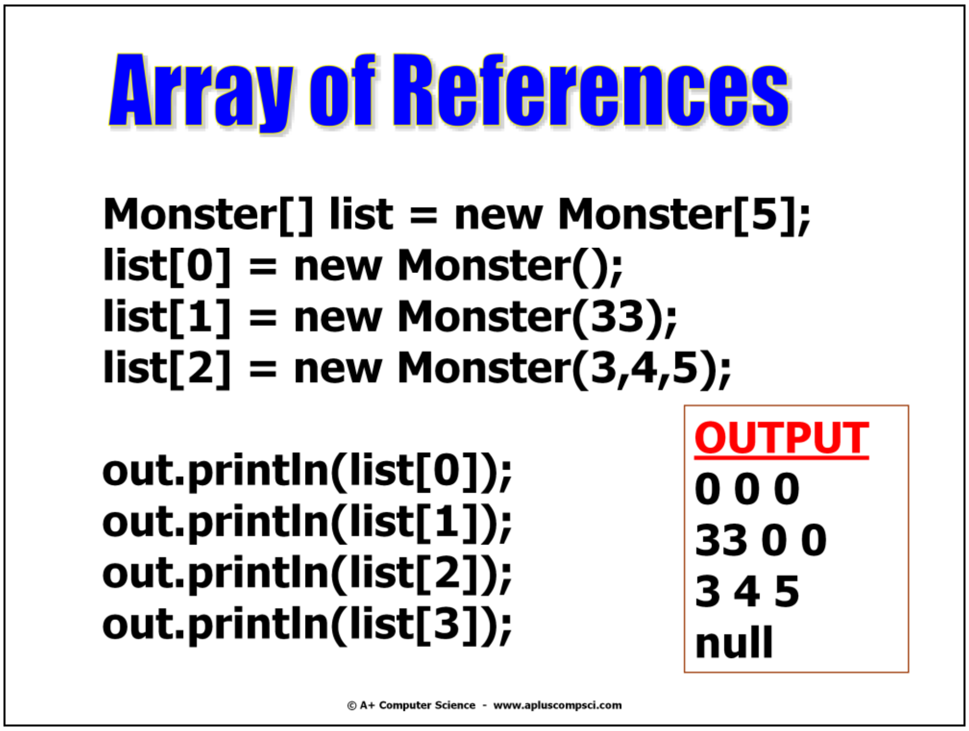
```
Monster[] list = new Monster[5];

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
```

© A+ Computer Science - www.apluscompsci.com

List is storing Monster references. List has been instantiated and has the capacity to store 5 Monster references. All spots in list are 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
```

© A+ Computer Science - www.apluscompsci.com

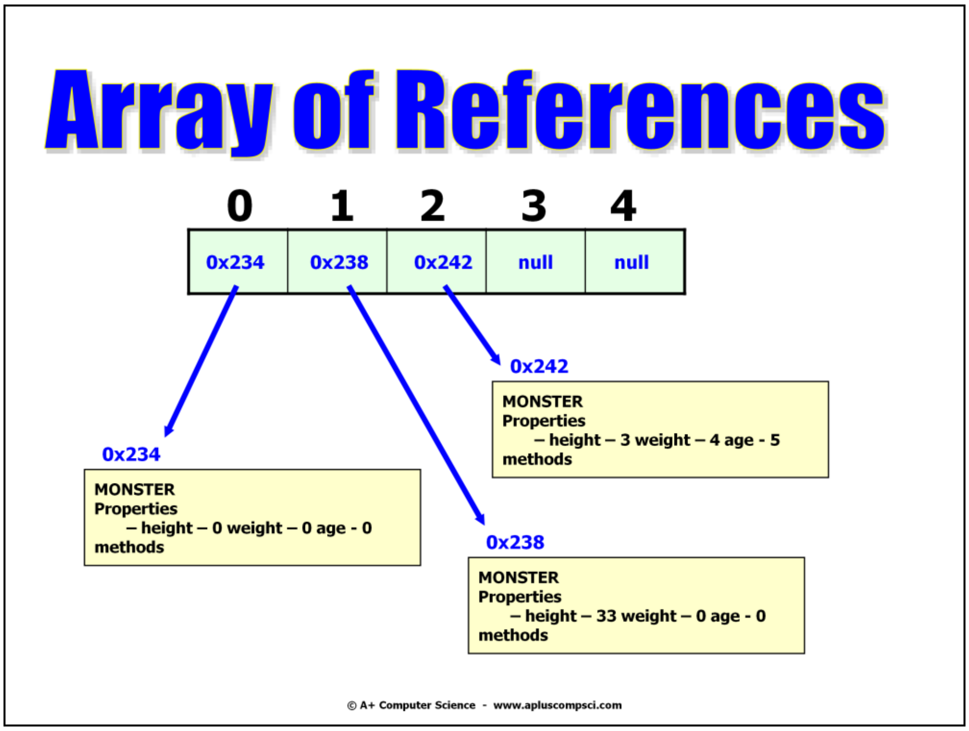
List is storing Monster references. List has been instantiated and has the capacity to store 5 Monster references.

spot 0 is storing the address of a default Monster.

spot 1 is storing the address of a Monster with ht of 33.

spot 0 is storing the address of a Monster with a ht of 3, a wt of 4, and an age of 5.

All other spots are null.


Array of References

0 1 2 3 4
0x234
0x238
0x242
null
null
0x242
MONSTER
Properties
– height – 3 weight – 4 age - 5
methods
0x234
MONSTER
Properties
– height – 0 weight – 0 age - 0
methods
0x238
MONSTER
Properties
– height – 33 weight – 0 age - 0
methods
© A+ Computer Science - www.apluscompsci.com

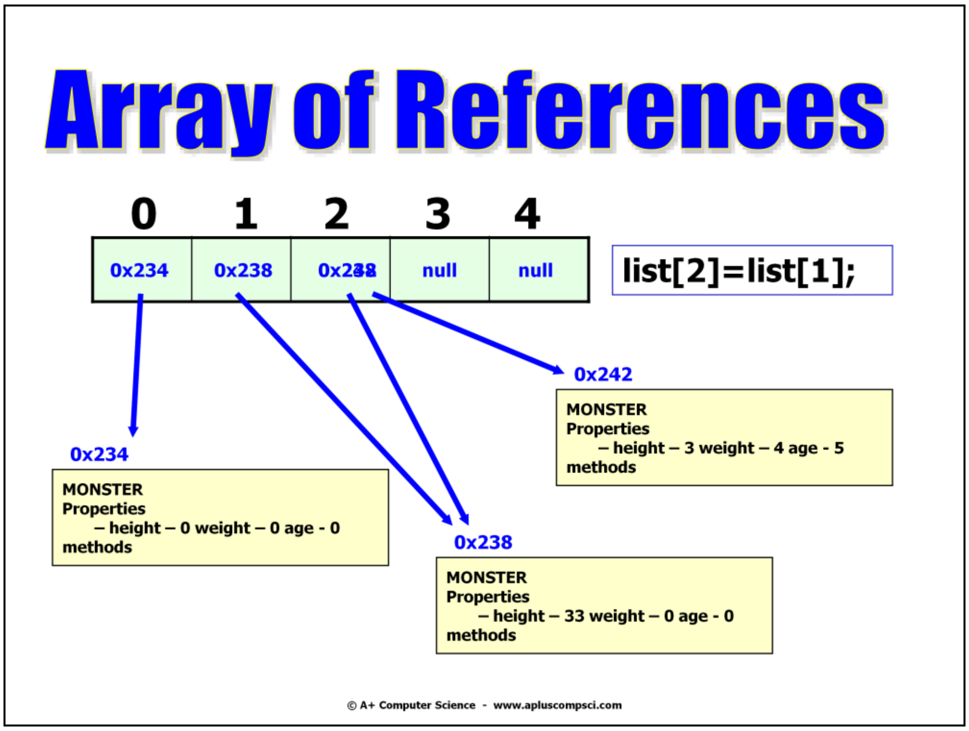
List is storing Monster references. List has been instantiated and has the capacity to store 5 Monster references.

spot 0 is storing the address of a default Monster.

spot 1 is storing the address of a Monster with ht of 33.

spot 0 is storing the address of a Monster with a ht of 3, a wt of 4, and an age of 5.

All other spots are null.

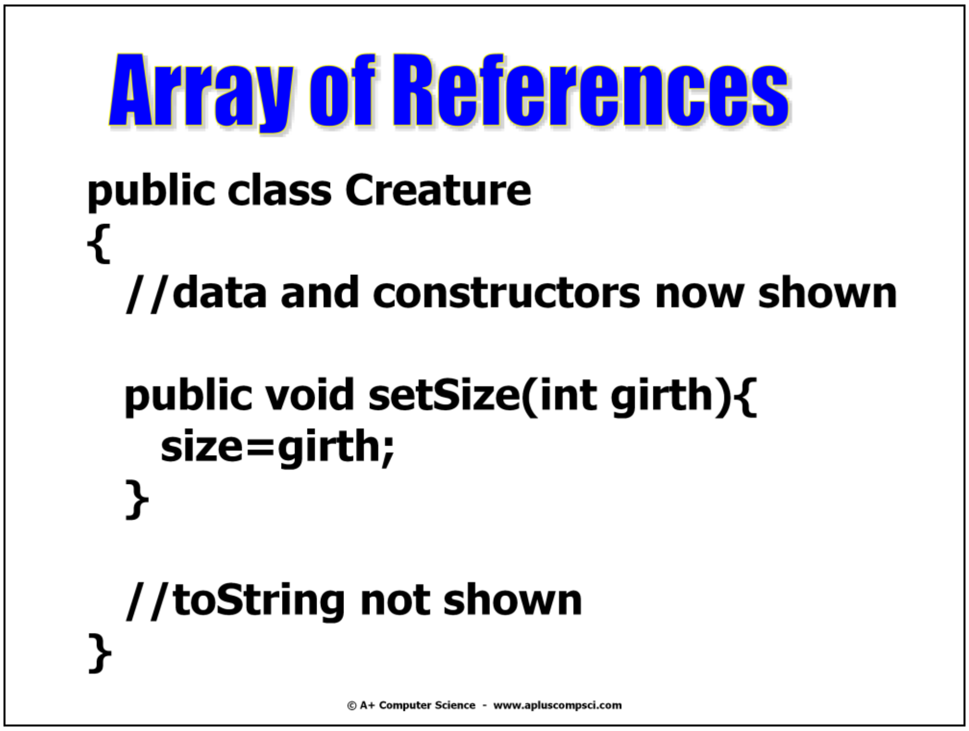

Array of References

In this example, the value of spot 1 is being copied to spot 2. spot 2 will contain the same value of as spot 1.

spot 2 was storing the address of a Monster with a ht of 3, wt of 4, and an age of 5.

After the `ray[2]=ray[1]` assignment, spot 2 is storing the address of a Monster with a ht of 33, wt of 0, and age of 0.

Open arrayofreferencestwo.java

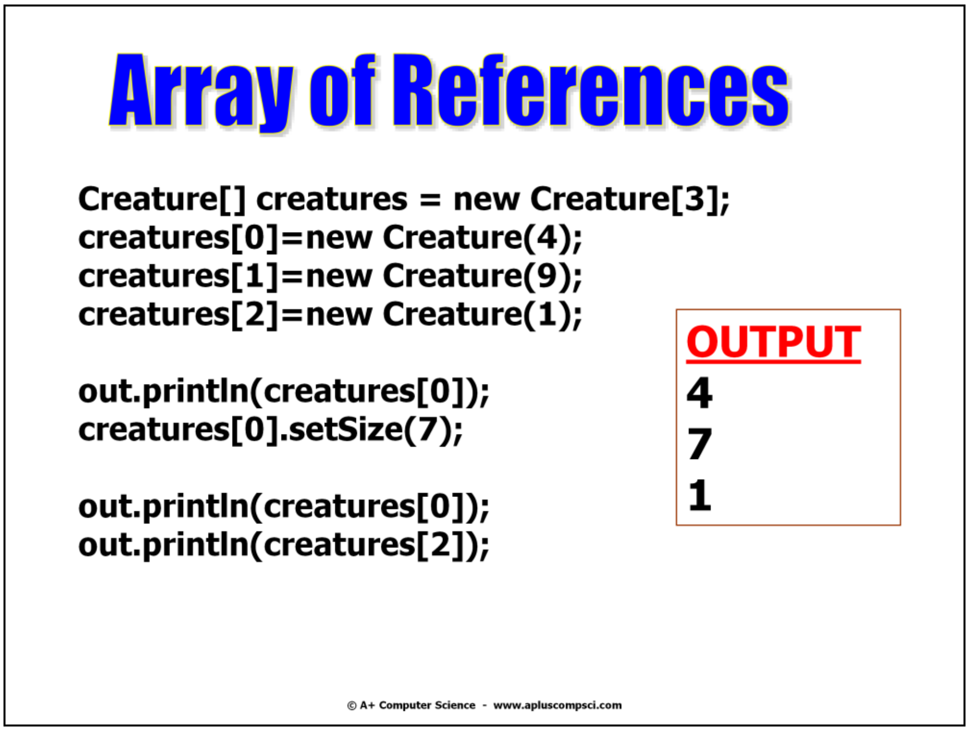


Array of References

```
public class Creature
{
  //data and constructors now shown

  public void setSize(int girth){
    size=girth;
  }

  //toString not shown
}
```

© A+ Computer Science - www.apluscompsci.com

Creatures is a class designed to store information about creatures.



Array of References

```
Creature[] creatures = new Creature[3];
creatures[0]=new Creature(4);
creatures[1]=new Creature(9);
creatures[2]=new Creature(1);

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

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

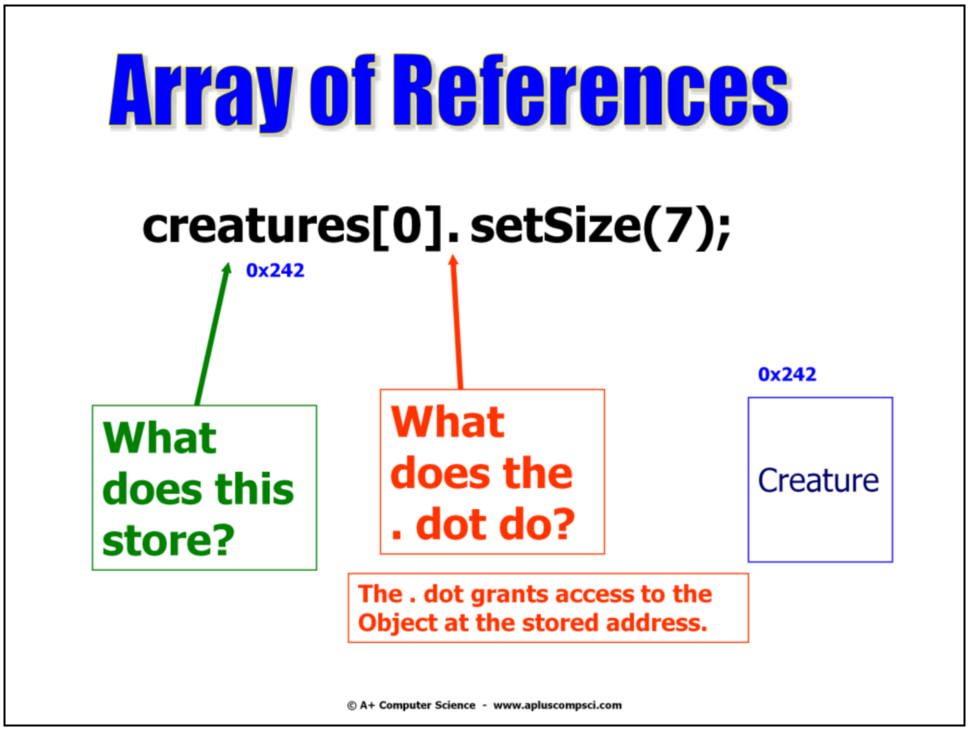
OUTPUT

```
4
7
1
```

© A+ Computer Science - www.apluscompsci.com

creatures is an array that stores addresses / locations of Creature objects.

creatures can store 3 Creature references.



Array of References

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

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

creatures[0] is storing the address / location of a Creature. When the . dot is applied to creatures[0], access is granted to the Creature objects referred to by creatures[0].

Open creature.java herd.java herdrunner.java

Start work on the labs