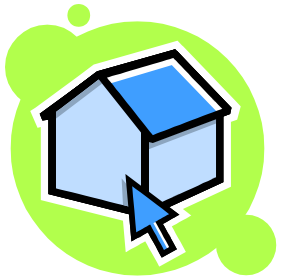
References

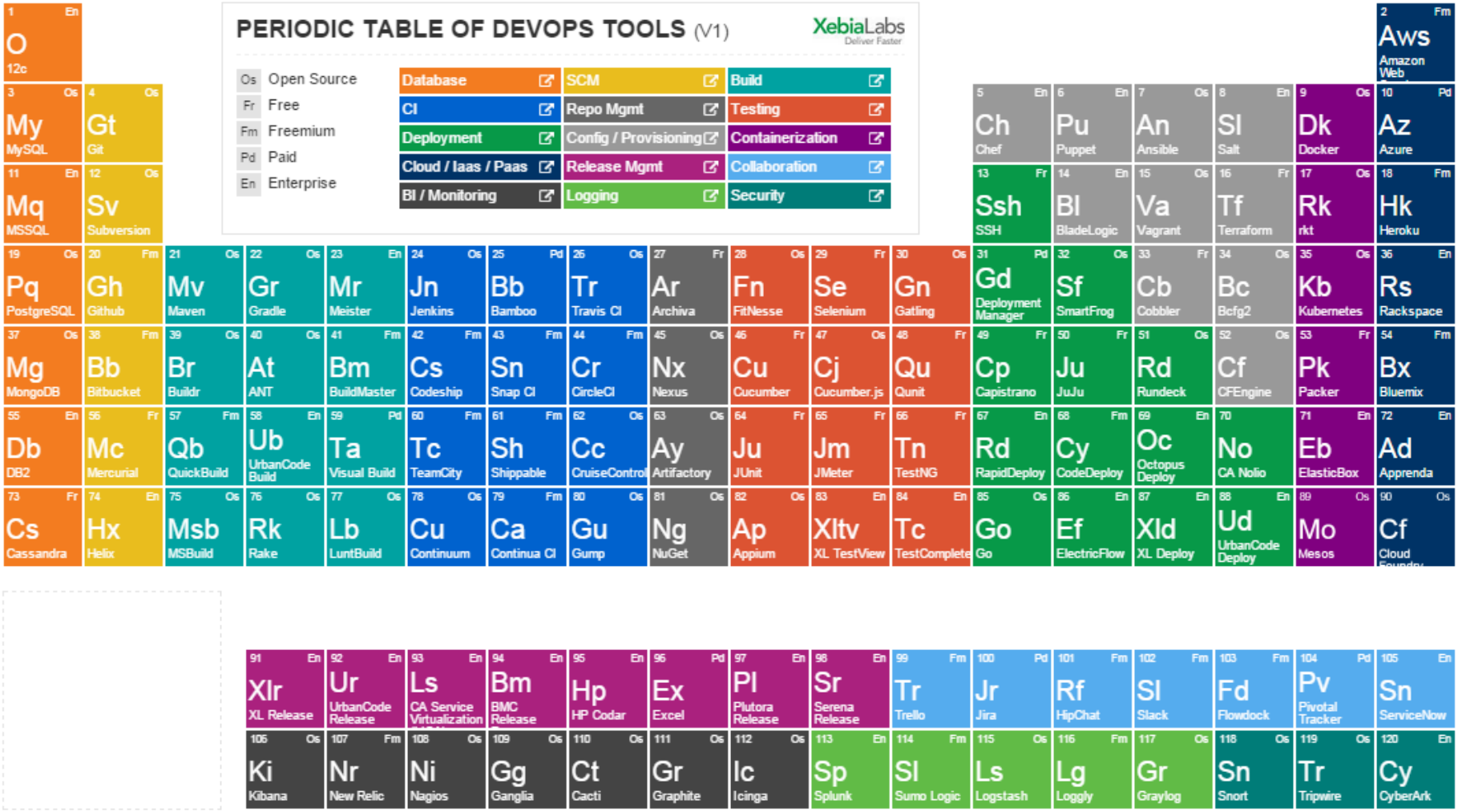
Parameters

Array of References



Lab 03

Periodic Table of Developers' Tools



XebiaLabs

What is a

What is a reference?

References

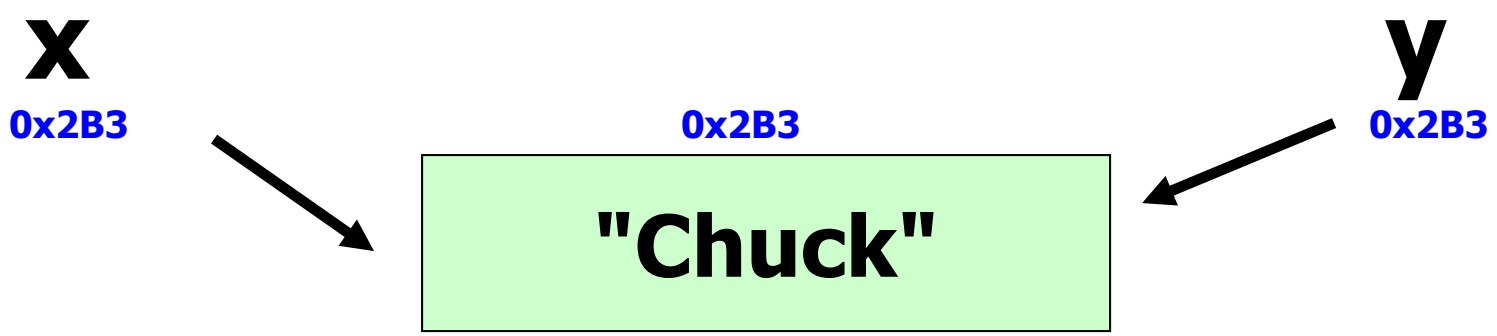
In Java, any variable that refers to an Object is a reference variable.

The variable stores the memory address of the actual Object.

References

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

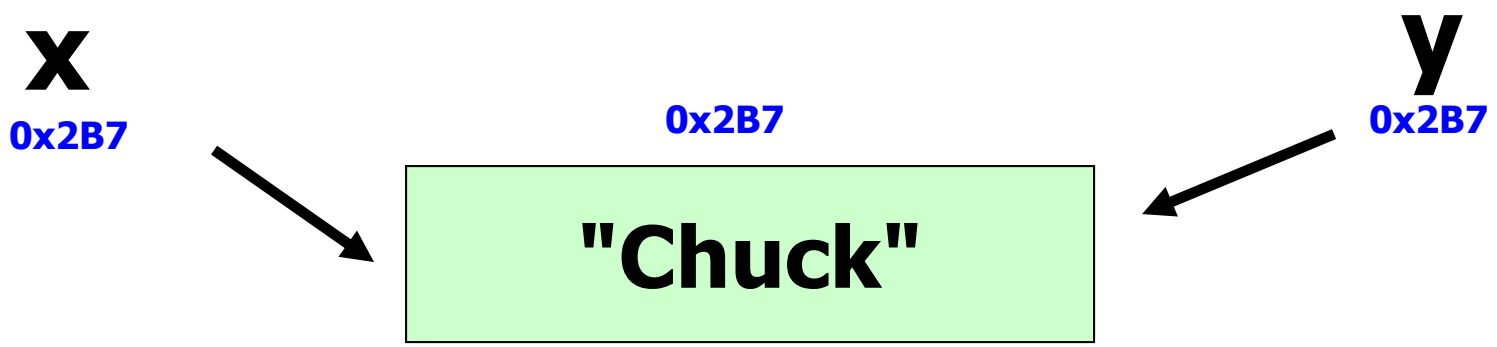
x and y store the same memory address.



References

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

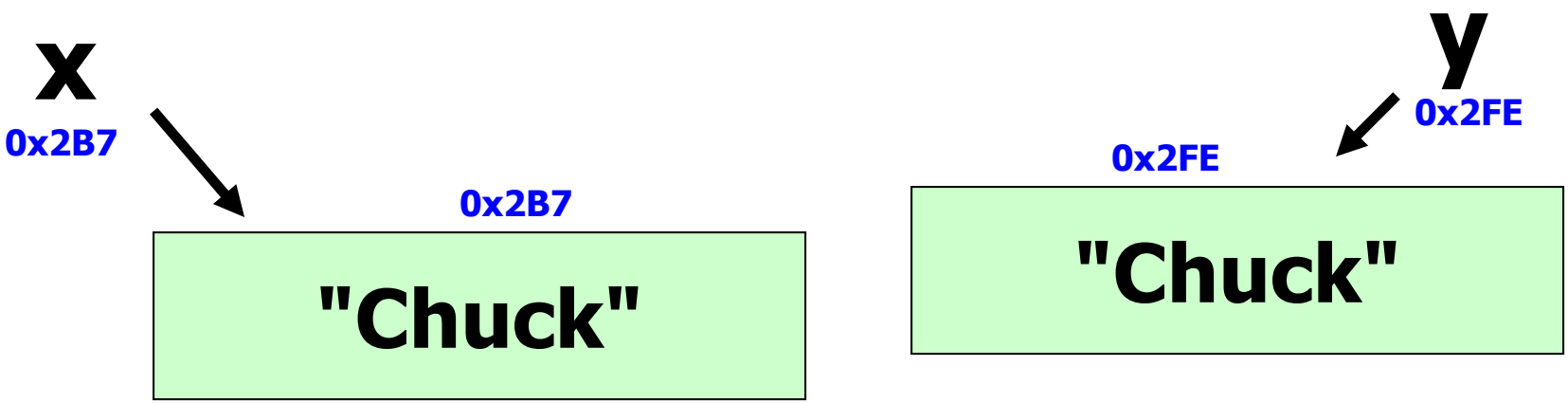
x and y store the same memory address.



References

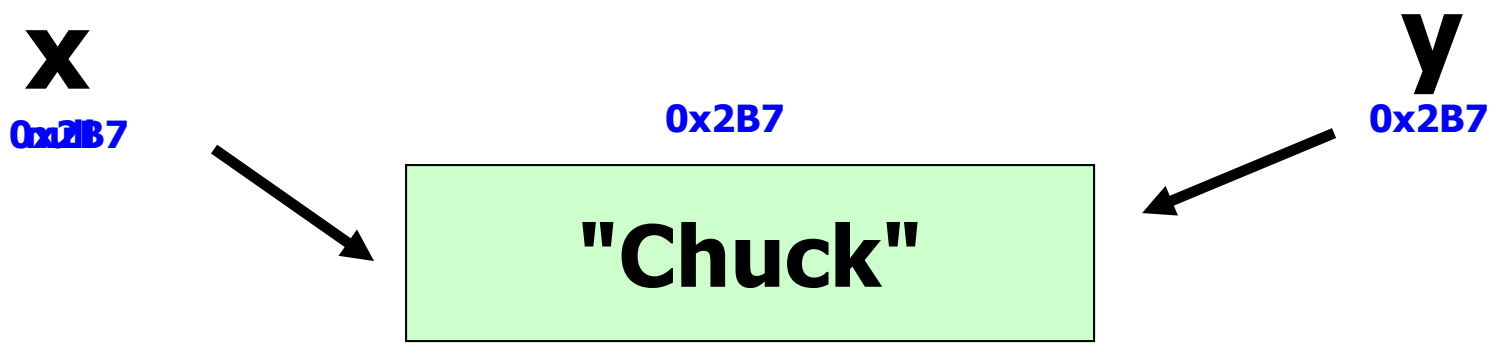
```
String x = new String("Chuck");
String y = new String("Chuck");
```

x and y store different memory addresses.



References

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



open references.java

What is a parameter?

parameters

A parameter/argument is a channel used to pass information to a method. Parameters provide important information for methods.

```
window.setColor( Color.red );
```

parameters

A parameter/argument is a channel used to pass information to a method. setColor() is a method of the Graphics class the receives a Color.

```
void setColor(Color theColor)
```

```
window.setColor( Color.red );
```

method call with parameter

parameters

```
void fillRect (int x, int y, int width, int height)
```

```
window.fillRect( 10, 50, 30, 70 );
```

method call with parameters

parameters

```
void fillRect(int x, int y, int width, int height)
```

```
window.fillRect( 10, 50, 30, 70 );
```

The call to fillRect would draw a rectangle at position 10,50 with a width of 30 and a height of 70.

Java Parameter Passing Information

Java passes all parameters by **VALUE**.

Primitives are passed as values by **VALUE**.

References are passed as addresses by **VALUE**.

Passing by Value

Passing by value simply means that a copy of the original is being sent to the method.

Passing by Value

If you are sending in a primitive, then a copy of that primitive is sent.

If you are sending in a reference or memory address, then a copy of that memory address is sent.

Passing by Value

```
public static void swap( int x, int y){
    int t = x;
    x = y;
    y = t;
    out.println(x + " " + y);
}

//test code
int one=5, two=7;
out.println(one + " " + two);
swap(one,two);
out.println(one + " " + two);
```

OUTPUT

```
5 7
7 5
5 7
```

Passing by Value

```
public static void swap( int x, int y)
{
   int t = x;
   x = y;
   y = t;
}
```

This attempted swap has local effect, but does not affect the original variables. Copies of the original variable values were passed to method swap.

Passing by Value

```
public static void swap(Integer x, Integer y){
   Integer t=x;
   x=y;
   y=t;
   out.println(x + " " + y);
}

//test code
Integer one=8, two=9;
out.println(one + " " + two);
swap(one,two);
out.println(one + " " + two);
```

OUTPUT

```
8 9
9 8
8 9
```

Passing by Value

```
public static void swap( Integer x, Integer y )
{
  Integer t=x;
  x=y;
  y=t;
}
```

This attempted swap has local effect, but does not affect the original variables. Copies of the original references were passed to method swap.

passbyvalueone.java

Passing by Value

```
public static void changeOne(int[] ray)
{
  ray[0] = 0;
  ray[1] = 1;
}

//test code
int[] nums = {5,4,3,2,1};
out.println(Arrays.toString(nums));
changeOne(nums);
out.println(Arrays.toString(nums));
```

OUTPUT

```
[5, 4, 3, 2, 1]
[0, 1, 3, 2, 1]
```

Passing by Value

```
public static void changeOne(int[] ray)
{
  ray[0] = 0;
  ray[1] = 1;
}
```

Changing the values inside the array referred to by ray is a lasting change.
A copy of the original reference was passed to method changeOne, but that reference was never modified.

Passing by Value

```
public static void changeTwo(int[] ray)
{
   ray = new int[5];
   ray[0]=2;
   out.println(Arrays.toString(ray));
}

//test code
int[] nums = {5,4,3,2,1};
changeTwo(nums);
out.println(Arrays.toString(nums));
```

OUTPUT

```
[2, 0, 0, 0, 0]
[5, 4, 3, 2, 1]
```

Passing by Value

```
public static void changeTwo(int[] ray)
{
  ray = new int[5];
  ray[0]=2;
}
```

Referring ray to a new array has local effect, but does not affect the original reference.
A copy of the original reference was passed to method changeTwo.

passbyvaluetwo.java

Passing by Value

```
class A{
  private String x;
  public A( String val ){
    x = val;
  }
  public void change( ){
    x = "x was changed";
  }
  public String toString(){
     return x;
  }
}

class B{
  public void mystery(A one, A two) {
    one = two;
    two.change();
  }
}

//test code in the main in another class
B test = new B();
A one = new A("stuff");
A two = new A("something");
System.out.println(one + " " + two);
test.mystery(one,two);
System.out.println(one + " " + two);
```

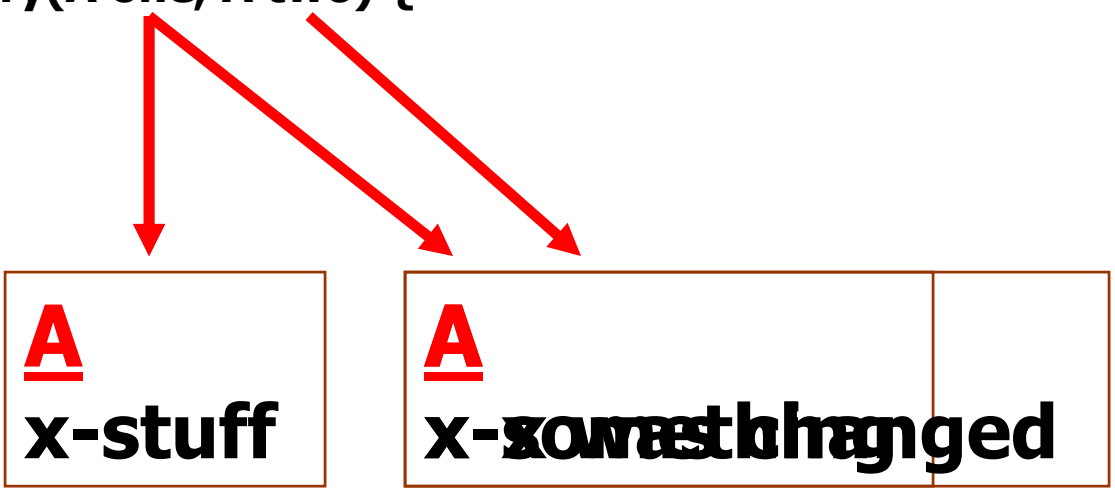
OUTPUT

stuff something
stuff x was changed

Passing by Value

```
class A{
   private String x;
   public A( String val ){
     x = val;
   }
   public void change( ){
     x = "x was changed";
   }
   public String toString(){
      return x;
   }
}

class B{
   public void mystery(A one, A two) {
     one = two;
     two.change();
   }
}
```



Passing by Value

```
class A{
  private String x;
  public A( String val ){
    x = val;
  }
  public void change( ){
    x = "x was changed";
  }
  public String toString(){
     return x;
  }
}

class B{
  public void mystery(A one, A two) {
    one = two;
    two.change();
  }
}

//test code in the main in another class
B test = new B();
A one = new A("stuff");
A two = new A("something");
System.out.println(one + " " + two);
test.mystery(one,two);
System.out.println(one + " " + two);
```

mystery() is passed the address of one and the address of two.

two's address is copied to one. This copy has local effect.

two.change() changes the value in the A referred to by two. This change effects the entire program.

passbyvaluethree.java

Passing by Value

```
public void changeRow(int r)
{
  setVals(mat[r]);
}

public void setVals(int[] ray)
{
  ray[1]=89;
  ray[2]=87;
  ray[3]=71;
}
```

open matrixparams.java

Start work on Lab 03

Array of References

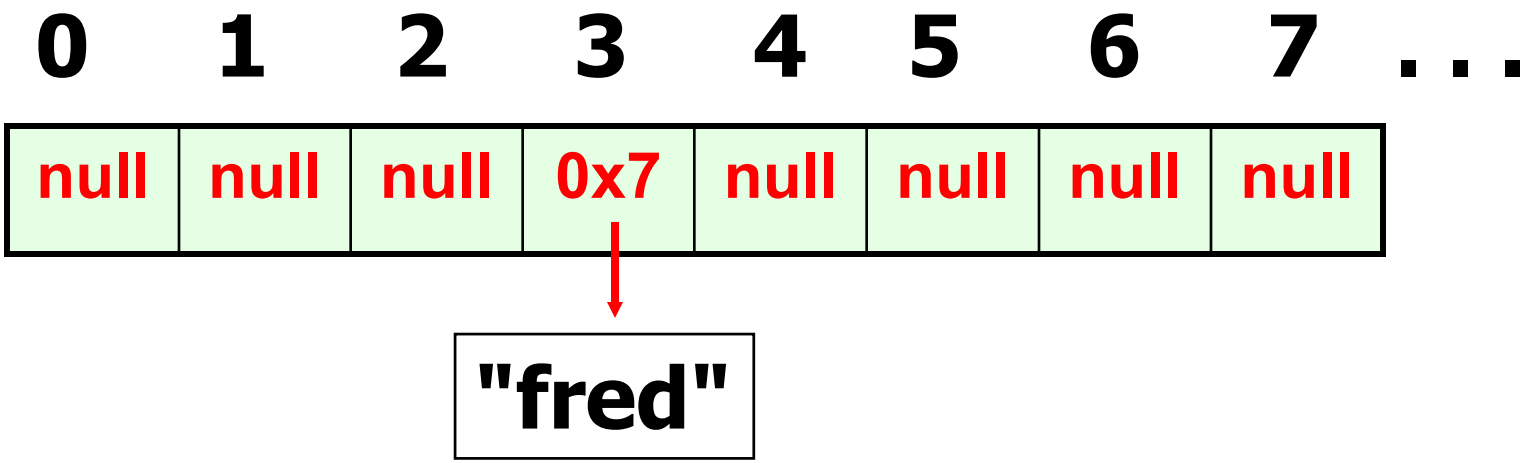
Array of References

```
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	

Array of References

list[3] = "fred";



Open arrayofreferencesone.java

Array of Monster References

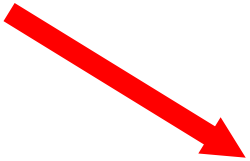
class Monster

```
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

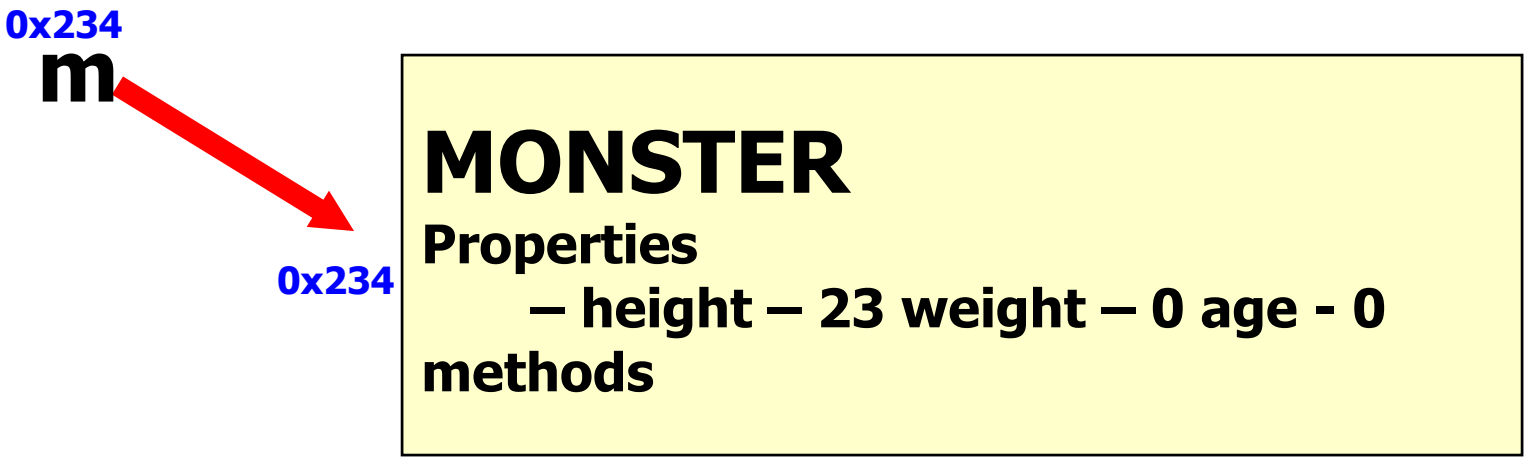
– height – 0 weight - 0 age - 0

methods

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

Monster Instantiation 2

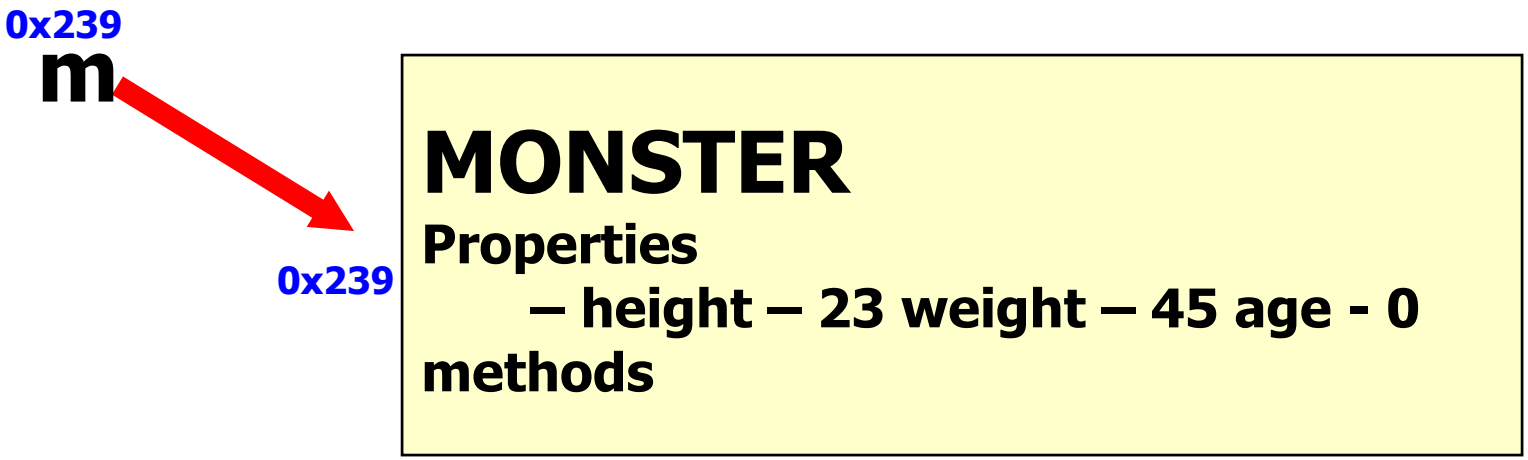
Monster m = new Monster(23);



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

Monster Instantiation 3

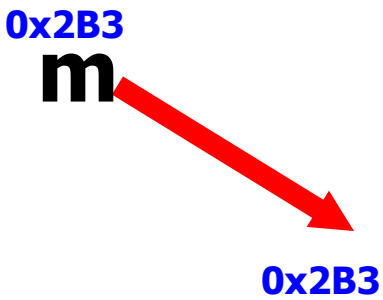
Monster m = new Monster(23, 45);



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

Monster Instantiation 4

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.

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

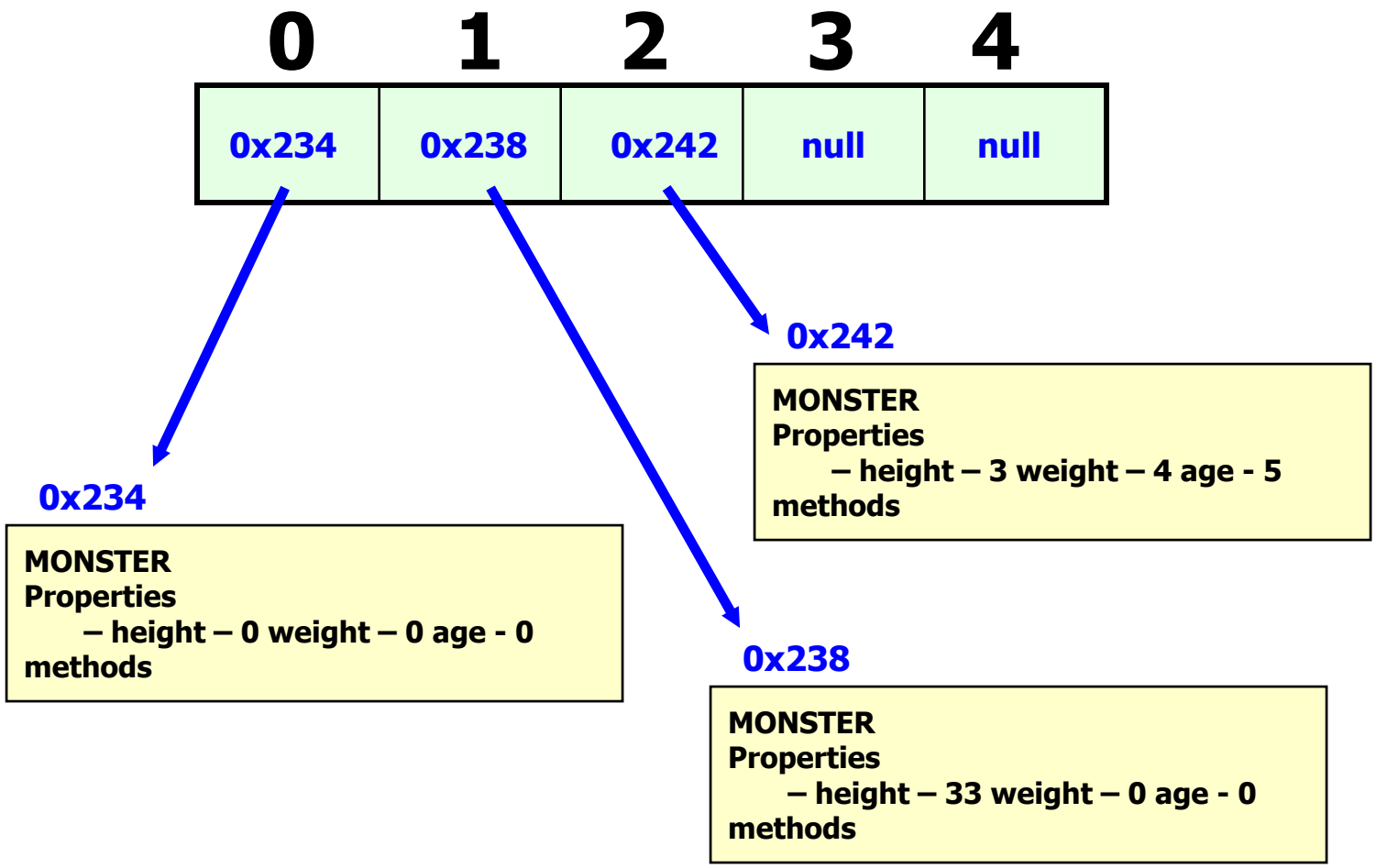
```
Monster[] list = new Monster[5];
list[0] = new Monster();
list[1] = new Monster(33);
list[2] = new Monster(3,4,5);

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

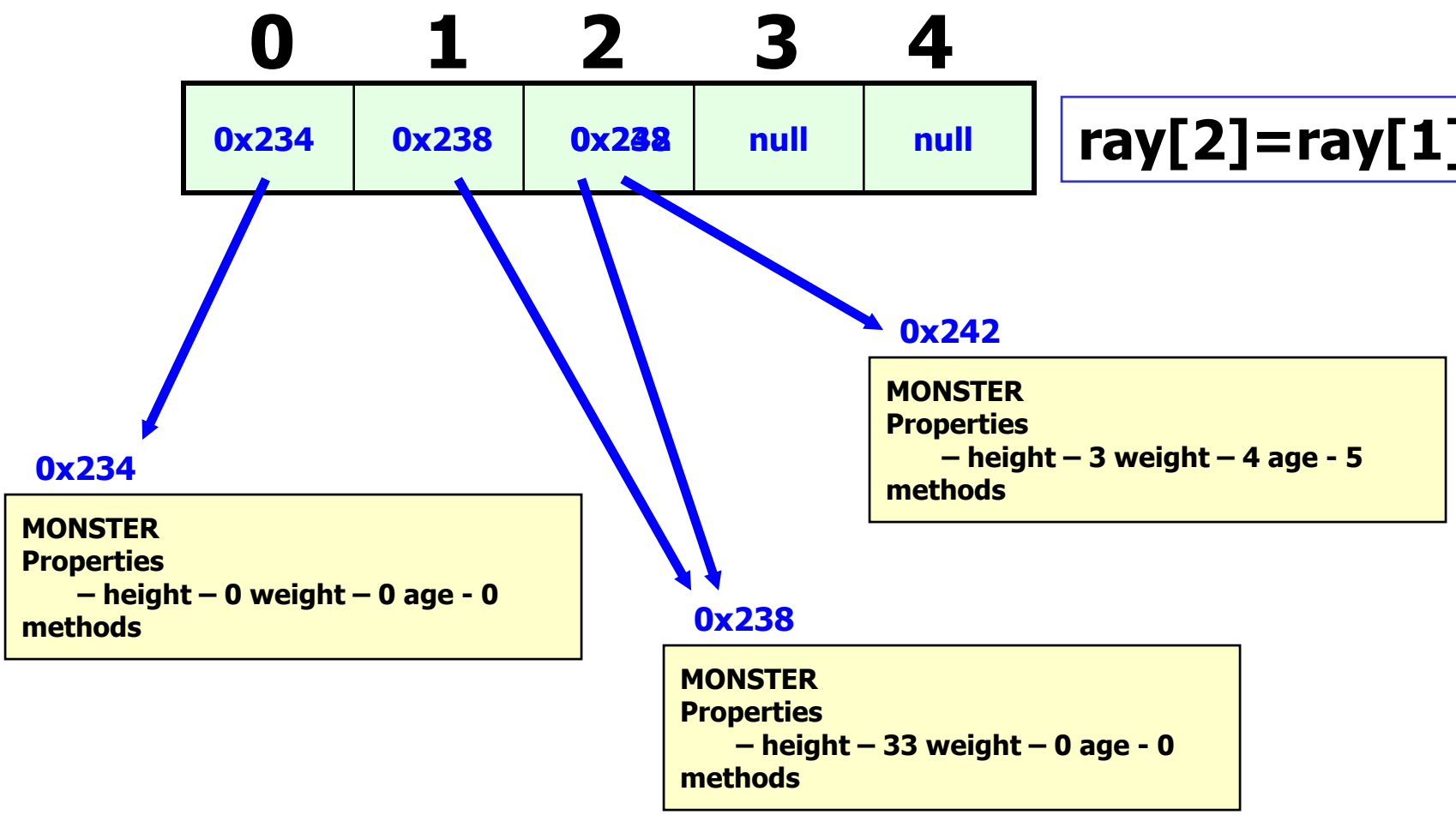
OUTPUT

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

Array of References



Array of References



Open

arrayofreferencestwo.java

Array of References

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

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

  //toString not shown
}
```

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
```

Array of References

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

0x242

What does this store?

What does the . dot do?

0x242

Creature

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

Open

arrayofreferencesthree.java

Continue work on Lab 03