References Parameters

Array of References





Periodic Table of Developers' Tools

1 En			PERIO							XebiaLal Deliver Fac	ster						2 Fm AWS Amazon Web
3 Os My MySQL	Gt		Os Open Source Fr Free Fm Freemium Pd Paid		Deployment 🗷		Repo Mgmt (2')		Containerization 3			5 En Ch Chef	6 En Pu Puppet	An Ansible	SI Salt	Dk Docker	AZ Azure
MQ MSSQL	Sv Subversion	En Enterprise		se	BI / Monitori		Release Mg Logging	C	Security 🕜				BI BladeLogic	Va Vagrant	Tf Terraform	Rk rkt	18 Frr Hk Heroku
Pq PostgreSQL	Gh Github	MV Maven	Gr Gradle	Mr Meister	Jn Jenkins	Bb Bamboo	Tr Travis Cl	27 Fr Ar Archiva 45 Os	Fn FitNesse	Se Selenium	Gn Gatling	Gd Deployment Manager	Sf SmartFrog	Cb Cobbler	Bcfg2	Kb Kubernetes	RS Rackspace
Mg MongoDB	Bb Bitbucket	Br Buildr	At ant	Bm BuildMaster	Cs Codeship	Sn Snap Cl	Cr CircleCl	Nx Nexus	Cu Cucumber	Cj Cucumber.js	Qu Qunit	Cp Capistrano	Ju JuJu 68 Fm	Rd Rundeck	Cf CFEngine	Pk Packer	BX Bluemix
Db DB2	Mc Mercurial	Qb QuickBuild	Ub UrbanCode Build	Ta Visual Build	Tc TeamCity	Sh Shippable	Cc CruiseContro	Ay Artifactory	Ju Junit	Jm _{JMeter}	Tn TestNG	Rd RapidDeploy	Су	Oc Octopus Deploy	No CA Nolio	Eb ElasticBox	Ad Apprenda
Cs Cassandra	HX Helix	Msb MSBuild	Rk Rake	Lb LuntBuild	Cu Continuum	Ca Continua Cl	Gu Gump	Ng NuGet	Ap Appium	XItv XL TestView	Тс		Ef	XId XL Deploy	1.1.1	Mo Mesos	Cf Cloud
			91 En	92 En	93 En	94 En	95 En	96 Pd	97 En	98 En	99 Fm	100 Pd	101 Fm	102 Fm	103 Fm	104 Pd	105 En
			XIr XL Release	Ur UrbanCode Release	LS CA Service Virtualization	BMC Release	Hp HP Codar	Ex Excel	Pl Plutora Release	Sr Serena Release	Tr Trello	Jr Jira	Rf HipChat	SI Slack	Fd Flowdock	Pv Pivotal Tracker	Sn ServiceNow
			Ki Kibana	Nr New Relic	Ni Nagios	Gg Ganglia	Ct Cacti	Gr Graphite	IC Icinga	Sp Splunk	SI Sumo Logic	LS Logstash	Lg Loggly	Gr Graylog	Sn Snort	Tr Tripwire	Cy CyberArk



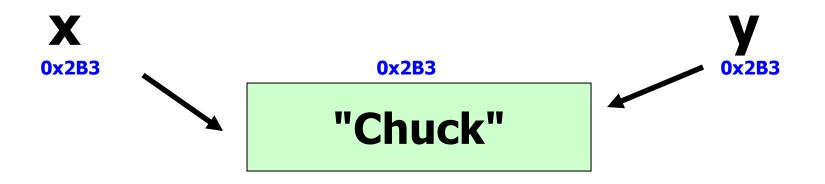
What is a reference of the second sec

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

The variable stores the memory address of the actual Object.

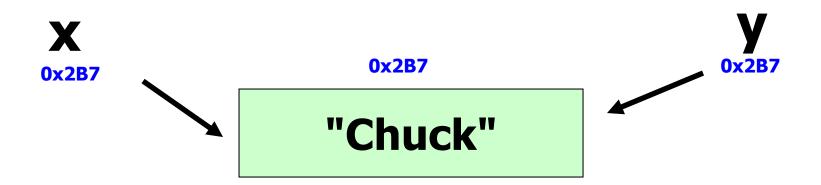
String x = new String("Chuck"); String y = x;

x and y store the same memory address.



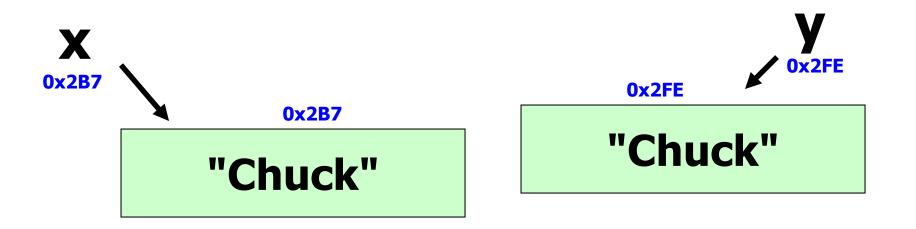
```
String x = "Chuck";
String y = "Chuck";
```

x and y store the same memory address.

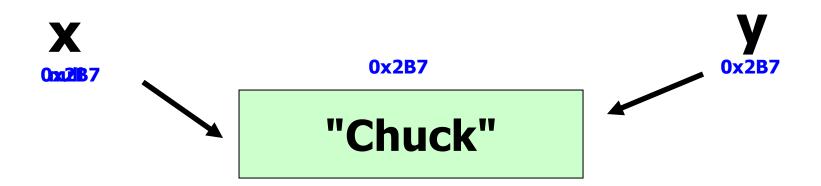


```
String x = new String("Chuck");
String y = new String("Chuck");
```

x and y store different memory addresses.



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



open references.java

What is a was a state of the st

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

window.setColor(Color.red);

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

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

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

method call with 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 simply means that a copy of the original is being sent to the method.

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.

```
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);
```

```
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.

```
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);
```

```
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

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

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

//test code
```

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

```
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.

```
public static void changeTwo(int[] ray)
 ray = new int[5];
 ray[0]=2;
 out.println(Arrays.toString(ray));
                             [2, 0, 0, 0, 0]
//test code
                             [5, 4, 3, 2, 1]
int[] nums = {5,4,3,2,1};
changeTwo(nums);
out.println(Arrays.toString(nums));
```

```
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

```
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 \text{ test} = \text{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);
```

OUTPUTstuff something stuff x was changed

```
Passing
class A{
 private String x;
 public A( String val ){
  x = val;
                                         by Value
 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();
                 x-stuff
                               x-xowæsthhænged
```

```
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 \text{ test} = \text{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

```
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

Array of References

Array of References

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

0 1 2 3 4 5 6 7 ...



```
list[3] = "fred";
0 1 2 3 4 5 6 7 ...
null null 0x7 null null null null
```

"fred"

upen arravofreferencesone.java

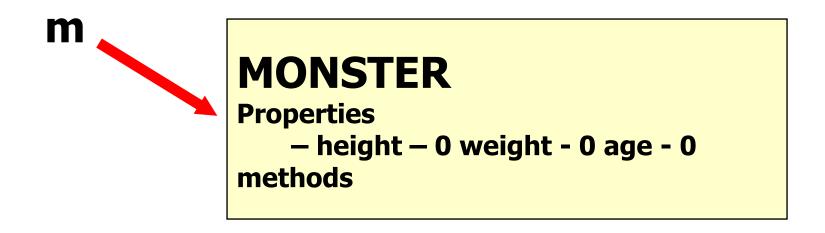
Array of Monster References

class Monster

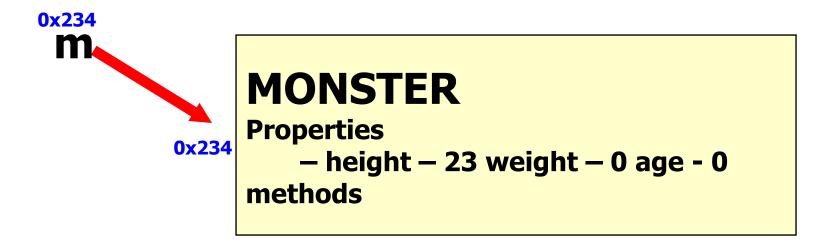
class Monster{

```
// 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 \}
```

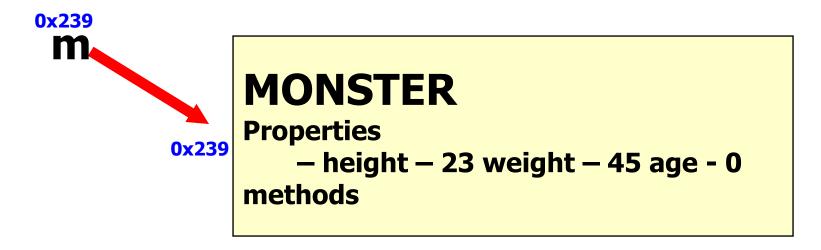
Monster m = new Monster();



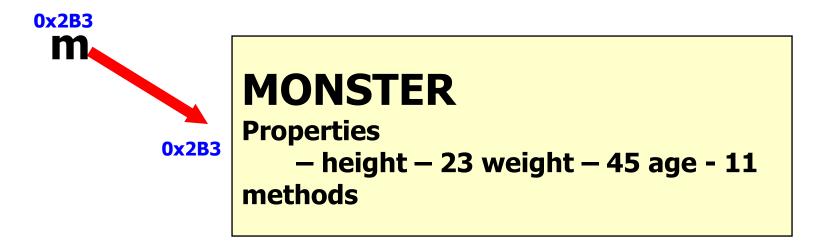
Monster m = new Monster(23);



Monster m = new Monster(23, 45);



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



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]);

<u>OUTPUT</u>

null null

null

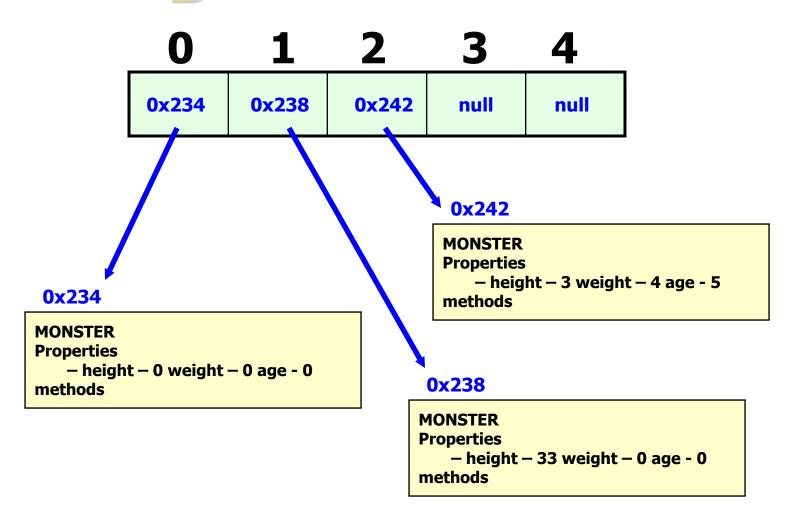
null

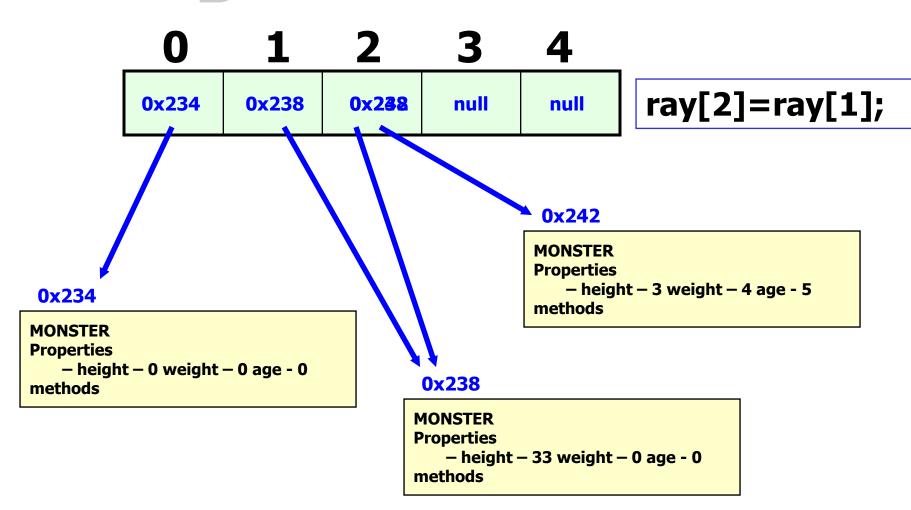
null

```
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





Open

arrayofreferencestwo.java

```
public class Creature implements Comparable
 //data and constructors now shown
 public void setSize(int girth){
   size=girth;
 //toString not shown
```

```
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]);

creatures[0].setSize(7);

What does this store?

0x242

What does the dot do?

0x242

Creature

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

Open

arrayofreferencesthree.java

Continue work on Lah Uz