Java 8 OCA Webinar

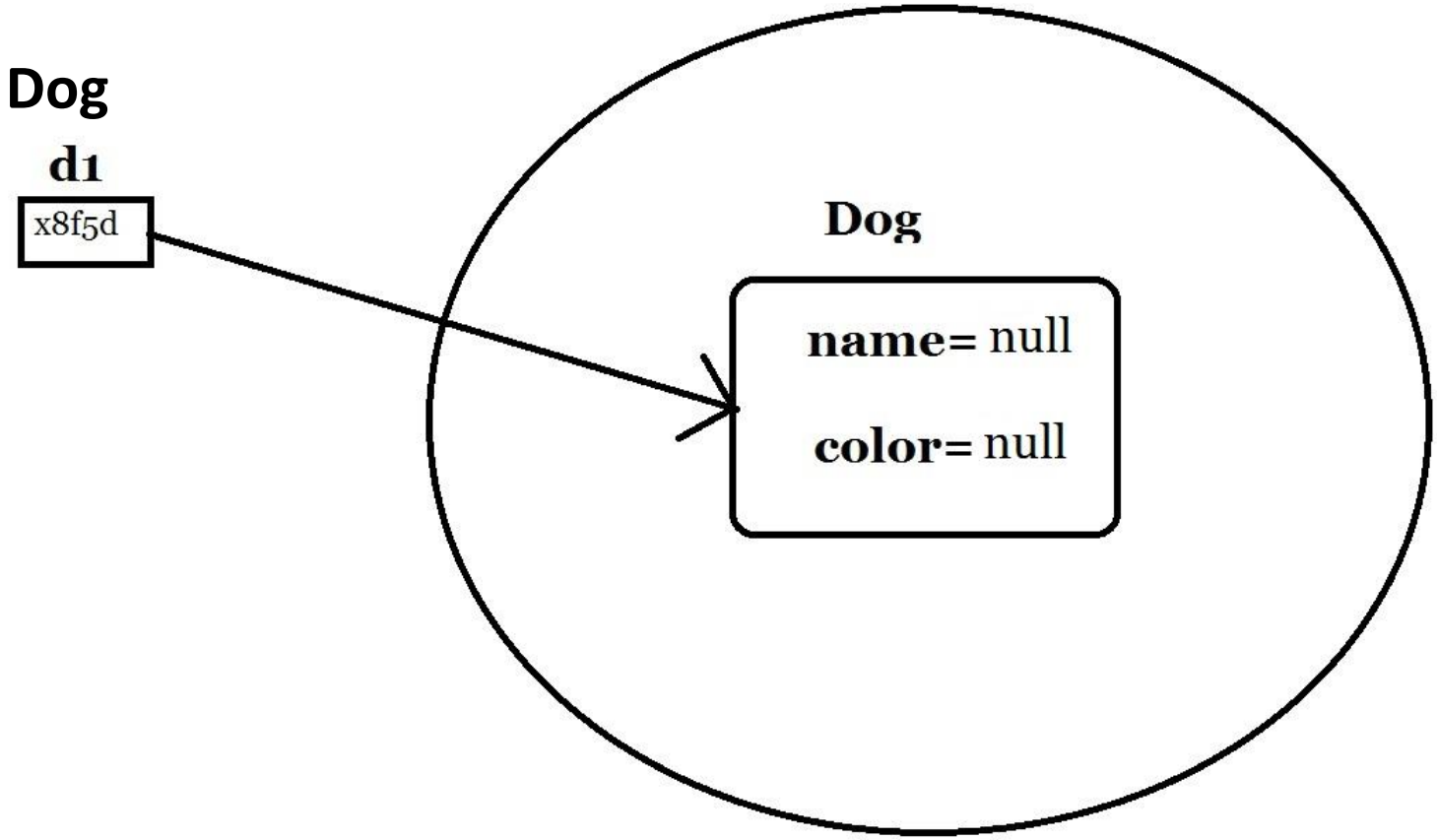
- A Taste of Java Classes
- A Taste of Java Collections
- A Taste of Java Lambdas

Classes and Objects

```
class Dog
{
    String name;
    String color;
}
```

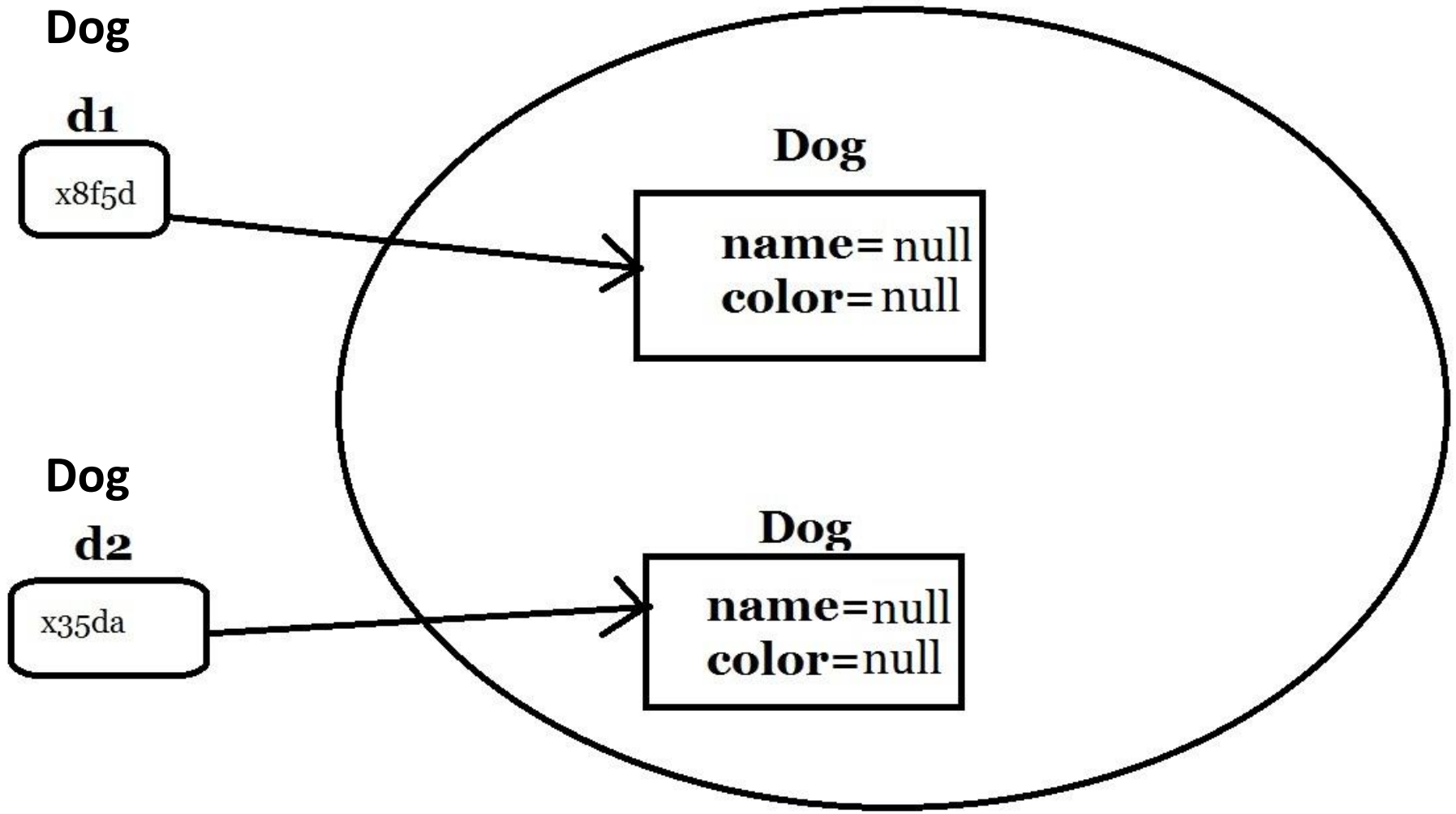
```
class Dog
{
    String name;
    String color;
}
```

```
Dog d1 = new Dog();
```


Dog
d1
x8f5d
Dog
name= null
color= null

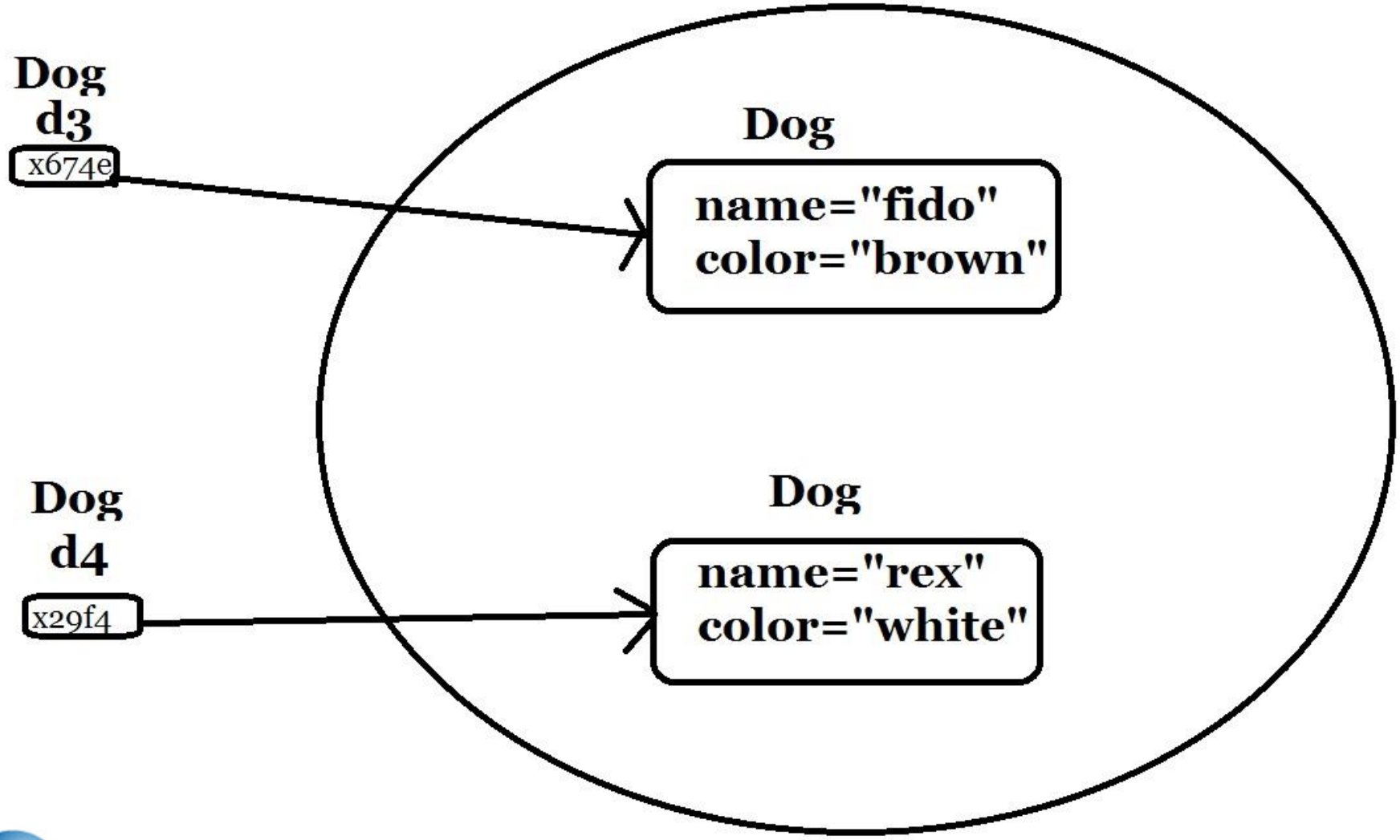
```
class Dog
        {
        String name;
        String color;
        }
```

```
Dog d1 = new Dog();
Dog d2 = new Dog();
```


Dog
d1
x8f5d
Dog
name= null
color= null
Dog
d2
x35da
Dog
name=null
color=null

```
class Dog
        {
        String name;
        String color;
        Dog(String n, String c)
                {
                name=n;
                color=c;
                }
        }
```

```
Dog d3 = new Dog("fido","brown");
Dog d4 = new Dog ("rex","white");
```


Dog
d3
x674e
Dog
name="fido"
color="brown"
Dog
d4
x29f4
Dog
name="rex"
color="white"

```
class Dog
        {
        String name;
        String color;
        Dog(String n, String c)
                {
                name=n;
                color=c;
                }
        String getName() { return name;}
        String getColor()  { return color;}
        }
```

System.out.println(“d3 name=“+d3.getName());

d3 name = fido

System.out.println(“d3 color=“+d3.getColor());

d3 color = brown

System.out.println(“d4 name=“+d4.getName());

d4 name = rex

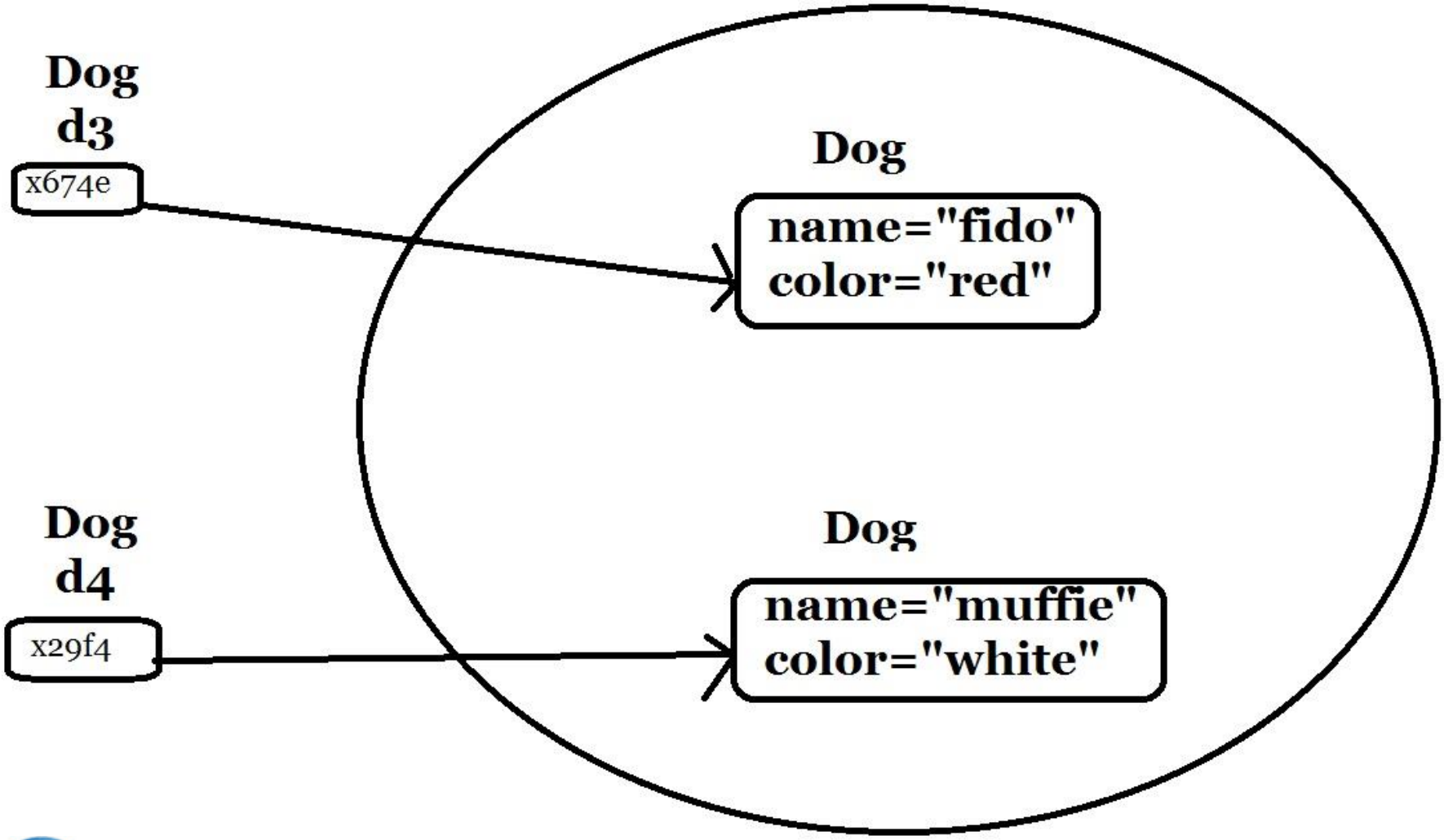
System.out.println(“d4 color=“+d4.getColor());

d4 color = white

```
class Dog {
        String name;
        String color;
        Dog(String n, String c){
                name=n;
                color=c;
                }
        String getName() { return name;}
        String getColor()  { return color;}
        void setName(String n){name=n;}
        void setColor(String c){color=c;}
        }
```

d3.setColor(“red”);

d4.setName(“muffie”);


Dog
d3
x674e
Dog
name="fido"
color="red"
Dog
d4
x29f4
Dog
name="muffie"
color="white"

System.out.println("d3 name="+d3.getName());

d3 name = fido

System.out.println("d3 color="+d3.getColor());

d3 color = red

System.out.println("d4 name="+d4.getName());

d4 name = muffie

System.out.println("d4 color="+d4.getColor());

d4 color = white

Collections

Collections

There are a few basic operations you'll normally do with collections:

- Add objects to the collection.
- Remove objects from the collection.
- Find out if an object (or group of objects) is in the collection.
- Retrieve an object from the collection (without removing it).
- Iterate through the collection, looking at each element (object) one after another.

Collections

Collections come in four basic flavors:

- **Lists** *Lists* of things (classes that implement List).
- **Sets** *Unique* things (classes that implement Set).
- **Maps** Things with a *unique* ID (classes that implement Map).
- **Queues** Things arranged by the order in which they are to be processed.

Collections

There are four basic sub-flavors:

- **Unordered**
- **Ordered**
- **Unsorted**
- **Sorted**

ArrayList

```
import java.util.*;
public class TestArrayList {
public static void main(String[] args) {
List<String> test = new ArrayList<String>();
String s = "hi";
test.add("string");
test.add(s);
test.add(s+s);
System.out.println(test.size());
System.out.println(test.contains(42));
System.out.println(test.contains("hihi"));
test.remove("hi");
System.out.println(test.size());
} }
```

output:

```
3
false
true
2
```

Using Sets

```
import java.util.*;
class SetTest {
public static void main(String[] args) {
boolean[] ba = new boolean[5];
Set s = new HashSet();
ba[0] = s.add("a");
ba[1] = s.add(new Integer(42));
ba[2] = s.add("b");
ba[3] = s.add("a");
ba[4] = s.add(new Object());
for(int x=0; x<ba.length; x++) System.out.print(ba[x] + " ");
System.out.println("\n");
for(Object o : s) System.out.print(o + " ");
}
}
```

Output produced:

```
true true true false true  // the fourth invocation of add() failed, because it attempted to insert a
      duplicate entry
a java.lang.Object@e09713 42 b // the order of objects printed in the second for loop is not predictable
```

Using Maps

```
class MapTest {
public static void main(String[] args) {
Map<Object, Object> m = new HashMap<Object, Object>();
m.put("k1", new Dog("aiko")); // add some key/value pairs
m.put("k2", Pets.DOG);
m.put(Pets.CAT, "CAT key");
Dog d1 = new Dog("clover"); // let's keep this reference
m.put(d1, "Dog key");
m.put(new Cat(), "Cat key");
System.out.println(m.get("k1")); // #1
String k2 = "k2";
System.out.println(m.get(k2)); // #2
Pets p = Pets.CAT;
System.out.println(m.get(p)); // #3
System.out.println(m.get(d1)); // #4
System.out.println(m.get(new Cat())); // #5
System.out.println(m.size()); // #6
}}
```

- **Which produces**:

```
Dog@1c
DOG
CAT key
Dog key
null
5
```

Using the PriorityQueue Class

```
public static void main(String[] args) {
int[] ia = {1,5,3,7,6,9,8 }; // unordered data
PriorityQueue<Integer> pq1 = new
    PriorityQueue<Integer>(); // use natural order
// load queue
for(int x : ia) pq1.offer(x);
// review queue
for(int x : ia) System.out.print(pq1.poll() + " ");
```

Output produced:

1 3 5 6 7 8 9

Lambdas

Lambdas

- Lambdas facilitate the programming of ***behavior-as-data***.
- This is the idea of ***'passing behavior'*** to methods and variables.
- We accomplish this by using ***'lambda expressions'.***
- A ***'lambda expression'*** is a block of code than can be passed around.

Lambdas

- Lambdas are made up of 3 parts:

- 1 - paramater declaration
- 2 – the new Java 8 operator: ->
- 3 – the lambda body which defines behavior and is enclosed in curly braces.

Lambdas

- (Object o) -> { System.out.println(o); }

Lambdas

```
@FunctionalInterface
interface Printer  { void print(Object o); }

public class LambdaDemo {
    public static void main(String [] args)    {
    //the following line of code will not execute the lambda expression.
    //it is merely assigning it to a variable for later use.
    Printer  p = (Object o) -> {System.out.println(o);};

    //here we actually execute the lambda expression
    p.print("Hello World!");                }    }
```

Lambdas

An illustration of the power of lambdas in reducing the amount of code needed to perform tasks

Lambdas

```
public class DogClass1 {
String name;
String color;

DogClass1(){
name="";
color="";    }

DogClass1(String n, String c)  {
name=n;
color=c;    }

public String getColor(){return color;}
public String getName(){return name;}
public void setColor(String c){color=c;}
public void setName(String n){name=n;}          }
```

```
public class ld001 {

static boolean isDogRed(DogClass1 d)
{
return (d.color.equals("red"));
}

public static void main(String [] args)
{
DogClass1 d1 = new DogClass1("fido","brown");
DogClass1 d2 = new DogClass1("watson","red");
DogClass1 d3 = new DogClass1("milly","white");
DogClass1 d4 = new DogClass1("jackson","black");

System.out.println("is the Dog d1 color red:"+isDogRed(d1));
}

}
```

```
public class ld002 {

static boolean isDogDesiredColor(DogClass1 d, String c)
{
return (d.color.equals(c));
}

public static void main(String [] args)
{
DogClass1 d1 = new DogClass1("fido","brown");
DogClass1 d2 = new DogClass1("watson","red");
DogClass1 d3 = new DogClass1("milly","white");
DogClass1 d4 = new DogClass1("jackson","black");

System.out.println("is the Dog d1 color the desired color red:"+
isDogDesiredColor(d1,"red"));
System.out.println("is the Dog d2 color the desired color red:"+
isDogDesiredColor(d2,"red"));
System.out.println("is the Dog d3 color the desired color green:"+
isDogDesiredColor(d3,"green"));
}


}
```

```
public class ld003 {

static boolean isDogRover(DogClass1 d)
{
return (d.name.equals("rover"));
}

public static void main(String [] args)
{
DogClass1 d1 = new DogClass1("fido","brown");
DogClass1 d2 = new DogClass1("watson","red");
DogClass1 d3 = new DogClass1("milly","white");
DogClass1 d4 = new DogClass1("jackson","black");

System.out.println("is the Dog d1 named rover:"+isDogRover(d1));
}


}
```

```
public class ld004 {

static boolean isDogDesiredName(DogClass1 d, String n)
{
return (d.name.equals(n));
}

public static void main(String [] args)
{
DogClass1 d1 = new DogClass1("fido","brown");
DogClass1 d2 = new DogClass1("watson","red");
DogClass1 d3 = new DogClass1("milly","white");
DogClass1 d4 = new DogClass1("jackson","black");

System.out.println("is the Dog d1 name the desired name 'rover':"+
isDogDesiredName(d1,"rover"));
System.out.println("is the Dog d2 name the desired name 'rover':"+
isDogDesiredName(d2,"rover"));
System.out.println("is the Dog d3 name the desired name 'milly':"+
isDogDesiredName(d3,"milly"));
}
}
```

```
public class ld005 {

        static boolean isDogDesiredColor(DogClass1 d, String c)
        {
                return (d.color.equals(c));
        }

        static boolean isDogDesiredName(DogClass1 d, String n)
        {
                return (d.name.equals(n));
        }


        public static void main(String [] args)
        {
                DogClass1 d1 = new DogClass1("fido","brown");
                DogClass1 d2 = new DogClass1("watson","red");
                DogClass1 d3 = new DogClass1("milly","white");
                DogClass1 d4 = new DogClass1("jackson","black");

                System.out.println("is the Dog d1 color the desired color red:"+isDogDesiredColor(d1,"red"));
                System.out.println("is the Dog d2 color the desired color red:"+isDogDesiredColor(d2,"red"));
                System.out.println("is the Dog d3 color the desired color green:"+isDogDesiredColor(d3,"green"));

                System.out.println("is the Dog d1 name the desired name 'rover':"+isDogDesiredName(d1,"rover"));
                System.out.println("is the Dog d2 name the desired name 'rover':"+isDogDesiredName(d2,"rover"));
                System.out.println("is the Dog d3 name the desired name 'milly':"+isDogDesiredName(d3,"milly"));
        }
}
```

```
public class ld006 {
        static boolean isDogDesiredColor(DogClass1 d, String c)
        {
                return (d.color.equals(c));
        }

        static boolean isDogDesiredName(DogClass1 d, String n)
        {
                return (d.name.equals(n));
        }

        static boolean isDogDesiredNameAndDesiredColor(DogClass1 d,String n, String c)
        {
                return (d.name.equals(n))&&(d.color.equals(c));
        }

        public static void main(String [] args)
        {
                DogClass1 d1 = new DogClass1("fido","brown");
                DogClass1 d2 = new DogClass1("watson","red");
                DogClass1 d3 = new DogClass1("milly","white");
                DogClass1 d4 = new DogClass1("jackson","black");

                System.out.println("is Dog d4 color 'black':"+isDogDesiredColor(d4,"black"));
                System.out.println("is Dog d4 named 'jackson':"+isDogDesiredName(d4,"jackson"));
                System.out.println("is Dog d4 the desired name 'jackson' and desired color 'black':"+
                   isDogDesiredNameAndDesiredColor(d4,"jackson","black"));
        }
}
```

```
public class DogClass2 {
   String name;
   String color;
   int age;

DogClass2()   {
name="";
color="";
age=0;
}

DogClass2(String n, String c, int a)   {
name=n;
color=c;
age=a;
}
public String getColor(){return color;}
public String getName(){return name;}
public void setColor(String c){color=c;}
public void setName(String n){name=n;}
public int getAge(){return age;}
public void setAge(int a){age=a;}
}
```

```
public class ld007 {
static boolean isDogDesiredColor(DogClass2 d, String c)   {
                    return (d.color.equals(c));   }

static boolean isDogDesiredName(DogClass2 d, String n)   {
                    return (d.name.equals(n));   }

static boolean isDogDesiredAge(DogClass2 d, int a)   {
                    return (d.age==a);   }

static boolean isDogDesiredNameAndDesiredColor(DogClass2 d,String n, String c)   {
                    return (d.name.equals(n))&&(d.color.equals(c));   }

static boolean isDogDesiredNameAndDesiredAge(DogClass2 d,String n, int a)   {
                    return (d.name.equals(n))&&(d.age==a);   }

static boolean isDogDesiredColorAndDesiredAge(DogClass2 d,String n, int a)   {
                    return (d.color.equals(n))&&(d.age==a);   }

static boolean isDogDesiredNameAndDesiredColorAndDesiredAge(
          DogClass2 d,String n, String c, int a)   {
                    return (d.name.equals(n))&&(d.color.equals(c))&&(d.age==a);   }
```

```
@FunctionalInterface
interface Tester<T>{
        boolean test(T a);
}
public class ld008 {
static boolean performCheck(DogClass2  d, Tester<DogClass2> t)   {
                return t.test(d);   }
public static void main(String [] args)   {
        DogClass2 d1 = new DogClass2("fido","brown",1);
        DogClass2 d2 = new DogClass2("watson","red",2);
        DogClass2 d3 = new DogClass2("milly","white",3);
        DogClass2 d4 = new DogClass2("jackson","black",4);

System.out.println("is Dog d4 the desired color 'black':"+
performCheck(d4, d -> d.color.equals("black"))  );
System.out.println("is Dog d4 the desired name 'jackson' and desired color 'black':"+
performCheck(d4, d -> d.name.equals("jackson") && d.color.equals("black"))  );
System.out.println("is Dog d2 the desired name 'watson' and desired color 'red' and desired age
'2':"+
performCheck(d1, d->d.color.equals("red")&&d.age==2&&d.name.equals(“fido")
) );
        }       }
```