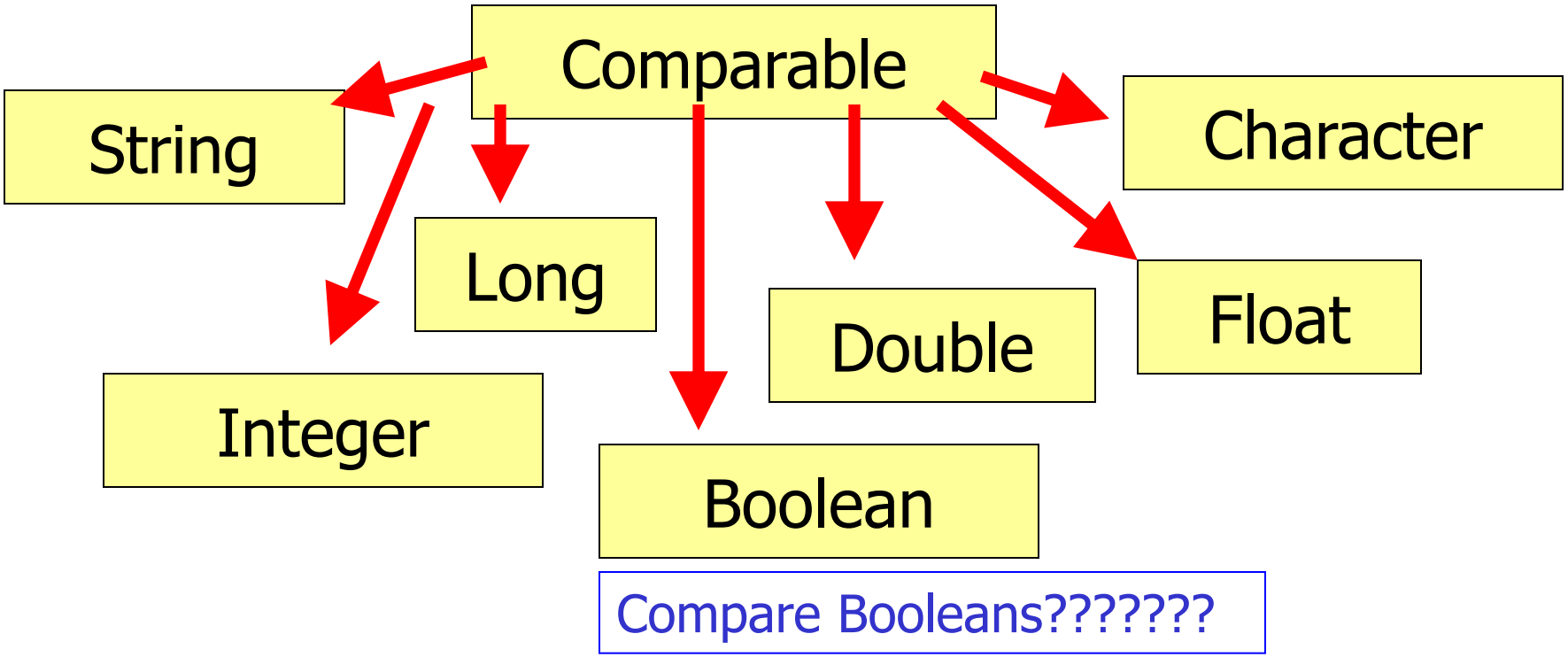
Interfaces

Comparable

The Comparable Interface

Most all of the classes in Java that can be compared implement Comparable.



Interfaces

```
Comparable x = 54;
Comparable y = 67;

out.println(x.compareTo(y));
```

Why is this okay??

OUTPUT

-1

Interfaces

```
Comparable x = 9.21;
Comparable y = 8.54;

out.println(x.compareTo(y));
```

OUTPUT

1

Interfaces

```
Comparable x = "23";
Comparable y = "45";

out.println(x.compareTo(y));
```

OUTPUT

-2

Interfaces

```
Comparable x = “dog";
Comparable y = “hog";

out.println(x.compareTo(y));
```

OUTPUT

-4

Interfaces

```
Comparable x = “dog";
Comparable y = “dig";

out.println(x.compareTo(y));
```

OUTPUT

6

Interfaces

```
Comparable x =
                new Comparable();

out.println(x);
```

Is this okay??

OUTPUT

no output
compile error

Why use an interface?

```
public interface Comparable
{
   int compareTo(Object o);
}
```

ABSTRACT

Lots of unknowns!

No instance variables!
No constructors!
No method implementations!

Open comparableone.java

Open comparabletwo.java

Making the abstract concrete

What is an interface?

```
public interface MyHope
{
  boolean makeAFiveInCompSciAP();
}
```

abstract - an idea of what is wanted

concrete - having enough information to actually make it happen

What is an interface?

```
public interface MyHope
{
  boolean makeAFiveInCompSciAP();
}
```

Do know what I want you to do?
Do I know if you are going to do it?
Do I know how you are going to do it?

What is an interface?

```
public class Student implements MyHope
{
  //instance variables and constructors not shown

  boolean makeAFiveInCompSciAP(){
    //implementation now shown
  }
}
```

Now the abstract becomes concrete.
More is now known.

What is an interface?

```
public interface Comparable
{
   int compareTo(Object o);
}
```

abstract - an idea of what to do

concrete - having enough information to actually do it

Writing compareTo()

```
public class Creature implements Comparable
{
  private int size;

  public Creature(int girth) { size=girth; }

  public int compareTo(Object obj)
  {
    Creature other = (Creature)obj;
    if(size>other.size)
      return 1;
    else if(size<other.size)
      return -1;
    return 0;
  }

  public String toString() { return "" + size; }
}
```

The abstract becomes concrete.

Writing compareTo()

```
public class Word implements Comparable<Word>
{
  private String orig;

  public Word(String s) { orig = s; };

  public int compareTo(Word other)
  {

     //must add code to complete

     return 1;
  }
  public String toString() { return orig; }
}
```

Because Word implements Comparable, Java knows that Word will have a compareTo() method.

Open comparablethree.java

Open sort.java

How Java uses Comparable

Why use an interface?

If an entire hierarchy of classes implements the same interface, you can write very generic code to manipulate any of those classes.

Interfaces

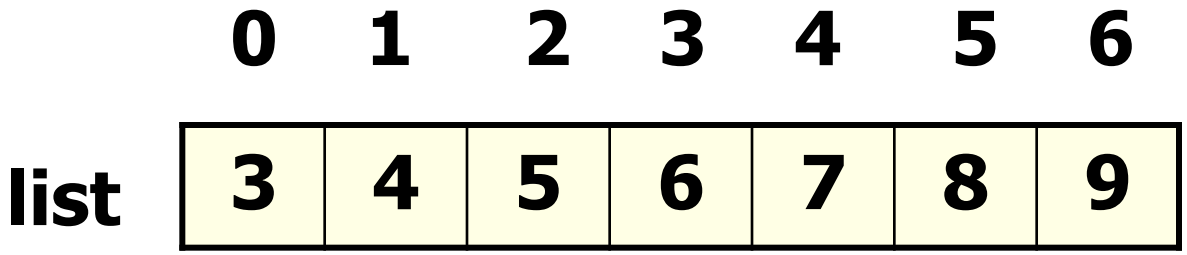
```
Comparable[] list =
                new Comparable[25];

//load with Comparables

Arrays.sort(list);
```

Interfaces

Arrays.sort() will use the compareTo() method of each object when sorting the array.

Interfaces

```
Comparable[] list =  {3,8,7,6,5,4,9};
Arrays.sort(list);
for(Comparable num : list)
{
  out.println(num);
}
```



OUTPUT

```
3
4
5
6
7
8
9
```

Open sorttwo.java

Interfaces

```
public void sort(Comparable[] stuff)
{
   for(int i=0;i<stuff.length-1;i++)
   {
    int spot=i;
    for(int j=i;j<stuff.length;j++){
     if(stuff[j].compareTo(stuff[spot])>0)
       spot=j;
    }
    Comparable save=stuff[i];
    stuff[i]=stuff[spot];
    stuff[spot]=save;
   }
}
```

Why would this method be passed an array of Comparable?

Does this demonstrate the power of interfaces and hierarchies?

Open sortthree.java

More Interfaces

Interfaces

```
public interface Exampleable
{
   int writeIt(Object o);
   int x = 123;
}
```

Methods are public abstract!
Variables are public static final!

Interfaces

```
public interface Exampleable
{
   public abstract int writeIt(Object o);
   public static final int x = 123;
}
```

Methods are public abstract!
Variables are public static final!

Interfaces

An interface is a list of methods that must be implemented.

An interface may not contain any implemented methods.

Interfaces cannot have constructors!!!

Interfaces

Interfaces are typically used when you know what you want an Object to do, but do not know what will be used to get it done.

If only the behavior is known, use an interface.

Interfaces

```
public interface Locatable
{
   public int getX();
   public int getY();
}
```

Interfaces

```
public interface Movable
{
   public void setPos( int x, int y);
   public void setX( int x );
   public void setY( int y );
}
```

Interfaces

```
class Ship implements Locatable, Movable
{
        private int xPos, yPos;

        //how many methods must
        //be implemented?
}
```

Interfaces

class A implements B { }

A class can implement multiple interfaces.

class A implements B,C { } //legal

Interfaces

Interfaces are true abstract classes. All methods listed in an interface are abstract; as a result, you must implement every method.

Open interface.java