Filtering Data

Interfaces in More Depth

How Does Java Know?

```
public ArrayList<QuakeEntry>
filter(ArrayList<QuakeEntry> quakeData,
       Filter f) {
  ArrayList<QuakeEntry> answer = new ArrayList<QuakeEntry>();
  for(QuakeEntry qe : quakeData) {
    if (f.satisfies(qe)) {
      answer.add(qe);
    }
  }
  return answer;
}
```

How does Java know which .satisfies to call?

Java Remembers Actual Types

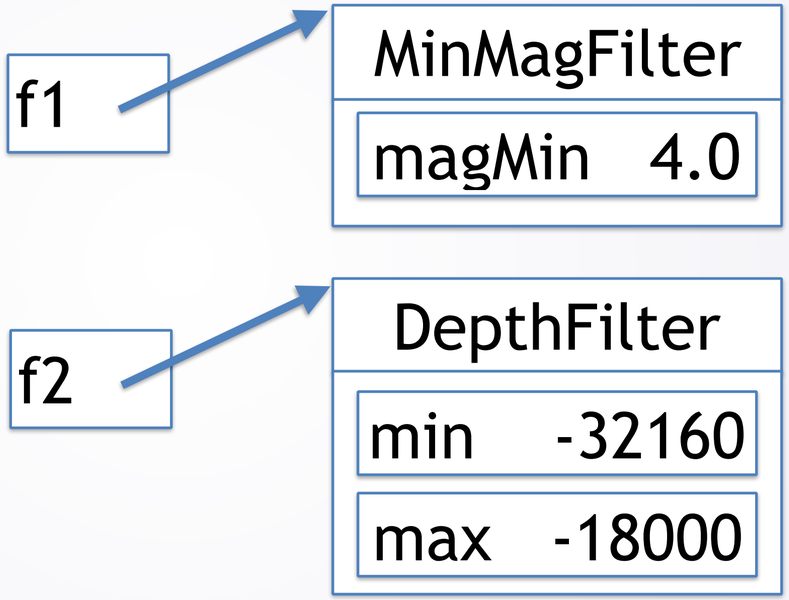
```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
     ……
f1.satisifies(someQuake);
     ……
f2.satisifies(someQuake);
```

Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ......
f1.satisifies(someQuake);
      ......
f2.satisifies(someQuake);
```

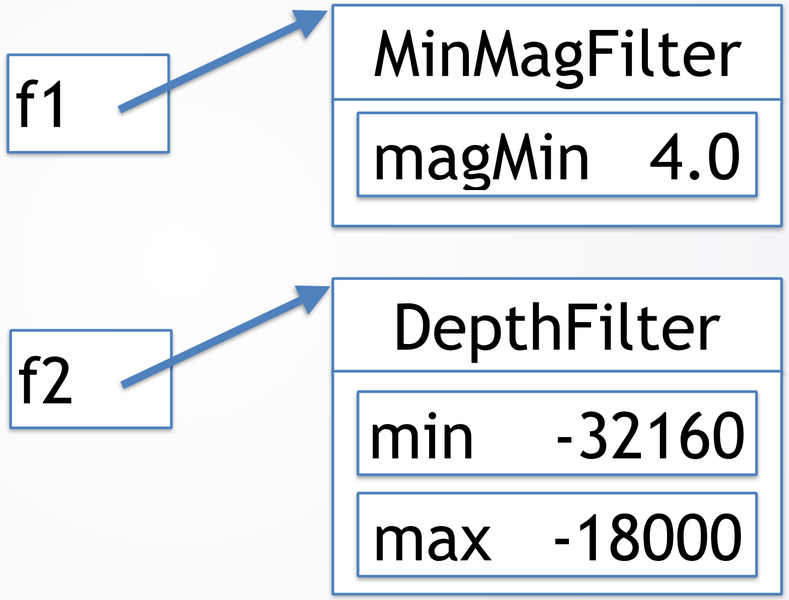
Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ……
f1.satisifies(someQuake);
      ……
f2.satisifies(someQuake);
```



Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ......
f1.satisifies(someQuake);
      ......
f2.satisifies(someQuake);
```



Java Remembers Actual Types

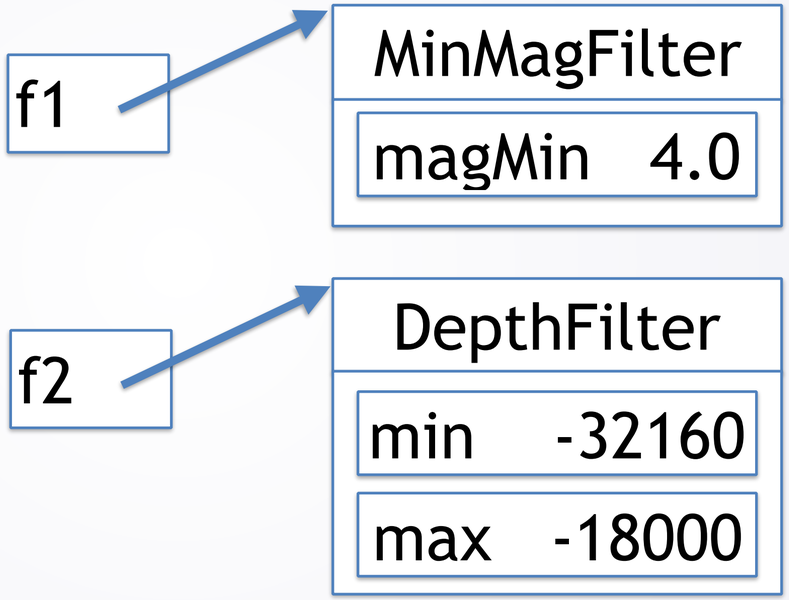
```
public class MinMagFilter implements Filter {
    private double magMin;
    public MinMagFilter(double min) {
        magMin = min;
    }
    public boolean satisfies(QuakeEntry qe) {
        return qe.getMagnitude() >= magMin;
    }
}
```

Java Remembers Actual Types

```
public class MinMagFilter implements Filter {
    private double magMin;
    public MinMagFilter(double min) {
        magMin = min;
    }
    public boolean satisfies(QuakeEntry qe) {
        return qe.getMagnitude() >= magMin;
    }
}
```

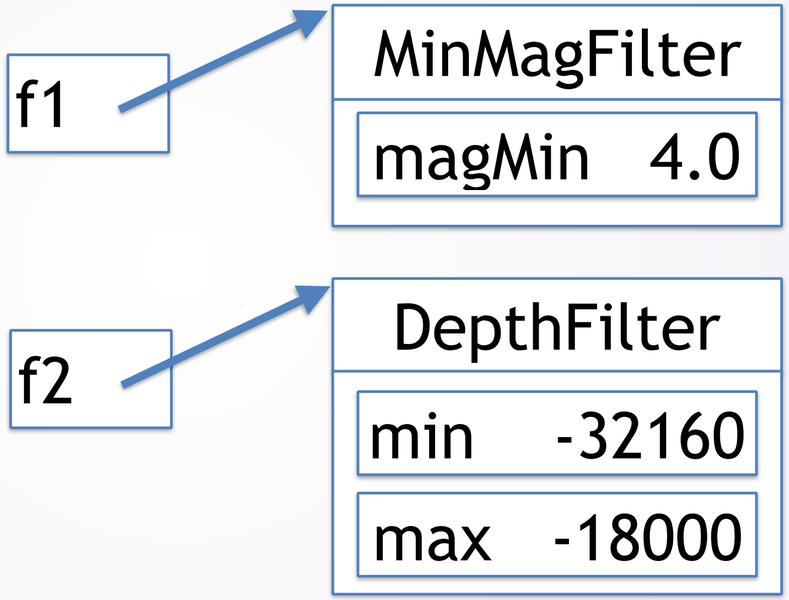
Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ......
f1.satisifies(someQuake);
      ......
f2.satisifies(someQuake);
```



Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ......
f1.satisifies(someQuake);
      ......
f2.satisifies(someQuake);
```



Java Remembers Actual Types

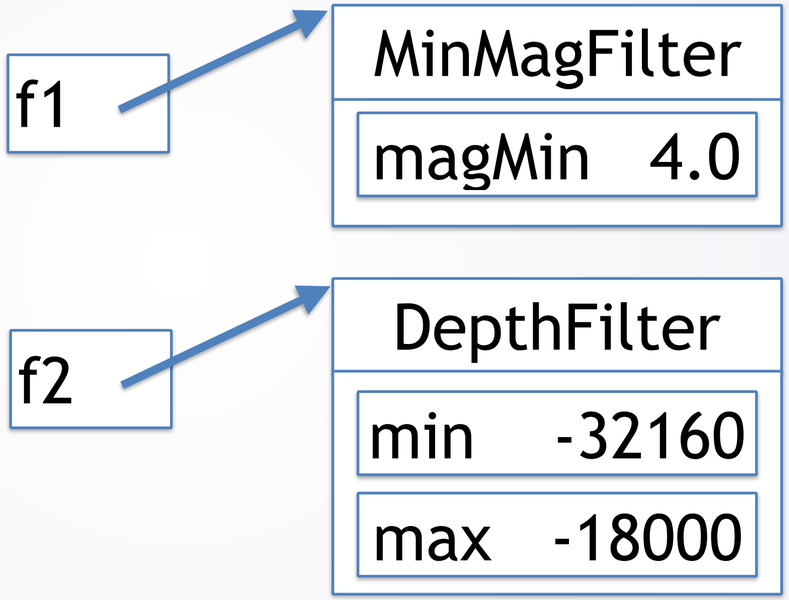
```
public class DepthFilter implements Filter {
    private double min;
    private double max;
    public DepthFilter(double minDepth, double maxDepth) {
        min = minDepth;
        max = maxDepth;
    }
    public boolean satisfies(QuakeEntry qe) {
        return qe.getDepth() >= min &&
               qe.getDepth() <= max;
    }
}
```

Java Remembers Actual Types

```
public class DepthFilter implements Filter {
    private double min;
    private double max;
    public DepthFilter(double minDepth, double maxDepth) {
        min = minDepth;
        max = maxDepth;
    }
    public boolean satisfies(QuakeEntry qe) {
        return qe.getDepth() >= min &&
               qe.getDepth() <= max;
    }
}
```

Java Remembers Actual Types

```
Filter f1 = new MinMagFilter(4.0);
Filter f2 = new DepthFilter(-32160,-18000);
      ......
f1.satisifies(someQuake);
      ......
f2.satisifies(someQuake);
```



Which Method to Call?

- Java remembers actual type
  - When you do `new`
- Uses actual type to determine method to call
  - "Dynamic Dispatch"