More Collections

- Iterators
- Collection Tools
- Maps

See Also

https://docs.oracle.com/javase/tutorial/collections/

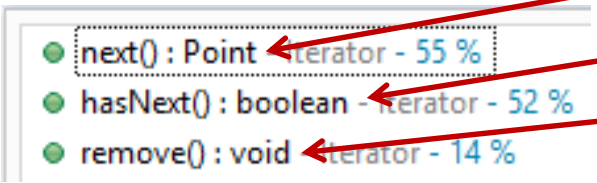
https://docs.oracle.com/javase/tutorial/collections/intro/

http://www.javatpoint.com/collections-in-java

Iterators

- Stepping through a list from start to finish is common.
- An Iterator is an object that walks the collection on your behalf.
- Each collection has its own Iterator implementation optimized for the collection's internal data.

```
List<Point> points = new LinkedList<Point>();

points.add(new Point(1,2));
points.add(new Point(3,4));
points.add(new Point(5,6));

Iterator<Point> i = points.iterator();
i.
```



Get next object in the collection

"true" if there is a next object

Remove the last object returned by "next()"

Using Iterators

```
List<Point> points = new LinkedList<Point>();

points.add(new Point(1,2));
points.add(new Point(3,4));
points.add(new Point(5,6));

Iterator<Point> i = points.iterator();
Point a = i.next();
System.out.println(a);
System.out.println(i.next());

for(Iterator<Point> j = points.iterator();j.hasNext();) {
    Point b = j.next();
    System.out.println(b);
}

for(Point b : points) {
    System.out.println(b);
}
```

Fetch the next object

Very common, but tedious

The compiler knows about iterators and will write the code for you

Collections

- The "Collections" class has many static methods for searching and sorting.
- Exactly like "Arrays" but for collection objects

```
List<String> strs = new ArrayList<String>();
strs.add("Joe");
strs.add("Betty");
strs.add("Andrew");

System.out.println(strs);

Collections.sort(strs);

System.out.println(strs);
```

```
[Joe, Betty, Andrew]
[Andrew, Betty, Joe]
```

- "String" has a "compareTo(String other)" method that compares one string to another

compareTo(other)

- Common interface for comparing two objects
- Return 1 if "this" is greater, -1 if less, 0 if the same

```java
class Point implements Comparable<Point> {
    int x,y;

    public Point(int i, int j) ...

    @Override
    public String toString() ...

    @Override
    public int compareTo(Point other) {
        if(x > other.x) return 1;  // "this" is greater
        if(x < other.x) return -1; // "this" is less
        return 0; // "this" is the same

    }

}
```

```java
Point a = new Point(1,2);
Point b = new Point(3,4);

System.out.println(a.compareTo(b)); // -1

System.out.println(b.compareTo(a)); // 1

System.out.println(a.compareTo(a)); // 0
```

compareTo(other)

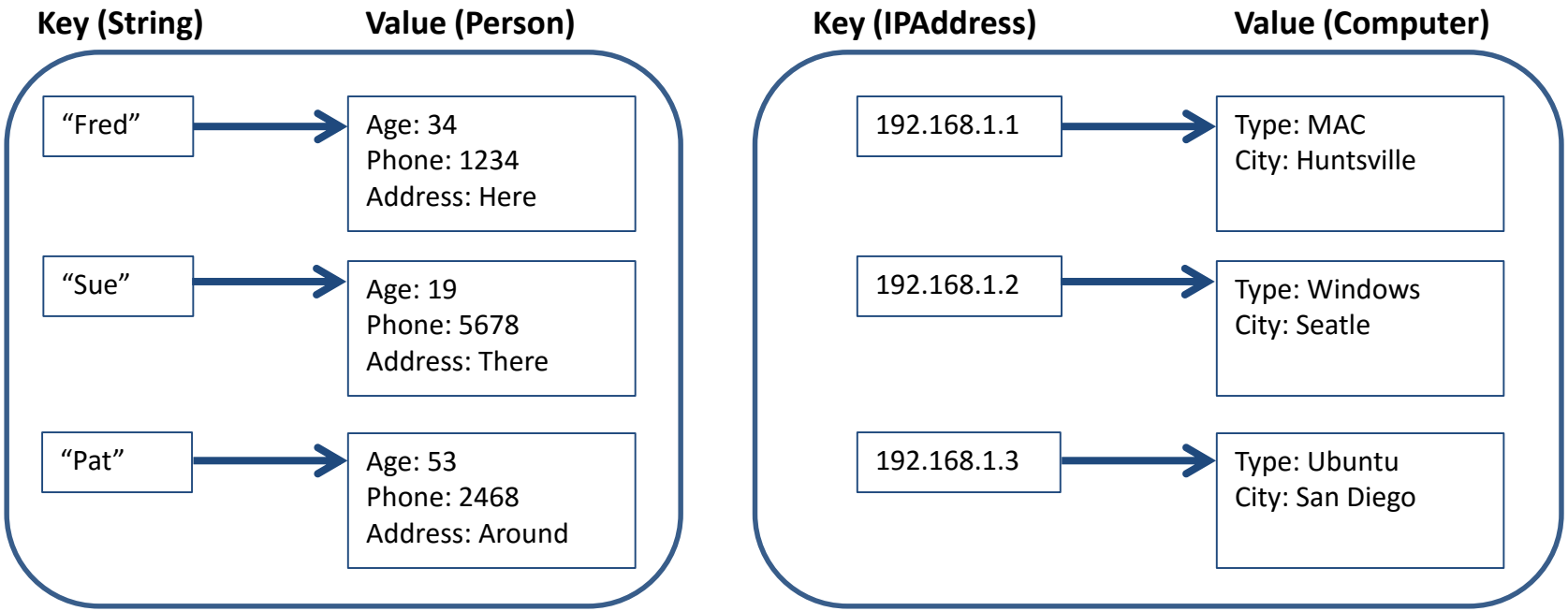
- The Collections.sort calls the "compareTo" on the objects to do its magic.

```
List<Point> points = new ArrayList<Point>();
points.add(new Point(4,1));
points.add(new Point(1,3));
points.add(new Point(9,2));

System.out.println(points);

Collections.sort(points);

System.out.println(points);
```

```
[4,1, 1,3, 9,2]
[1,3, 4,1, 9,2]
```

Maps

- A Map is an association between keys and values.
- You tell the map a value to associate with a key.
- Later you show the map a key and it tells you the value associated with it.



HashMap

- Uses the key's hashcode for quick searching.
- Returns a null pointer if not found, but you can also have null values in the map.

```
Map<String, Point> locs = new HashMap<String, Point>();

locs.put("Home", new Point(0,0));
locs.put("Work", new Point(1,1));
locs.put("Arcade", new Point(2,5));
locs.put("Blackhole", null);

Point a = locs.get("Home");
System.out.println(a);

a = locs.get("Arcade");
System.out.println(a);

a = locs.get("Qdoba");
System.out.println(a);

a = locs.get("Blackhole");
System.out.println(a);
```

- "put" to associate key/value
- Keys must be unique
- "get" to find value for key

```
0,0
2,5
null
null
```

Map Iterators

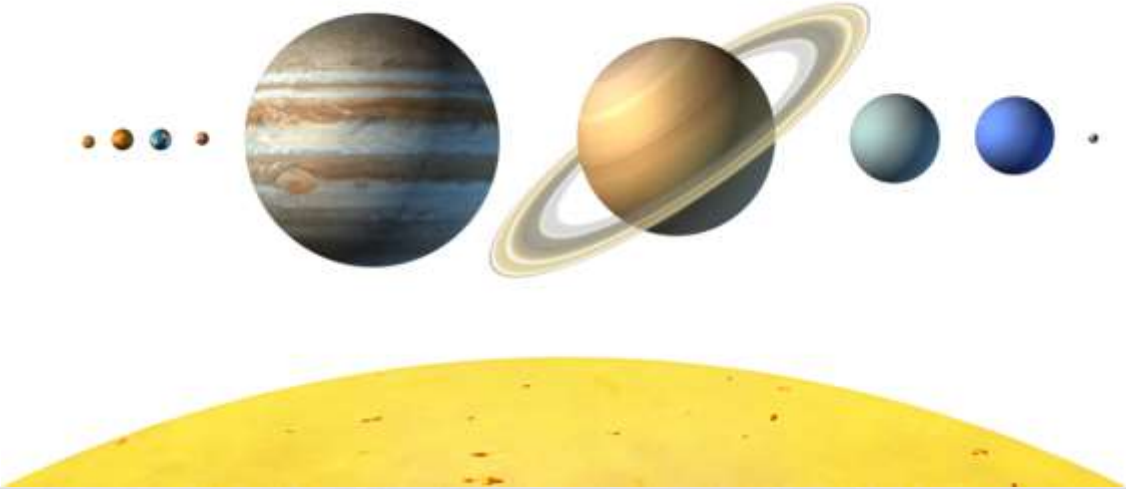
```
Map<String, Point> locs = new HashMap<String, Point>();

locs.put("Home", new Point(0,0));
locs.put("Work", new Point(1,1));
locs.put("Arcade", new Point(2,5));
locs.put("Blackhole", null);

if(locs.containsKey("Qdoba")) {
    System.out.println("Let's Eat!");
} else {
    System.out.println("Not Hungry.");
}

Set<String> keys = locs.keySet();
System.out.println(keys);

for(Entry<String, Point> e : locs.entrySet()) {
    System.out.println("Key:"+e.getKey()+
            " Value:"+e.getValue());
}
```

```
Not Hungry.
[Home, Blackhole, Arcade, Work]
Key:Home Value:0,0
Key:Blackhole Value:null
Key:Arcade Value:2,5
Key:Work Value:1,1
```

Ad infinitum

- You can make very complicated composite collections.

```
List<Map<String, Point>> system =
        new ArrayList<Map<String, Point>>();


Map<String,List<Map<String, Point>>> galaxy =
        new HashMap<String,List<Map<String, Point>>>();
```



Tinkering

- Sort your list of Points based on X then Y coordinates.
- Give a name to each Point and put it in a HashMap.
- Iterate over the Map and print the key/values.