

# Counting Website Visits

HashMap for Unique IPs

# Unique IPs: Another Approach

`{157.55.39.203=1, 152.3.135.44=3, 152.3.135.63=2, 110.76.104.12=1}`

- Printed HashMap using `toString()`
  - Fine for testing, does not look nice

# Unique IPs: Another Approach

`{157.55.39.203=1, 152.3.135.44=3, 152.3.135.63=2, 110.76.104.12=1}`

```
157.55.39.203 visited 1 time  
152.3.135.44 visited 3 times  
152.3.135.63 visited 2 times  
110.76.104.12 visited 1 time
```

- Printed HashMap using `toString()`
  - Fine for testing, does not look nice
  - Suppose you want nice output

# Iterate Over HashMap?

```
public void testCounts(){
    LogAnalyzer la = new LogAnalyzer();
    la.readFile("short-test_log");
    HashMap<String,Integer> counts = la.countVisitsPerIP();
    for (String ip: counts){
        System.out.println(ip + " visited " +
                           counts.get(ip) +
                           " times");
    }
}
```

- Idea: iterate over HashMap?

# Iterate Over HashMap?

```
public void testCounts(){
    LogAnalyzer la = new LogAnalyzer();
    la.readFile("short-test_log");
    HashMap<String,Integer> counts = la.countVisitsPerIP();
    for (String ip: counts){
        System.out.println(ip + " visited " +
                           counts.get(ip) +
                           " times");
    }
}
```

for-each not applicable to expression type  
required: array or java.lang.Iterable; found: java.util.HashMap<java.lang...



- Idea: iterate over HashMap?
  - Can't do it this way



# Iterate Over HashMap?

```
public void testCounts(){
    LogAnalyzer la = new LogAnalyzer();
    la.readFile("short-test_log");
    HashMap<String,Integer> counts = la.countVisitsPerIP();
    for (String ip: counts.keySet()){
        System.out.println(ip + " visited " +
                           counts.get(ip) +
                           " times");
    }
}
```

# Iterate Over HashMap?

```
public void testCounts(){
    LogAnalyzer la = new LogAnalyzer();
    la.readFile("short-test_log");
    HashMap<String,Integer> counts = la.countVisitsPerIP();
    for (String ip: counts.keySet()){
        System.out.println(ip + " visited " +
                           counts.get(ip) +
                           " times");
    }
}
```

- This code works
  - Can iterate over a HashMap's **keySet**

# Key Set

Key	Value
Cat	2
Snake	2
T-Rex	1

`.keySet()`

**Set<K>**

**keySet()**


Returns a **Set** view of the keys contained in this map.

- `.keySet()`: gives Set of keys



# Key Set

Key	Value
Cat	2
Snake	2
T-Rex	1

`.keySet()` 

Cat
Snake
T-Rex

**Set<K>**

**keySet()**

Returns a **Set** view of the keys contained in this map.

- `.keySet()`: gives Set of keys

# Key Set

Key	Value
Cat	2
Snake	2
T-Rex	1

.keySet () →

Cat
Snake
T-Rex

**Set<K>**

**keySet ( )**

Returns a **Set** view of the keys contained in this map.

- .keySet(): gives Set of keys
  - New concept: **Set**, like “set” in math

# What Can You Do with a Set?

- Iterate over set with for each

```
for (String s : ips) { ... }
```
- `.add(element)`: add to Set
- `.remove(element)`: remove from Set
- `.contains(element)`: check if in Set
- Many more methods: check documentation  
<http://docs.oracle.com/javase/8/docs/api/java/util/Set.html>