

# Finding Unique IP Addresses

Equality

# ArrayList: Had Strings

```
public int countUniqueIPs() {  
    ArrayList<String> uniqueIps = new ArrayList<String>();  
    for (LogEntry le : records) {  
        String ipAddr = le.getIpAddress();  
        if (!uniqueIps.contains(ipAddr)) {  
            uniqueIps.add(ipAddr);  
        }  
    }  
    return uniqueIps.size();  
}
```

# ArrayList: Had Strings

```
public int countUniqueIPs() {  
    ArrayList<String> uniqueIps = new ArrayList<String>();  
    for (LogEntry le : records) {  
        String ipAddr = le.getIpAddress();  
        if (!uniqueIps.contains(ipAddr)) {  
            uniqueIps.add(ipAddr);  
        }  
    }  
    return uniqueIps.size();  
}
```

# ArrayList: Had Strings

```
public int countUniqueIPs() {  
    ArrayList<String> uniqueIps = new ArrayList<String>();  
    for (LogEntry le : records) {  
        String ipAddr = le.getIpAddress();  
        if (!uniqueIps.contains(ipAddr)) {  
            uniqueIps.add(ipAddr);  
        }  
    }  
    return uniqueIps.size();  
}
```

# What If: `ArrayList<LogEntry>`?

```
public int countUniqueIPs() {  
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();  
    for (LogEntry le : records) {  
        if (!uniqueIps.contains(le)) {  
            uniqueIps.add(le);  
        }  
    }  
    return uniqueIps.size();  
}
```

# What If: ArrayList<LogEntry>?

```
public int countUniqueIPs() {  
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();  
    for (LogEntry le : records) {  
        if (!uniqueIps.contains(le)) {  
            uniqueIps.add(le);  
        }  
    }  
    return uniqueIps.size();  
}
```



# What If: ArrayList<LogEntry>?

```
public int countUniqueIPs() {  
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();  
    for (LogEntry le : records) {  
        if (!uniqueIps.contains(le)) {  
            uniqueIps.add(le);  
        }  
    }  
    return uniqueIps.size();  
}
```

# What If: ArrayList<LogEntry>?

```
public int countUniqueIPs() {
```

```
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();
```

```
    for (LogEntry le : records) {
```

```
        if (!uniqueIps.contains(le)) {
```

```
            uniqueIps.add(le);
```

```
        }
```

```
    }
```

```
    return uniqueIps.size();
```

```
}
```

Gives wrong answer:

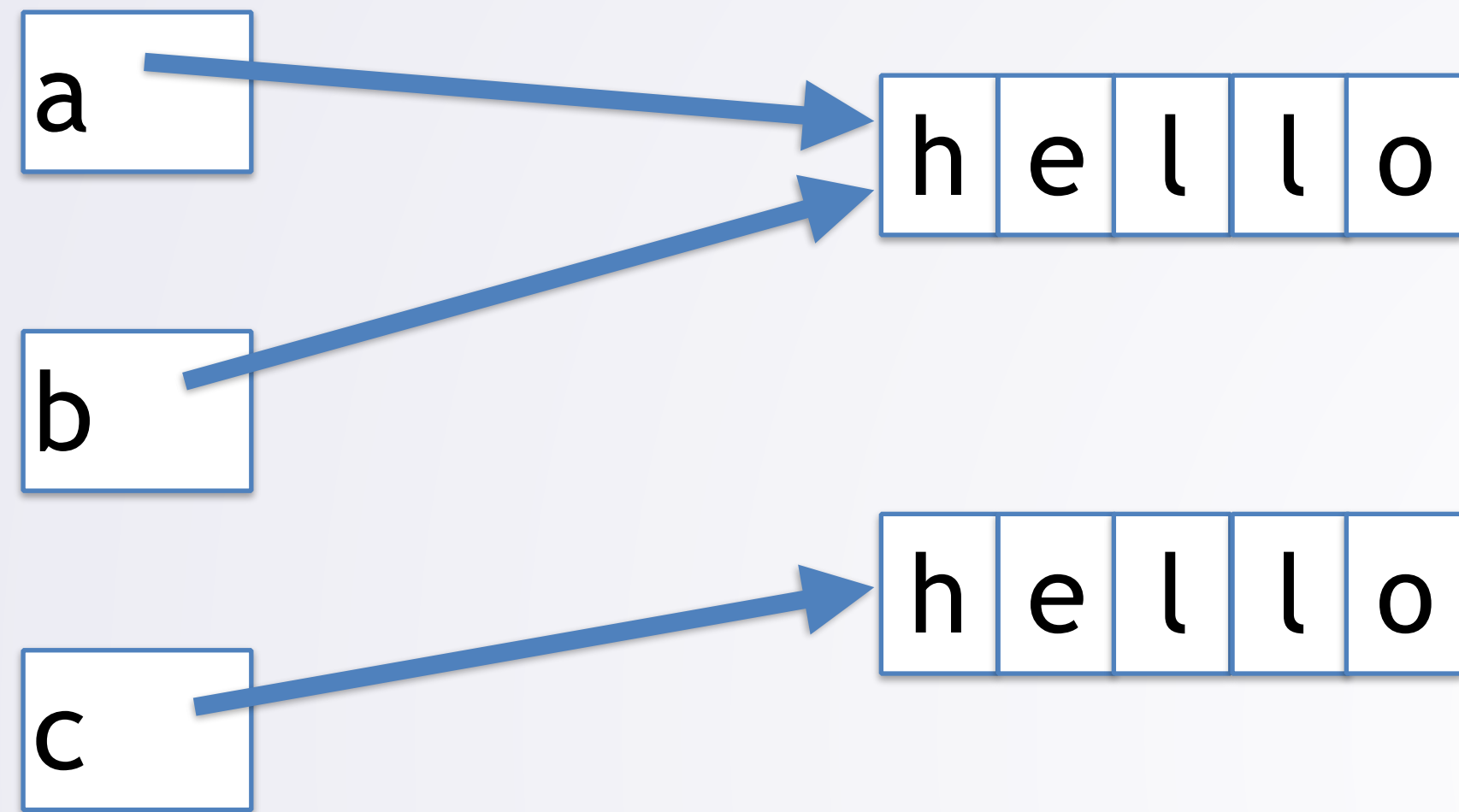
- Tells total entries, not **unique** IPs
- Why?



# How Does contains Check Equality?

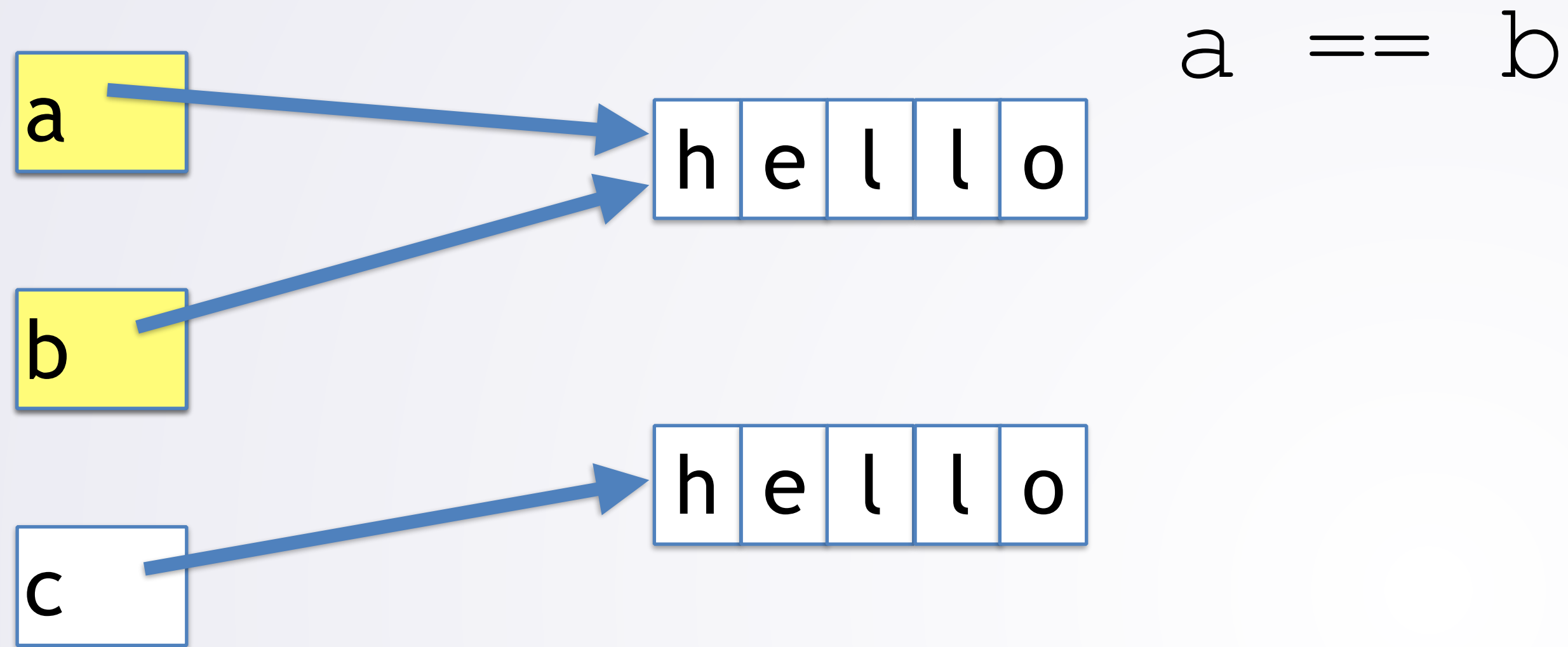
```
public int countUniqueIPs() {  
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();  
    for (LogEntry le : records) {  
        if (!uniqueIps.contains(le)) {  
            uniqueIps.add(le);  
        }  
    }  
    return uniqueIps.size();  
}
```

# == vs .equals



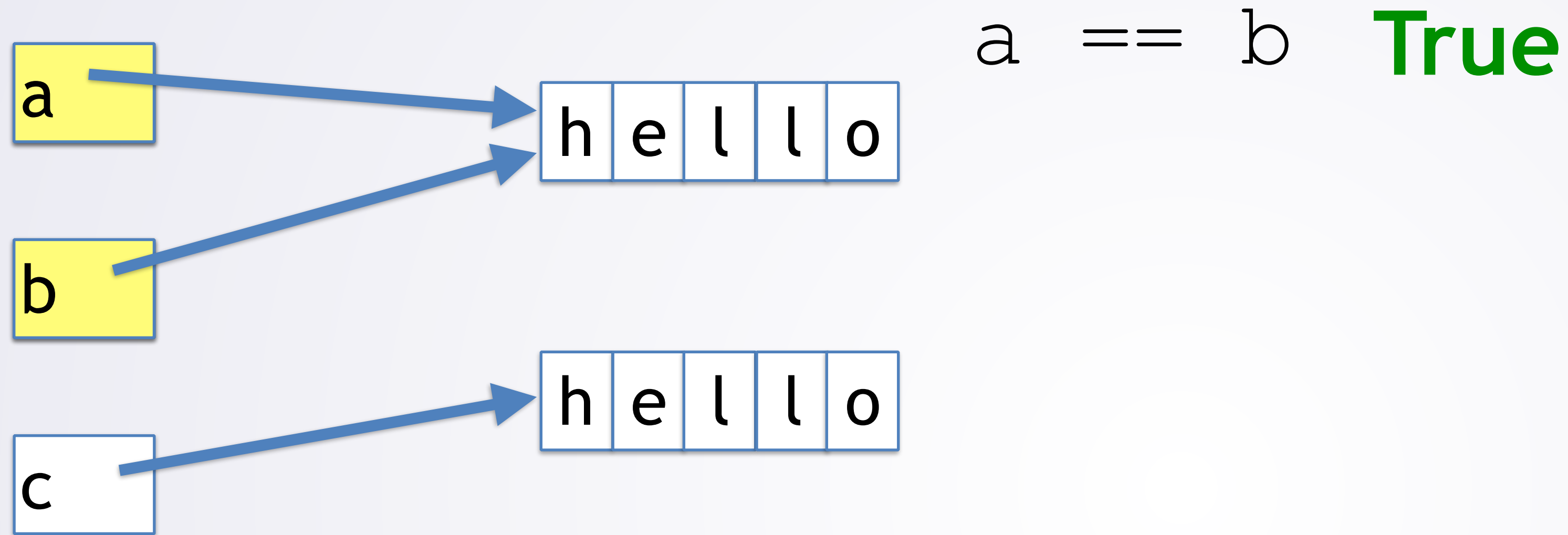
- Two different kinds of equality:
  - ==: same exact object

# == vs .equals



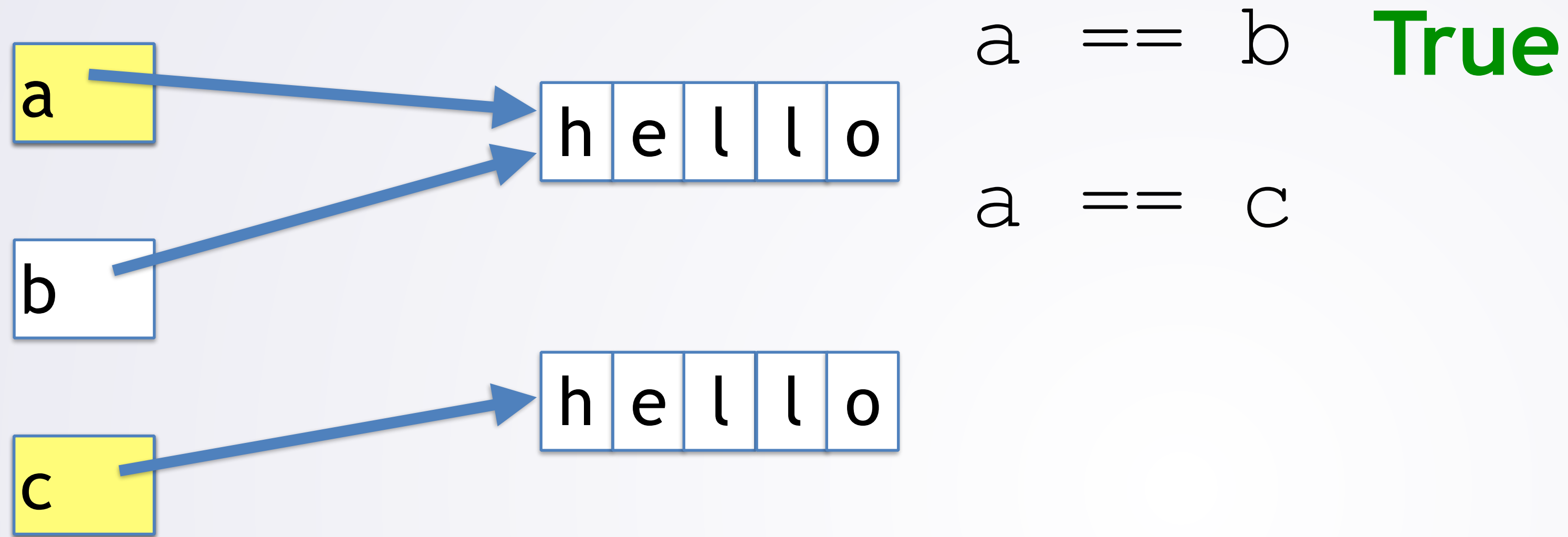
- Two different kinds of equality:
  - ==: same exact object

# == vs .equals



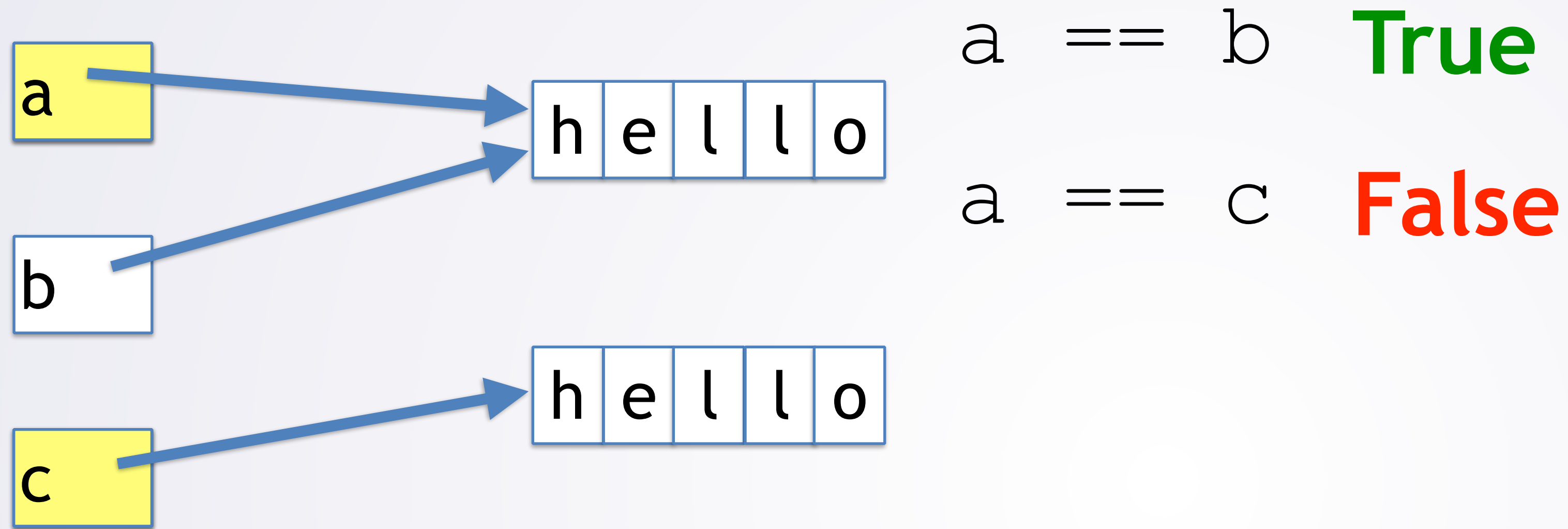
- Two different kinds of equality:
  - ==: same exact object

# == vs .equals



- Two different kinds of equality:
  - ==: same exact object

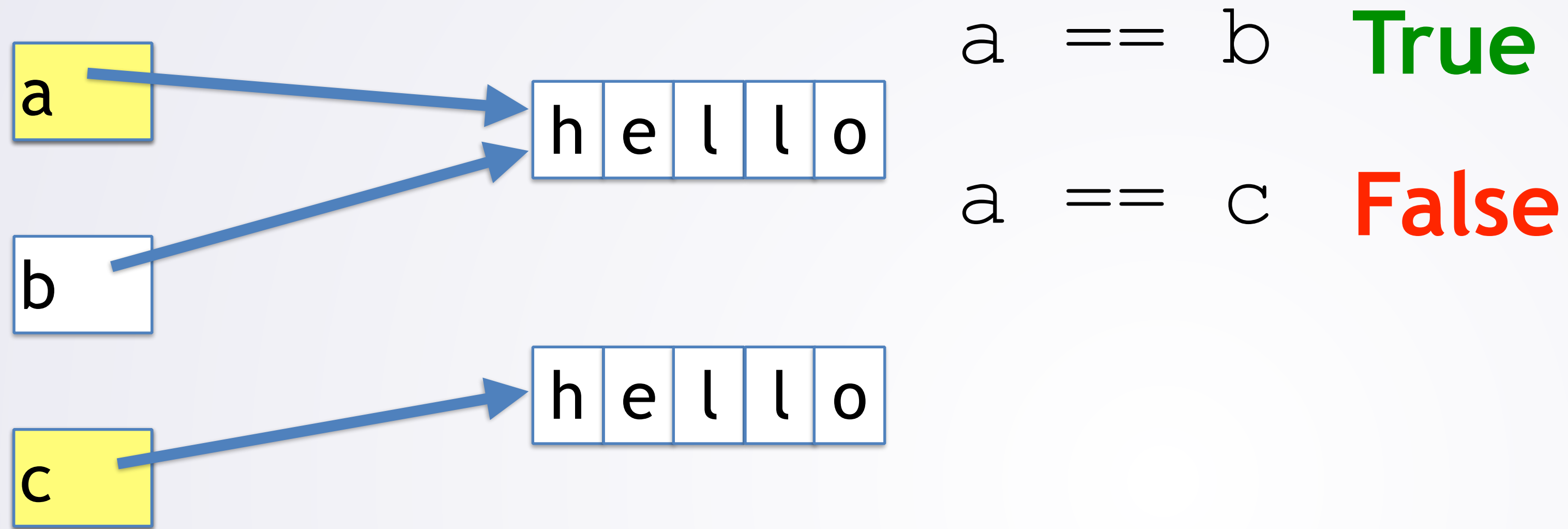
# == vs .equals



- Two different kinds of equality:
  - `==`: same exact object

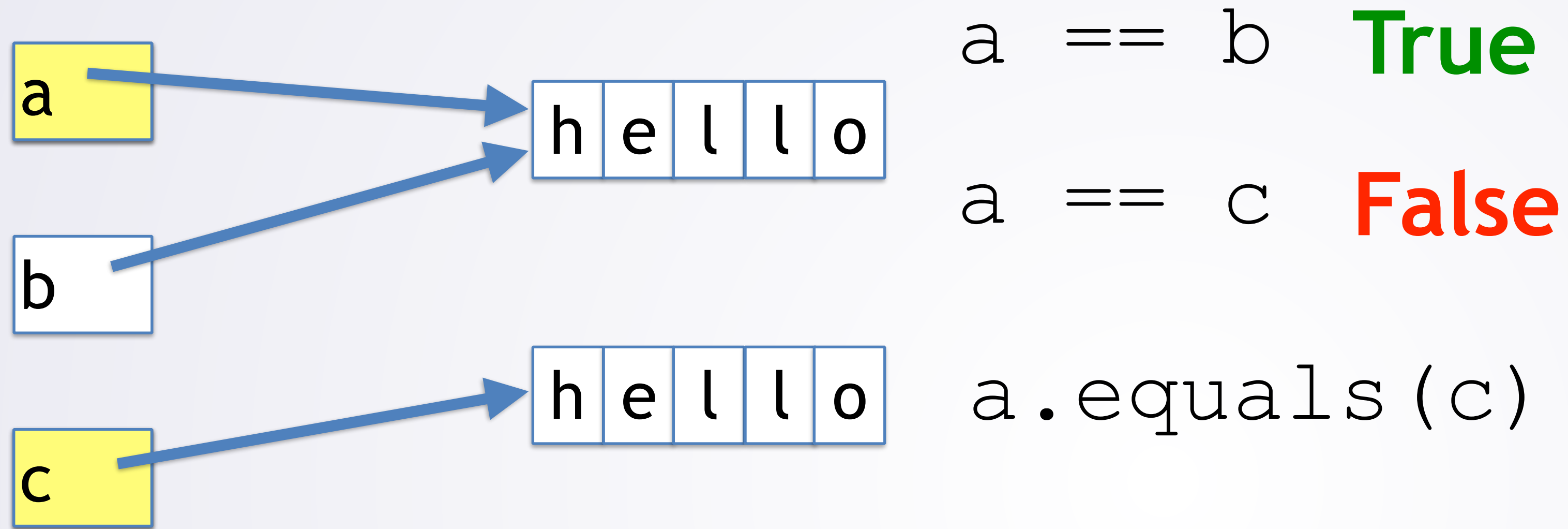


# == vs .equals



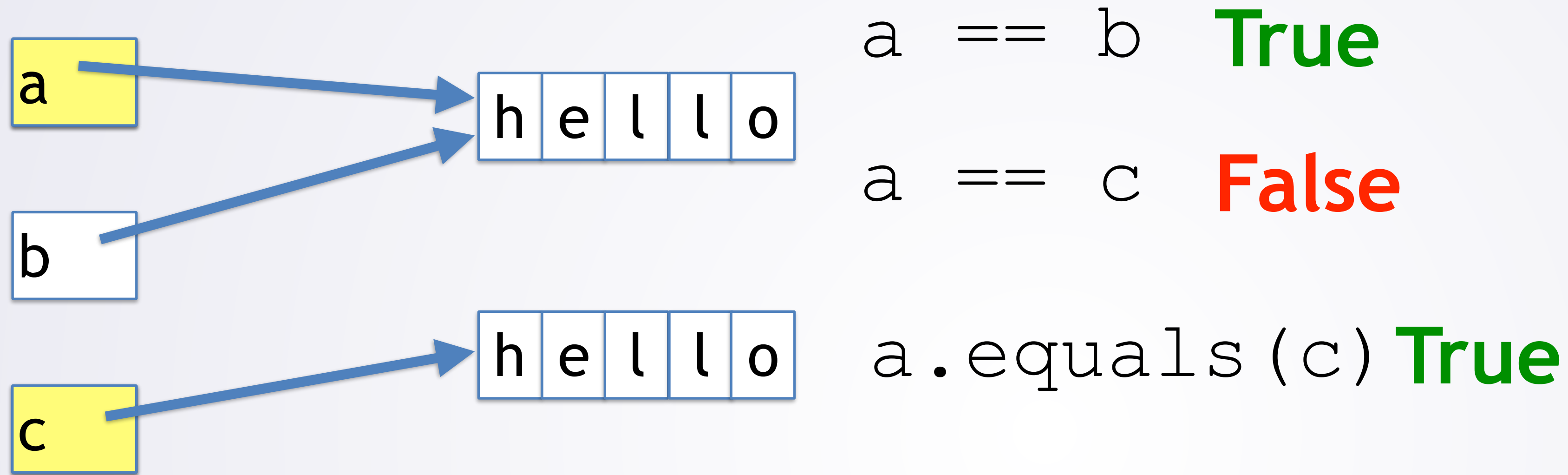
- Two different kinds of equality:
  - `==`: same exact object
  - `.equals()`: same meaning

# == vs .equals



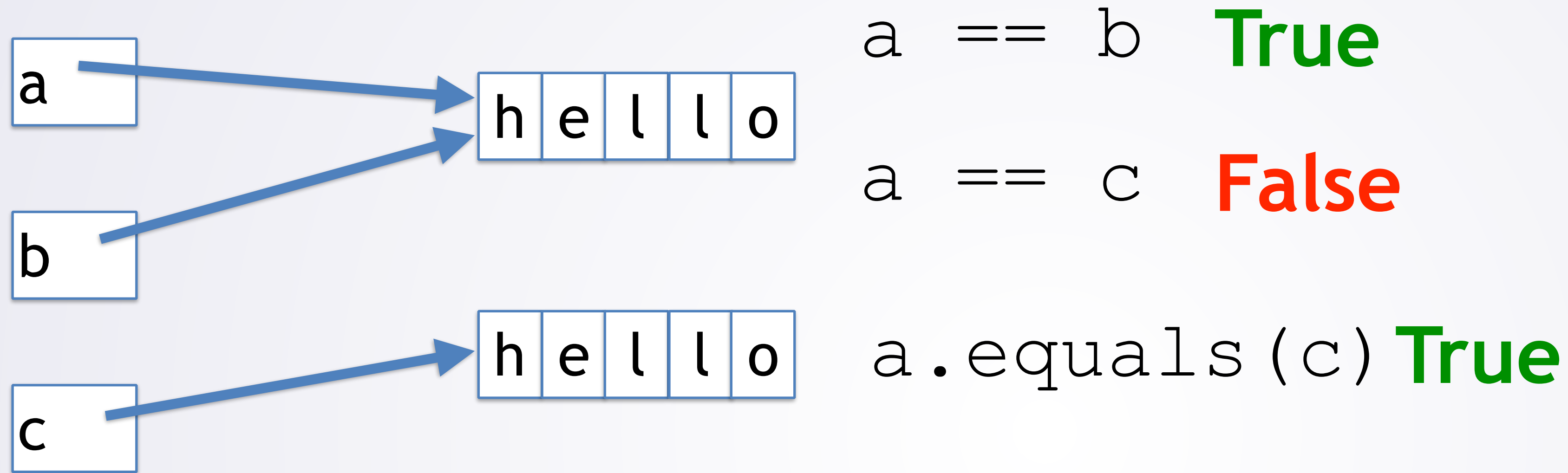
- Two different kinds of equality:
  - `==`: same exact object
  - `.equals()`: same meaning

# == vs .equals



- Two different kinds of equality:
  - `==`: same exact object
  - `.equals()`: same meaning

# == vs .equals



- Two different kinds of equality:
  - `==`: same exact object
  - `.equals()`: same meaning
    - Defined by class: default `==`

# Equality for LogEntry

- Should we write .equals for LogEntry?
- When are two LogEntries the same?
  - Same IP? **No**
  - Same IP and same request String? **No**
  - Same exact object **Yes**
- Same object is default behavior
  - No need to write .equals()
  - Will learn how to write .equals() later

# Understand Why This Is Broken

```
public int countUniqueIPs() {  
    ArrayList<LogEntry> uniqueIps = new ArrayList<LogEntry>();  
    for (LogEntry le : records) {  
        if (!uniqueIps.contains(le)) {  
            uniqueIps.add(le);  
        }  
    }  
    return uniqueIps.size();  
}
```



# ...And Why This Worked

```
public int countUniqueIPs() {  
    ArrayList<String> uniqueIps = new ArrayList<String>();  
    for (LogEntry le : records) {  
        String ipAddr = le.getIpAddress();  
        if (!uniqueIps.contains(ipAddr)) {  
            uniqueIps.add(ipAddr);  
        }  
    }  
    return uniqueIps.size();  
}
```