The mysterious hash

How can I find out if my delicious hash contains a specific "ingredient" in constant time?

Announcements

- Apt Set 2 - Due tonight
- Jotto – Due Sept 24

Primitives

```
int i = 5;
int j = 5;

if(i == j)
  doSomething();
else
  doSomethingElse();
```


Objects

```
int[] array1 = new int[5];
int[] array2 = new int[5];


if(array1 == array2)
  doSomething();
else
  doSomethingElse();
```

Objects

```
int[] array1 = new int[5];
int[] array2 = new int[5];


if(array1 == array2)
  doSomething();
else
  doSomethingElse();
```

DO NOT USE == FOR OBJECTS

Objects

- Primitives are saved as values
- Objects are saved as reference values – value that points to a location somewhere in memory

```
int i = 5;
int[] j = {1};
```

Address	Variable	Contents
@123	i	5
@124	j	@397
@397		1

Objects

- Primitives are saved as values
- Objects are saved as reference values – value that points to a location somewhere in memory

- `int[] a = {1,2,3};`
- `int[] b = {1,2,3};`

- **Does a == b?**

Address	Variable	Contents
@123	a	@418
@124	b	@397

Objects

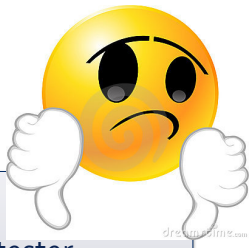
```
int[] array1 = new int[5];
int[] array2 = new int[5];


if(array1 == array2)
  doSomething();
else
  doSomethingElse();
```

DO NOT USE == FOR OBJECTS

APTs

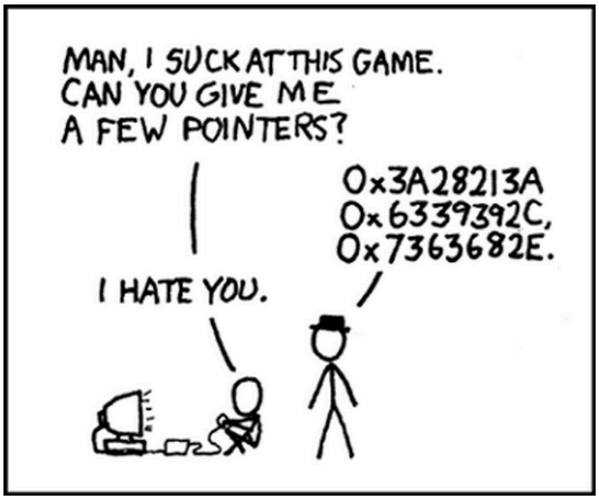
- WARNING

```
String s1 = "hello";
String s2 = "hello";
```



Eclipse	APT tester
if (s1 == s2)	if (s1 == s2)

Pointers



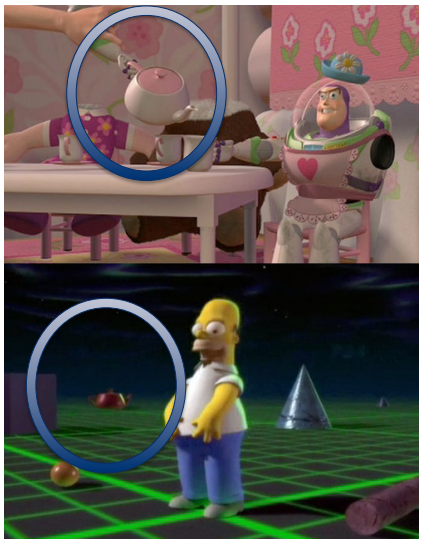
Objects

```
int[] array1 = new int[5];
int[] array2 = new int[5];

if(array1 == array2)
  doSomething();
else
  doSomethingElse();
```

DO NOT USE == FOR OBJECTS

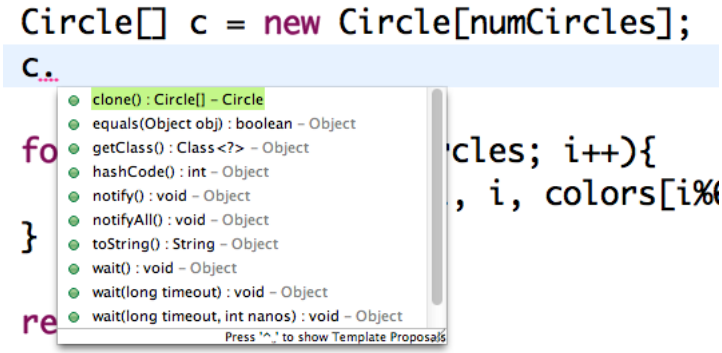
How to compare objects?

- Is this the same teapot?
- teapot1.equals(teapot2);



.equals()

- Built in Java function for Object
- All objects inherit .equals()

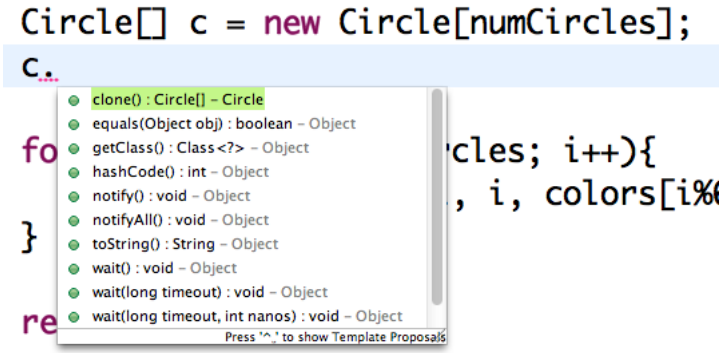


.equals()

```
1       ThreeInts a = new ThreeInts(5,5,5);
2       ThreeInts b = new ThreeInts(5,5,5);
3
4       if(a.equals(b))
5          System.out.println("equal");
6       else
7          System.out.println("not equal");
```

.equals()

- Built in Java function for Object
- All objects inherit .equals()
  - You can Override .equals() with your own code!

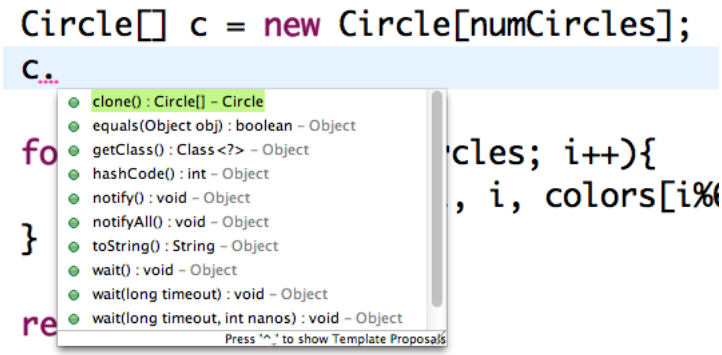


.equals()

```
1  public boolean equals(Object obj){
2        if (obj == this) {
3              return true;
4          }
5          if (obj == null || obj.getClass() !=
                        this.getClass()) {
6              return false;
7          }

8          YourObjectType temp = (YourObjectType) obj;
```

.hashCode()

- Built in Java function for Object
- All objects inherit .hashCode()
  - You can Override .hashCode() with your own code!



.hashCode()

- Hash Yourself!

```
String name = "Tabitha";
System.out.println(name.hashCode());
```

111673433

Hashing

- HashTable
  - array of fixed size

  - with a key to each location
  - each key is mapped to an index in the table

0	
1	joe 31
2	
3	mary 43
4	sam 14
5	
6	
7	
8	
9	

9/15/13 19

Hashing

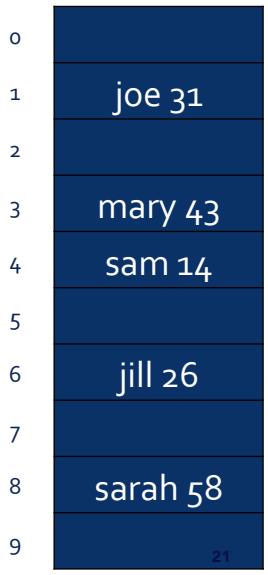
- Hash function
  - simple to compute
  - ensure two distinct keys get different cells

0	
1	joe 31
2	
3	mary 43
4	sam 14
5	
6	
7	
8	
9	

9/15/13 20

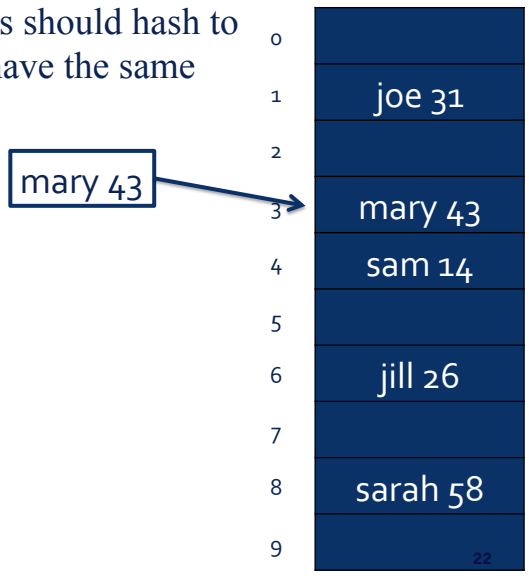
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



Hashing

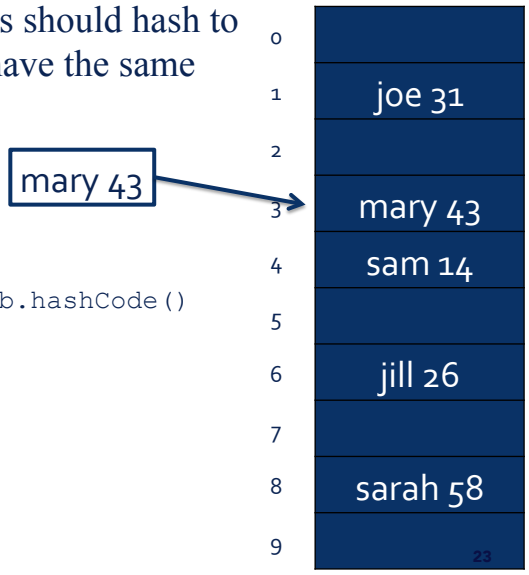
- Two equal objects should hash to the same place (have the same key)



Hashing

- Two equal objects should hash to the same place (have the same key)

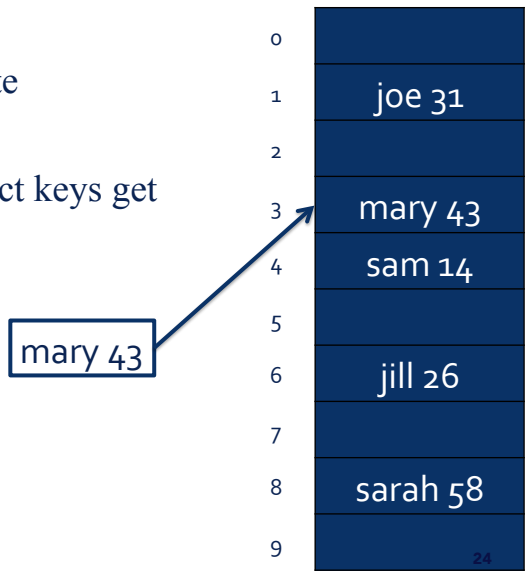
```
if a.equals(b)
then
   a.hashCode() == b.hashCode()
```



9/15/13 23

Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



9/15/13 24

Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells

jenny 23

0	
1	joe 31
2	
3	mary 43
4	sam 14
5	
6	jill 26
7	
8	sarah 58
9	

Hashing

- Separate Chaining
  - make your table into a list!

0		
1	joe 31	
2		
3	mary 43	jenny 23
4	sam 14	
5		
6	jill 26	
7		
8	sarah 58	
9		

.hashCode()

```
if a.equals(b)
then a.hashCode() == b.hashCode()
```

HOWEVER

```
if a.hashCode() == b.hashCode()
then a.equals(b) || !a.equals(b)
```

Today

- Do NOT use == for objects
- .equals()
- .hashCode()
- HashTables