toString() and equals ()

Date class

Printing - primitives vs Objects

```
System.out.println(num1);
```

- Will output

```
           16
```

Consider...

```
  System.out.println(myOblong);
  System.out.println(myDate);
```

Output

```
   Oblong@11b86e7
   Date@35ce36
```

`toString()`

- When you create a class the Java system provides the method `toString()` for that class. (Inherited from `Object`)
- The method `toString()` is used to convert an object to a String object.
- When an ***object reference*** is provided as a parameter to the method's `print()` or `println()`, the `toString()` method is called automatically.

Default `toString()`

- The default definition of `toString()` creates a string representing the name of the object's class, followed by the hash code of the object.

```
Oblong@11b86e7
```

- `Oblong` is the name of the object myOblong's class and the hash code for the object referenced by myOblong is @11b86e7

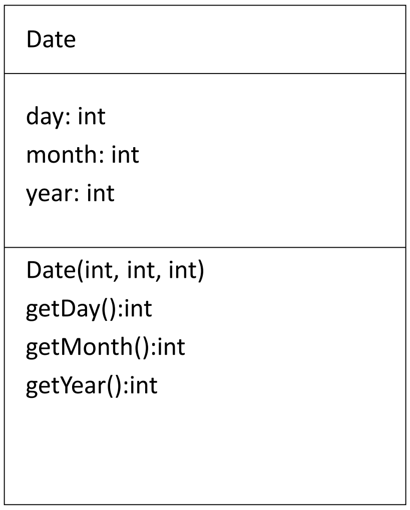
  (this will differ…)

toString()

- `toString()` is a public, value returning method
- It does not take any parameters and returns a `String`.
- The header/signature of the method `toString()` is always

```
public String toString()
```

UML for Date class



- For the class `Date` you want the method `toString()` to create and return the String in the format ***d/m/y***
- The String consists of the object's day, month, year and the forward slashes as shown

Definition for the `toString()` method for `Date`

```
public String toString()
{
      String str = new String();

      str = str + day + "/";
      str = str + month + "/";
      str = str + year;
      return str;
} //end
```

How it works

- `str` is a String variable used to create the required string.
- The `toString()` method is useful for outputting the values of the instance variables
- Only returns the (formatted) string; use the methods `print()` or `println()` to display the string in `main()`

How do you invoke `toString()`?

```
System.out.println(paddysDay.toString());
```

OR

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

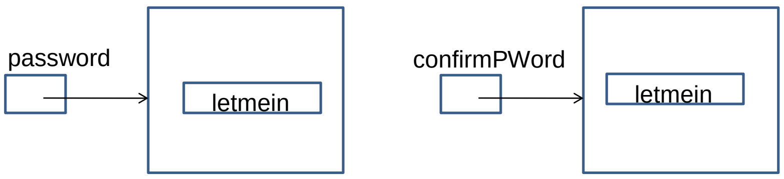
Test if two objects are equal consider the code

```
System.out.print("Enter your new password:  ");
String password = keyIn.nextLine();
System.out.print("Re-enter password to confirm: ");
String confirmPWord = keyIn.nextLine();

if(password == confirmPWord)
    System.out.print("password  confirmed  ");
else
    System.out.print("Not the same - please re-enter
                         password  ");
```

What will this do?

Test if two `String` objects are equal

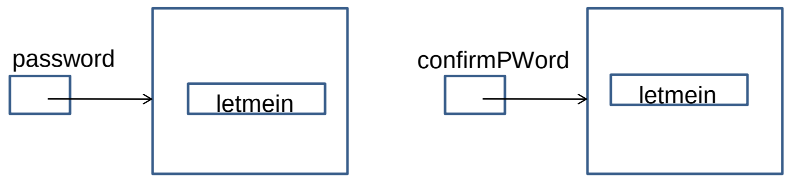


```
if (password == confirmPWord)
...
```

Here, the == checks to see if the two object references point to the same item in memory

This always returns false

Test if two `String` objects are equal



```
if (password.equals(confirmPWord))
...
```

Test if two objects are equal

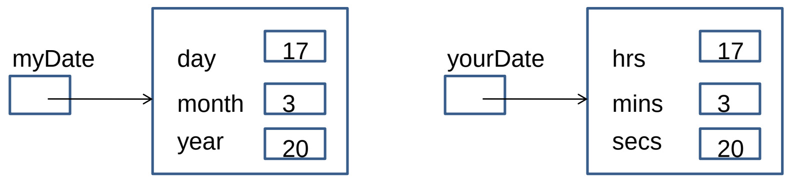
```
Clock myDate = new Date(17, 3, 20);
Clock yourDate = new Date(17, 3, 20);
```

- Consider the code

```
        if(myDate == yourDate)
```

- What would this do?

Objects `myDate` and `yourDate`



```
if (myDate == yourDate)
...
```

`equals()` method

```
public boolean equals(Date anotherDate)
{
  return(day == anotherDate.day
        && month == anotherDate.month
        && year == anotherDate.year);
}
```

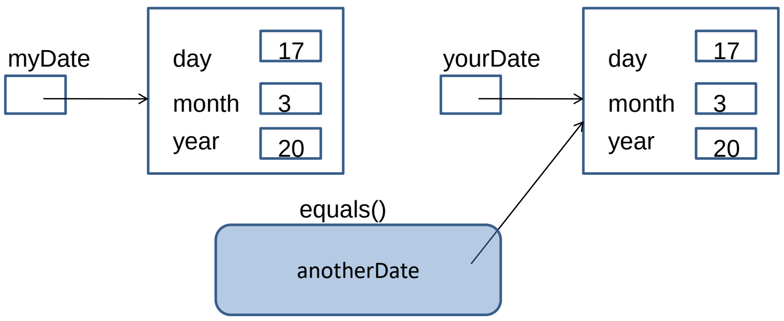
Calling `equals()`

```
if (myDate.equals(yourDate))
...
```

- In the expression:
  `myDate.equals(yourDate)` `myDate` accesses the method `equals()`

- The value of the parameter `yourDate` is passed to the parameter `anotherDate`

Objects `myDate` and `yourDate`



Summary

- The methods `toString()` and `equals()` are very useful to programmers.
- `toString()` allows the programmer to easily see the state of an object by printing out all its instance variables together.
- `equals()` allows the programmer to test two objects for equality.