# Dates and Formatters

# Dates

- There are two classes for dealing with dates in Java:
  - java.util.Date: This has been around since 1.0, but most functionality has been replaced by…
  - java.util.Calendar: You should use this class for most date needs, though the old Date class is often necessary
- Date is an ordinary class; Calendar is a factory class

# Creation

```
Date nowDate = new Date();
Calendar nowCalendar = Calendar.getInstance();
```

# Printing the Date

- You can print the contents of the calendar object in a couple of different ways:
  - With a SimpleDateFormat object
  - Or by getting the fields' data directly

```
// Format the date in a standard output format
Calendar calendar = Calendar.getInstance();
SimpleDateFormat formatter = new SimpleDateFormat();
System.out.println("Right NOW is: " + formatter.format(calendar.getTime()));

// Now format it my own way
System.out.println("Today is " + calendar.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG, Locale.US) +
                   ", which is day " + calendar.get(Calendar.DAY_OF_MONTH) +
                   " in the month of " + calendar.getDisplayName(Calendar.MONTH, Calendar.LONG, Locale.US) +
                   " in the year " + calendar.get(Calendar.YEAR));
```

# Manipulating Dates

- Change field values directly to manipulate Calendar values
  - Calendar.add() will add a given value to a field
  - Calendar.set() will set the value

```
// Hmm. How about sixty days from now?
calendar.add(Calendar.DAY_OF_MONTH, 60);
System.out.println("Sixty days from now is: " + formatter.format(calendar.getTime()));

// Or my birthday...
calendar.set(Calendar.MONTH, Calendar.FEBRUARY);
calendar.set(Calendar.DAY_OF_MONTH, 21);
calendar.set(Calendar.YEAR, 1973);
System.out.println("Sixty days from now is: " + formatter.format(calendar.getTime()));
```

# Use Cases

- Generally, use Calendar for manipulating getting or setting individual field values
- Date objects are prevalent in existing code, both open source and proprietary

# Formatters

- The primary class is java.util.Formatter
- Uses C-inspired syntax for formatting strings
- The primary method is format()
  - The first argument is the format specifier (see table below)
  - Remaining arguments are interpreted and put in their proper place in the specifier

# Specifiers

- General form:

  %[argument_index$][flags][width][.precision]conversion

  - argument_index, specifies which of the variable args should be used for conversion (only used when a single argument feeds multiple conversions
  - flag, modifies output, dependent on conversion
  - width, indicates the minimum number of characters to be written to the output
  - precision, usually used to restrict the number of characters, specific behavior depends on the conversion
  - conversion, a character indicating how the argument should be formatted, the set of valid conversions for a given argument depends on the argument's data type

# Conversions

| Conversion | Type | Description |
|---|---|---|
| ‘b’, ‘B’ | General | “false” for null or false boolean, else “true” |
| ‘h’, ‘H’ | General | “null” for null, otherwise hash code in hex |
| ‘s’, ‘S’ | General | “null” for null, otherwise per Formattable |
| ‘c’, ‘C’ | Character | A unicode character |
| ‘d’ | Integral | A decimal integer |
| ‘o’ | Integral | An octal integer |
| ‘x’, ‘X’ | Integral | A hexadecimal integer |
| ‘e’, ‘E’ | Integral | A decimal number in scientific notation |
| ‘f’ | Floating-point | A decimal number |
| ‘g’, ‘G’ | Floating-point | A decimal number may be scientific notation |
| ‘a’, ‘A’ | Floating-point | Hexadecimal floating-point |
| ‘t’, ‘T’ | Date/Time | Prefix for date time conversions |
| ‘%’ | Percent sign | A literal ‘%’ |
| ‘n’ | Line separator | Platform specific line separator |

# Other Formatters

- Specialized formatters in java.text
  - SimpleDateFormat formats date/time
  - DecimalFormat formats decimal numbers
  - Use the format() method to format the value
  - Use the parse() method to convert a string in the correct pattern to the appropriate object
  - Each formatter supports unique format strings and specialized methods
- The static String.format methods are a shortcut to java.util.Formatter
  - Useful for one-liners