

# More About Data Types

---



**Jim Wilson**

MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim jwhh.com



# Overview



Time of events

Human-friendly time

Date and time formatting

Primitive type wrapper classes

Classes and interfaces





## What time is it?

- Not as simple of a question as it seems

## The details of time can be complex

- The best way to represent time depends on what you want to do with it



# Time and Date



## Time of events

Primarily interested in  
sequencing and timestamp



## Local human-friendly time

Date and/or time of day



## Global human-friendly time

Date and time of day  
Understands time zone



# Tracking Time of Events



## Instant class

- Optimized for time-stamping events
- Works well for relative time comparisons
- Can be converted into more complex date/time types



# Instant Class

```
static void checkRelationship(Instant otherInstant) {  
    Instant nowInstant = Instant.now();  
    if(otherInstant.compareTo(nowInstant) > 0)  
        System.out.println("Instant is in the future");  
    else if(otherInstant.compareTo(nowInstant) < 0)  
        System.out.println("Instant is in the past");  
    else  
        System.out.println("Instant is now");  
}
```



# Local Human-friendly Time



**LocalTime**

Time of day

09:15:10.000000



**LocalDate**

Date only

2022-12-25



**LocalDateTime**

Date and time of day

2022-12-25T09:15:10.000000



# Local Human-friendly Time



## **Focuses on the date and/or time value**

- No time zone

## **Provide common operations**

- Finding differences
- Increasing/decreasing values
- Manipulating content
- Convert to/from string



# Global Human-friendly Time



## **ZonedDateTime**

- Operations similar to LocalDateTime
- Understands time zones

## **Strong time zone support**

- Can work with values across time zones
- Can convert to different time zones





## Converting date/time values to/from string

- By default each type is limited to a single string format

### **DateTimeFormatter**

- Describe date/time formatting
- Includes several predefined formats
- Can be used when converting to string
- Can be used when parsing from string



# Converting to String with DateTimeFormatter

Main.java

```
LocalDate today = LocalDate.now();
```

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

```
DateTimeFormatter usDateFormat =
```

```
    DateTimeFormatter.ofPattern("MM-dd-yyyy");
```

```
System.out.println(today.format(usDateFormat));
```

2022-03-01

03-01-2022

# Parsing a String with DateTimeFormatter

```
String usDateString = "07-04-2022";
```

```
LocalDate failedDate = LocalDate.parse(usDateString); // ERROR!!
```

```
DateTimeFormatter usDateFormat =
```

```
    DateTimeFormatter.ofPattern("MM-dd-yyyy");
```

```
LocalDate theDate = LocalDate.parse(usDateString, usDateFormat);
```





## Primitive types

- byte, short, int, long
- float, double
- char
- boolean

## Primitive types represent data only

- Unable to provide methods for operating on that data



## Primitive wrapper classes

- Can hold primitive data values
- Provide methods
- Enable compatibility with richer aspects of Java type system

## Each primitive type has a wrapper class

- Byte, Short, Integer, Long
- Float, Double
- Character
- Boolean



## Methods handle common operations

- Converting to from other types
- Extracting values from strings
- Finding min/max values
- Many others





## We are off to a great start

- Store and manipulate data
- Conditional logic and looping
- Organize code into methods
- Interact with the user
- Utilize existing complex data types like strings, date, and time

## We're now ready to go to the next level

- Creating and using our own complex types







## Classes

- Contain state
- Contain code to manipulate that state
- Allow us to create custom data types

## Interfaces

- Model data type behavior
- Create contracts between data types



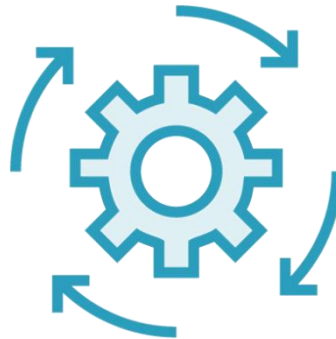
# Classes and Interfaces

Understanding classes and interfaces is essential to working in Java



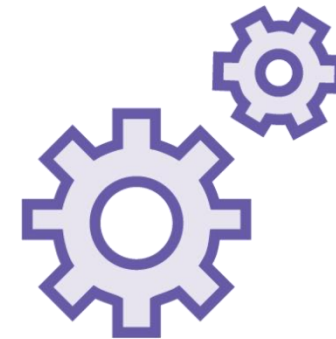
## Create rich applications

Simplifies modeling and implementing complex problems



## Get the most from Java

Required to utilize many of the most powerful features of Java



## Leverage Java Libraries

Enables the effective use of Java's vast universe of libraries





Next course to watch

Working with Classes and Interfaces in Java



# Summary



## Date and time types

- Each designed for specific use
- Time-stamping events
- Local date/time values
- Global date/time values

## DateTimeFormatter

- Describe date/time formatting
- Can be used when converting to string
- Can be used when parsing from string



# Summary



## Primitive wrapper classes

- Can hold primitive data values
- Provide methods
- Enable compatibility with richer aspects of Java type system





What to watch next

Working with Classes and Interfaces in Java

