CISC 3115

Java API Classes: Date & Time, Random, and Math

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Just discussed
  - Default constructor
  - Accessing objects via reference variables
  - Primitive and reference variables
  - Garbage collection
- A few classes in the Java Library (Java API)
  - Date & Time (LocalDateTime & ZonedDateTime), Random, Math

Java API and Library

- Java API: Java Application Programming Interface
  - Define the interface with which an application interacts with Java
    - Classes and methods that an application programmer can use in their own programs
- Java Library: implementation of the classes and methods

A Few Classes in Java Library

- Date & Time
  - LocalDateTime
  - ZonedDateTime
- Random
- Math

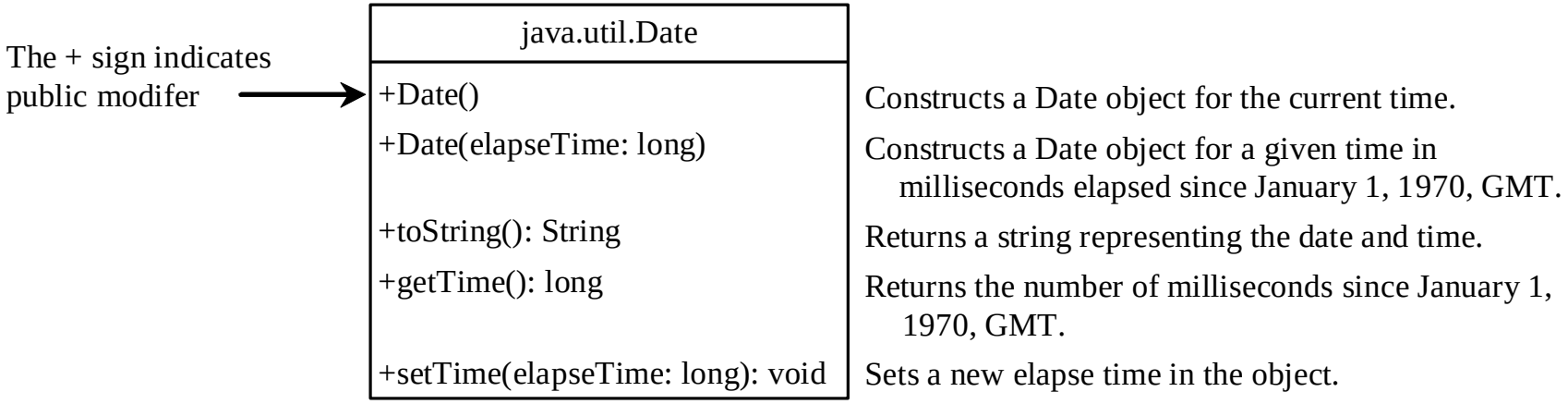
Representing Date

- It turns out that it is complex to represent date in programs
- Java has evolved a great deal to represent properly the concept of “date”
- Discuss Date and several related Java classes

The Date Class

- A system-independent encapsulation of date and time in the java.util.Date class.
  - Mostly deprecated.
- Represent a specific instant in time, with millisecond precision
- Example usage:
  - You can use the Date class to create an instance for the current date and time and use its toString method to return the date and time as a string.
- API documentation (may be intimidating to some, a good read nonetheless)
  - https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

The Date Class: UML Class Diagram



- Why “public”?

The Date Class: Example

- Showing current system date and time
  - the import statement
  - Java package: java.util is a package while Date is a class in the package

PrintDate.java

```java
import java.util.Date;

public class PrintDate {
  public static void main(String[] args) {
    Date date = new Date();
    System.out.println(date.toString());

    long millis = date.getTime();
    System.out.println("It has been " + millis + " milliseconds since the Unix epoch");
  }
}
```

Deprecated Methods?

- A number of methods of the Date class are marked as deprecated.
- What does it mean?
- What should we do to get/set a specific Date object?
- Use the java.time.LocalDateTime and the java.time.ZonedDateTime classes

The LocalDateTime Class

- Java provides a new system-independent encapsulation of date and time in the java.time.LocalDateTime class.
- Represent a specific instant in time, with millisecond precision as seen on a wall clock
- Common actions one would perform for a date
- API documentation (may be intimidating to some, a good read nonetheless)
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/LocalDateTime.html

The LocaleDateTime Class: UML Class Diagram

java.time.LocaleDateTime
of(year:int, month:int, dayOfMonth:int, hour:int, minute:int, second:int):LocaleDateTime now(): LocaleDateTime

The LocalDateTime Class: Example

- Show a specific time and showing current system date and time

PrintLocalDateTime.java

```java
import java.time.LocalDateTime;
import java.time.ZoneOffset;

public class PrintLocalDateTime {
  public static void main(String[] args) {
    int year = 2023, month = 9, day = 7, hour=14, minute=15, second=12;
    LocalDateTime localDateTime = LocalDateTime.of(year, month, day, hour, minute, second);
    System.out.println("The given date and time are " + localDateTime);

    LocalDateTime localDateTimeNow = LocalDateTime.now();
    System.out.println("Now is " + localDateTimeNow);
    long millis = localDateTimeNow.toInstant(ZoneOffset.of("-4")).toEpochMilli();
    System.out.println("It has been " + millis + " milliseconds since the Unix epoch");
  }
}
```

The ZonedDateTime class

- Representing a date-time with a time-zone.
- API documentation:
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/ZonedDateTime.html
- Several related and useful classes:
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/ZoneId.html
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/DateTimeFormatter.html

Example 1. Getting and Displaying Current Wall Clock Time

```
// get current wall clock time

java.time.LocalDateTime ldt =
java.time.LocalDateTime.now();

// set up String representation format

java.time.format.DateTimeFormatter formatter =

      java.time.format.DateTimeFormatter.ofPattern
("MM/dd/yyyy HH:mm:ss");

// obtain the string representation with time zone

String ldtString = ldt.format(formatter);

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

What about this?

```
// get current wall clock time

java.time.LocalDateTime ldt =
java.time.LocalDateTime.now();

// set up String representation format

java.time.format.DateTimeFormatter formatter =

      java.time.format.DateTimeFormatter.ofPattern("MM/
dd/yyyy HH:mm:ss z"); // <- there is "z" here.

// obtain the string representation with time zone

String ldtString = ldt.format(formatter);

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

Example 2. Getting and Displaying Current Time with Time Zone:

```
// get current time with time zone
java.time.ZonedDateTime zdt =
        java.time.ZonedDateTime.now();
// set up String representation format
java.time.format.DateTimeFormatter formatter =
       java.time.format.DateTimeFormatter.ofPattern("MM/
dd/yyyy HH:mm:ss z");
// obtain the string representation with time zone
String zdtString = zdt.format(formatter);
System.out.println(zdtString);
```

Example 3. Displaying Current Wall Clock Time with Time Zone

```
// get current wall clock time

java.time.LocalDateTime ldt = java.time.LocalDateTime.now();

// assume we are in time zone “America/New_York”

ZonedDateTime zdt =
ldt.atZone(java.time.ZoneId.of("America/New_York"))

// set up String representation format

java.time.format.DateTimeFormatter formatter =

        java.time.format.DateTimeFormatter.ofPattern("MM/dd/yyyy
HH:mm:ss z");

// obtain the string representation with time zone

String zdtString = zdt.format(formatter);

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

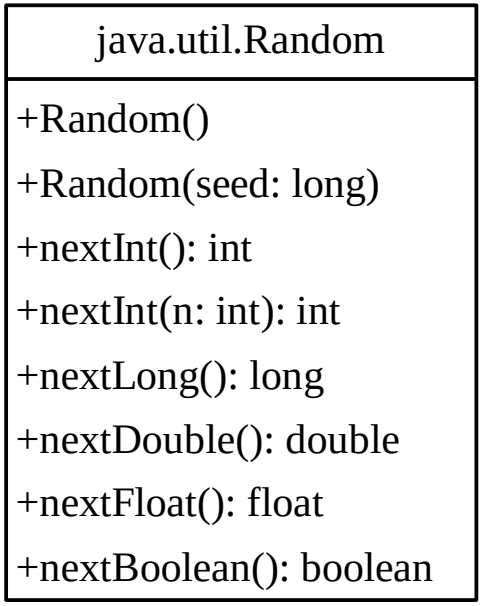
The Random Class

- A pseudo-random number generator in the java.util.Random class
- Use it to generate a sequence of pseudo-random numbers
- Example usage:
  - Generate a sequence random integers
  - Generate a sequence random float point numbers
  - Generate a sequence random Boolean values
- API documentation (may be intimidating to some, a good read nonetheless)
  - https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

The Random Class

- A pseudo-random number generator in the java.util.Random class
- Use it to generate a sequence of pseudo-random numbers
- Example usage:
  - Generate a sequence random integers
  - Generate a sequence random float point numbers
  - Generate a sequence random Boolean values
- API documentation (may be intimidating to some, a good read nonetheless)
  - https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

The Random Class: UML Class Diagram



Constructs a Random object with the current time as its seed.

Constructs a Random object with a specified seed.

Returns a random int value.

Returns a random int value between 0 and n (exclusive).

Returns a random long value.

Returns a random double value between 0.0 and 1.0 (exclusive).

Returns a random float value between 0.0F and 1.0F (exclusive).

Returns a random boolean value.

The Random Class: Example

- What is a “seed”?

RandomNumbers.java

```java
import java.util.Random;

class RandomNumbers {
  public static void main(String[] args) {
    Random random1 = new Random(3);
    System.out.print("From random1: ");
    for (int i = 0; i < 10; i++) {
      System.out.print(random1.nextInt(1000) + " ");
    }

    Random random2 = new Random(3);
    System.out.print("\nFrom random2: ");
    for (int i = 0; i < 10; i++) {
      System.out.print(random2.nextInt(1000) + " ");
    }

    Random random3 = new Random(4);
    System.out.print("\nFrom random3: ");
    for (int i = 0; i < 10; i++) {
      System.out.print(random3.nextInt(1000) + " ");
    }

    Random random4 = new Random();
    System.out.print("\nFrom random4: ");
    for (int i = 0; i < 10; i++) {
      System.out.print(random4.nextInt(1000) + " ");
    }

    Random random5 = new Random();
    System.out.print("\nFrom random5: ");
    for (int i = 0; i < 10; i++) {
      System.out.print(random5.nextInt(1000) + " ");
    }
  }
}
```

Pseudo-Random Numbers

- The Random class generates pseudo-random numbers, i.e., the numbers are generated by an algorithm
- Implication
  - They are in fact deterministic although appear random.
  - Given two identical seeds, the sequences of “random” numbers are identical as well

Math.random()

- The Math class in the Java Library has a random method
  - Generating pseudo-random double values in interval [0.0, 1.0)
  - Described in the API documentation
    - https://docs.oracle.com/javase/10/docs/api/java/lang/Math.html#random()

"When this method is first called, it creates a single new pseudorandom-number generator, exactly as if by the expression

```
new java.util.Random()
```

This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else. "

Questions?

- Concept of Java API and Java Library
- A few classes in the Java Library
  - Date and several related classes
  - Random
  - Math
- In which Java packages are they?