ROAD-TO-NINJA

Core Java - Beginner (Part 4)

Standard IO, String, Date-TimeZone-Locale

Organised by :

Supported by :

ABOUT ME

Name :	**Mohd Azman Kudus**
Age :	30 years
Java exp :	7 years

Question?

AGENDA

- Standard Input/Output
  - STDOUT
  - STDERR
  - STDIN
  - Interactive/Prompt
- String Manipulation
  - Length/SizeMatch
- Date
  - Calendar
  - TimeZone
  - Locale

STANDARD INPUT & OUTPUT

STDOUT

- Standard output.
- This output can be printed in console or pipe to another command/process.
- Usually with exit code 0 which means program terminated without any error.

```
System.out.print("Hello");
System.out.println("Hello World");
System.out.printf("%s - %s", "Hello", "World");
```

STDERR

- Standard error.
- Similar to STDOUT but only for error purpose.
- Usually will comes with exit code > 0.
- Cannot be piped to another program as STDIN.

```
System.err.print("Hello");
System.err.println("Hello World");
System.err.printf("%s - %s", "Hello", "World");
System.exit(1);
```

```
Unix/Linux/Mac/Solaris:
> echo $?

Windows:
> echo %errorlevel%
```

STDIN

- Standard input.
- Input comes from previous command's STDOUT.

```
public class FirstApp {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
public class SecondApp {
    public static void main(String[] args) {
        byte[] buffer = new byte[8192];
        int read = 0;
        try (InputStream stdin = System.in;
             ByteArrayOutputStream outStream = new ByteArrayOutputStream()) {
            while ((read = stdin.read(buffer)) != -1) {
                outStream.write(buffer, 0, read);
            }
            byte[] outArray = outStream.toByteArray();
            String stdinStr = new String(outArray, StandardCharsets.UTF_8);
            System.out.println("From STDIN: " + stdinStr);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

STDIN


```
> java FirstApp | java SecondApp
From STDIN: Hello World
```

INTERACTIVE

- Prompt input for user interaction or manual entry for data feed.

```
try (Scanner scanner = new Scanner(System.in)) {
    System.out.print("Enter message: ");
    String message = scanner.nextLine();

    System.out.print("Enter a number: ");
    int num1 = scanner.nextInt();

    for (int i = 1; i <= num1; i++) {
        System.out.println(i + " - " + message);
    }
}
```

Get all input after user press enter/return

Get single user input before space entered by user.

```
Enter message: Hello World
Enter a number: 5
1 - Hello World
2 - Hello World
3 - Hello World
4 - Hello World
5 - Hello World
```

STRING

STRING

- Sequence of characters.
- Can be manipulated as a whole or partial.

- Length / Count

```
String a = "Hello";
int length = a.length();
```

Number of characters.

- Actual size in byte

```
String a = "Hello";
int size = a.getBytes().length;
```

Number of bytes used to store the string.

- Blank

```
String a = "HELLO";
boolean empty = a.isEmpty();
```

same as if a.length() = 0

CHARACTER SETS

- Usually in Unicode (UTF-8) character sets to support other region font types and custom types.
- Most of the time, we use 1 byte character.
- There are multi-bytes characters that used around the world

UTF-8
from 1-4 bytes

Common.
Might confuse with multi-bytes.

ISO/IEC 8859-1
8 bits / 1 byte only

Not good for internationalisation

UTF-16
from 2-4 bytes

1-byte become 3-bytes.

Windows-1252
8 bits / 1 byte only
Latin characters only.

UTF-32
fixed to 4 bytes

Memory hog.
Faster operation.

STRING LOCATION

- Location/Index/Position
- Return positive integer if exists, else -1.

```
String a = "HELLO";
int index1 = a.indexOf("L");
int index2 = a.indexOf("X");

int index3 = a.lastIndexOf("L");
int index4 = a.lastIndexOf("X");
```

First character's position of first occurence

First character's position of last occurence

- Character at specific position/index.

```
char char1 = a.charAt(4);
char char1 = a.charAt(5);
```

STRING MATCH

- Exact match

```
String a = "HELLO";

boolean equal1 = a.equals("HELLO");
boolean equal2 = a.equals("hello");

int compare1 = a.compareTo("HELLO");
int compare2 = a.compareTo("hello");
```

If 0, then match.
Else, not match.

- Partial match

```
String a = "HELLO";
boolean contain1 = a.contains("ELL");
boolean contain2 = a.contains("EEL");

boolean start1 = a.startsWith("HE");
boolean start2 = a.startsWith("he");

boolean end1 = a.endsWith("LO");
boolean end2 = a.endsWith("lo");
```

Match anywhere

Match at the beginning

Match at the ending

STRING MATCH

- Regular Expression

```
String a = "HELLO";
boolean match1 = a.matches("^HE");

boolean match2 = a.matches("LOO$");

boolean match3 = a.matches(".*EL.*");
```

Match at the beginning

Match at the ending

Match anywhere

- Case insensitive match

```
String a = "HeLlo";
boolean equal3 = a.equalsIgnoreCase("HELLO");
boolean equal4 = a.equalsIgnoreCase("hello");
```

STRING CHANGE

- Concatenation / Join

```
String a = "Hello";
String b = "World";
String c = a + b;

String d = String.join("#", "Hello", "World", "Java");
```

The separator

- Partial / Extract

```
String a = "Hello";
String b = a.substring(2);

String c = a.substring(0, 4);
```

From position/index 2 to the end.

From position 0 to position before 4, which is 3 (4-1).

- Split

```
String a = "Hello-World-Java";
String[] b = a.split("-");
```

The delimiter

STRING CHANGE

- Change case

```
String a = "HeLlo";
String b = a.toUpperCase();

String c = a.toLowerCase();
```

HELLO

hello

- Remove leading and trailing whitespace characters

```
String a = "   HELLO   ";
String b = a.trim();
```

STRING CHANGE

- Replace

```
String a = "Hello";
String c = a.replace("ll", "tt");
```

Hetto

- Replace using regex

```
String a = "Hello";
String b = a.replaceFirst("[Hl]", "yy");

String b = a.replaceAll("[Hl]", "yy");
```

yyello

match any character within the brackets

yyeyyyyo

STRING FORMAT

- Like system.out.printf

```
String str1 = "Hello";
String str2 = "World";

String str3 = String.format("%s - %s", str1, str2);

int num1 = 123;
String num2 = String.format("%d, num1);
String num3 = String.format("%05d, num1);

double num4 = 456.789;
String num5 = String.format(%.2f, num4);
String num5 = String.format(%010.2f, num4);
```

String

Sequence of arguments based on sequence of format.

Whole number

Decimal number

Leading zero to fill the desired length.

- Decimal formatting

```
int num1 = 123456;
DecimalFormat format1 = new DecimalFormat("0000000000");
String str1 = format1.format(num1);

double num2 = 456789.0123;
DecimalFormat format2 = new DecimalFormat("###,###,###.00");
String str2 = format2.format(num2);
```

Thousand separator

STRING CONVERT

- Character array

```
String str1 = "Hello";
char[] chars1 = str1.toCharArray();
```

- Numeric

```
String a = "123";
int aa = Integer.parseInt(a);

String b = "123.456";
double bb = Double.parseDouble(b);
```

Parse to number

STRING BUILDER

- Object to construct and manipulate String without re-create the resulting String.

```
String[] strAry = new String[] {"abc","def","ghi","jkl"};

StringBuilder builder = new StringBuilder("Test");
for (String s : strAry) {
    builder.append(s);
}

builder.insert(7, "First");


builder.delete(3,8);



String finalStr = builder.toString();
```

Manipulating without re-create and re-assign

insert at specific position

Delete from position 3 to position before 8 which is 7 (8-1)

Convert to String after finish manipulation

DATE & TIME

DATE & TIME

- Presentation of a timestamp.
- Smallest unit is nanoseconds. Most of the time, millisecond is sufficient.
- Can be formatted and parsed.

```
Calendar calendar1 = Calendar.getInstance();

Calendar calendar2 = new GregorianCalendar();

Calendar calendar3 = new GregorianCalendar(2018, 9, 1);

Calendar calendar4 = new GregorianCalendar(2018, 9, 1, 9, 28);

Calendar calendar5 = new GregorianCalendar(2018, 9, 1, 9, 28, 57);
```

Create calendar with specific timestamp up to seconds.

CALENDAR OPERATION

- Add year/month/day/hour/minute/seconds

```
Calendar calendar = Calendar.getInstance();
calendar.add(1, Calendar.YEAR);
calendar.add(2, Calendar.MONTH);
```

- Set year/month/day/hour/minute/seconds

```
Calendar calendar = Calendar.getInstance();
calendar.set(3, Calendar.DAY_OF_YEAR);
calendar.set(4, Calendar.HOUR);
```

Add/set value of specific time unit.

- Compare

```
Calendar calendar1 = Calendar.getInstance();

Calendar calendar2 = Calendar.getInstance();
calendar2.add(-3, Calendar.DAY_OF_YEAR);

Calendar calendar3 = Calendar.getInstance();
calendar3.add(3, Calendar.DAY_OF_YEAR);

boolean before = calendar1.before(calendar2);

boolean after = calendar1.after(calendar3);
```

DATE FORMAT

- Convert date value into human readable format.

```
DateFormat dateFormat = new SimpleDateFormat("EEEE, dd/MM/yyyy
hh:mm:ss.SSS aa Z");

Calendar calendar = Calendar.getInstance();
String dateStr = dateFormat.format(calendar.getTime());
```

Format can be used to convert to or read from String

- Parse string formatted date value into Date/Calendar object.

```
DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

String dateStr = "01/09/2018 09:29:57";
Date date = dateFormat.parse(dateStr);
Calendar calendar = Calendar.getInstance();
calendar.setTime(date.getTime());
```

DATE FORMAT

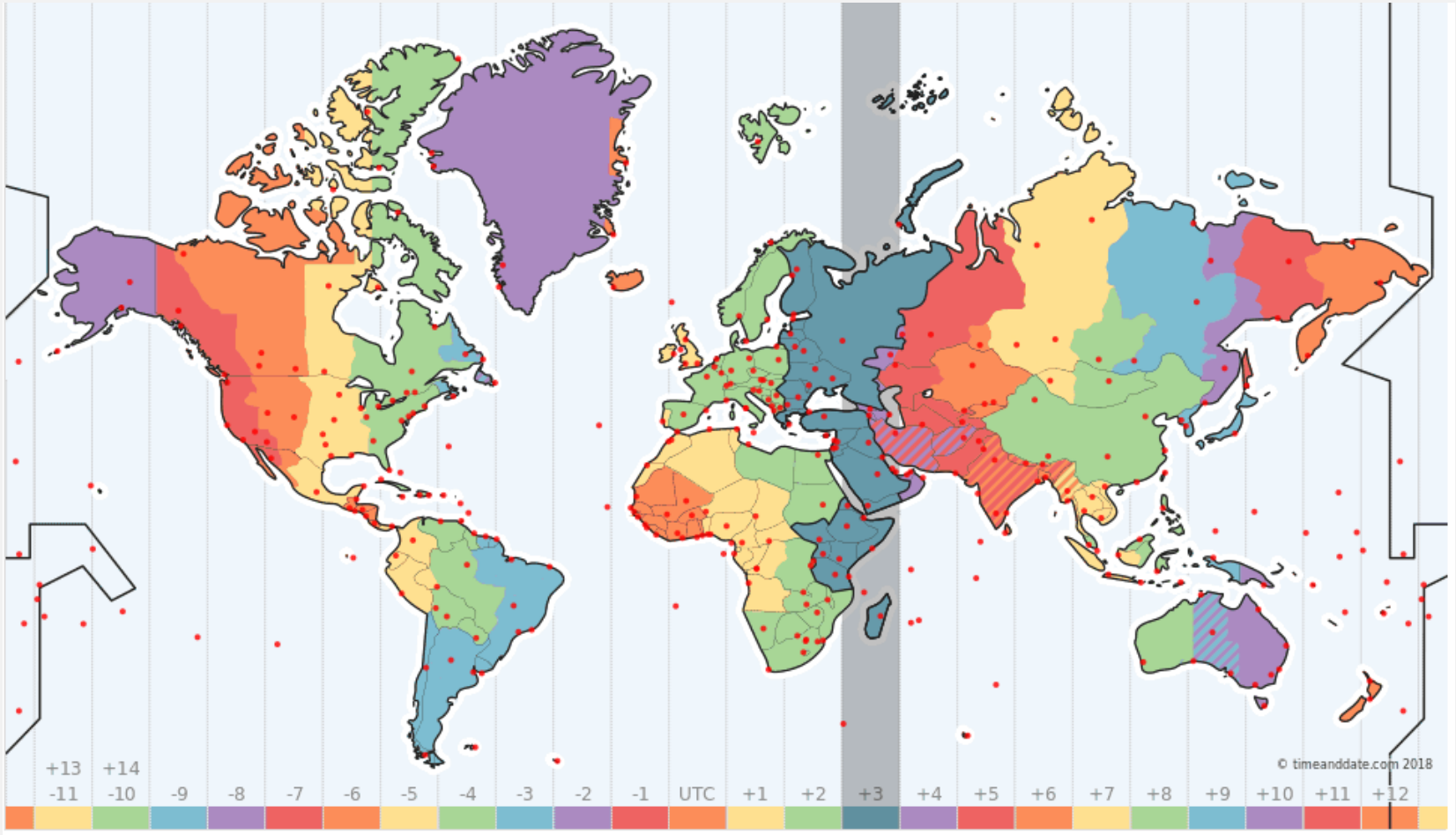
- Refer to the following URL for more details for the accepted formats.
- https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html
- Below are some common formats.

Format	Description
E	Day of week
d	Day of month (1-31)
M	Month of year (0-11)
y	Year
H	Hour (24 hours) (0-23)
h	Hour (12 hours) (1-12)

Format	Description
m	Minute (0-59)
s	Second (0-59)
S	Miliseconds
a	AM/PM
Z	TimeZone (GMT)

TIMEZONE

- A time zone is a region of the globe that observes a uniform standard time for legal, commercial, and social purposes



DATE & TIMEZONE

- Format to specific time zone

```
DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss.SSS
ZZZ");
dateFormat.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

Calendar calendar = Calendar.getInstance();
String dateStr = dateFormat.format(calendar.getTime());
```

TimeZone can be in region/UTC/GMT format.

- Check available timezones

```
for (String tz : TimeZone.getAvailableIDs()) {
    System.out.println(tz);
}
```

LOCALE

- Set of text or symbols that represents a language or region.

DATE & LOCALE

- Format to specific language

```
DateFormat dateFormat = new SimpleDateFormat("EEEE, dd MMMM yyyy
HH:mm:ss.SSS ZZZ", Locale.FRENCH);

Calendar calendar = Calendar.getInstance();
String dateStr = dateFormat.format(calendar.getTime());
```

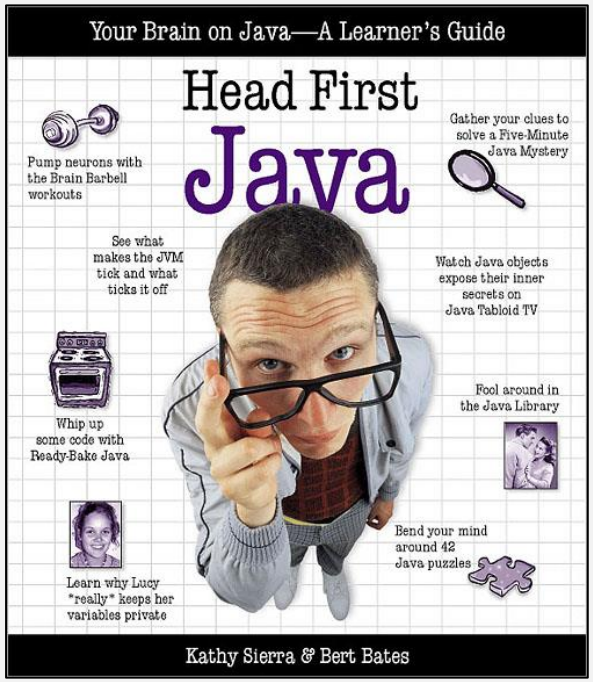
Java has some predefined locale.

mercredi, 02 septembre 2018 00:30:15.618 AM +0100

- Check available locale

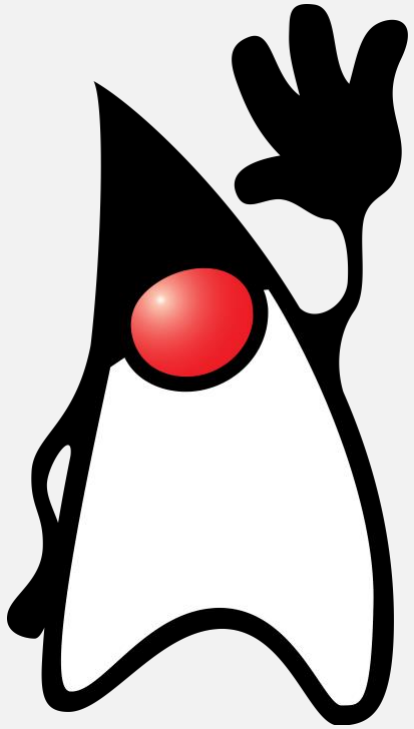
```
for (Locale l : Locale.getAvailableLocales()) {
    System.out.println(l.getDisplayName());
}
```

FURTHER READING



Head First Java (2nd Edition)

by Kathy Sierra (Author),
Bert Bates (Author)



THAT'S ALL FOR TODAY
SEE YOU IN THE NEXT CLASS