C17: Utility classes

Point

- Represents a location in 2D space:

```
class Point {
    public Point(int x, int y)        // constructor
    public int x                      // public data field !?
    public int y
    public void move(int x, int y) //move to new location
    public void translate(int x, int y) //change by offset
    public boolean equals(Object p) // ← Object
    public String toString()
}
```

Dimension

- Represents a rectangular size

```
class Dimension {
    public Dimension()
    public Dimension(int w, int h)
    public Dimension(Dimension d) // copy constructor

    public int width   //  public data field
    public int heigth

    public String toString()
}
```

Date

- Represents both times and dates

```
class Date {
    public Date()  //  current time and date!
    public Date(int y, int m, int d, int h, int m, int s)
    public int getMonth(), .., getSeconds(),
    public int getDay()       //  day of the week
    public int getYear()      //  year – 1900

    public long getTime()   // milliseconds since epoch
    public void setMonth(int m), .., setSeconds(int s)
```

More on Date

- Epoch = January 1st, 1970
- Compare dates:
  - public boolean after(Date day)
  - public boolean before(Date day)
- Use Date to measure elapsed realtime:

```
Date start = new Date(); // starting time
… // some code
Date end = new Date();  // ending time
long elapsedMillisecs = start.getTime()-end.getTime()
```

Math

- Supplies static constants and methods
  - public static final double E // = 2.71828
  - public static final double PI // = 3.1415926
  - Trigonometric ops (double → double):
    - sin, cos, tan, asin, acos, atan, atan2
  - Rounding ops: ceil, floor, rint, round
  - Exponentials: exp, pow, log, sqrt
  - Other: abs, max, min, random

Draw from weighted distribution

```
static public int weightedDistribution (int[] weights) {
    int sum = 0;  // sum of weights
    for(int i = 0; i < weights.length; i++)
        sum += weights[i];
    int val = (int) Math.floor(Math.random()*sum+1);
    for(int i = 0; i < weights.length; i++) {
        val -= weights[i];
        if (val < 0) return i; }
    return 0;  // should never happen
}
```

➔ weights {1,3,2} will yield p(0)=1/6, p(1)=1/2, p(2)=1/3

Random

- Math.random() can only generate uniform doubles >= 0.0, < 1.0,
- Class Random is more general, most important can supply seed for constructor ➔ reproducible sequences (setSeed(long))
- Can (directly) generate uniform ints, longs, floats, doubles
- And also: nextGaussian() !!!

Toolkit

- Abstract class specialized for each operating system, most methods should not be used, but some are useful utilities:
  - getFontList() returns list of names of available fonts
  - getImage() allows to load an image
  - getScreenSize() returns size in pixels (using an instance of Dimension)
  - getScreenResolution() in dots/inch
- Access actual Toolkit by calling
  - Toolkit.getDefaultToolkit()

System

- Supplies system-wide resources:
  - Streams: System.in, System.out, System.err
  - System.exit(int) terminates a program
  - long currentTimeMillis(): alternative way to access current time in milliseconds since epoch (January 1st, 1970)

    long result ➔ overflow in year 292280995

    (remember Y2k problem ☺)

String

- Immutable! i.e. cannot be changed

  String name = “John Smith”;

  char[] data = {‘q’,’e’,’d’};

  String quod = new String(data);
- Concatenation: +, but be careful, groups from left:

  System.out.println(“Catch-” + 2 + 2) ➔ “Catch22”

  System.out.println(2 + 2 + “warned”) ➔ “4warned”

  System.out.println(“” + 2 + 2 + “warned”) ➔ “22warned”

  // trick: empty leading string

String methods

- Will return copies in case of modifications
- Constructors from Strings and StringsBuffer, char and byte arrays
- concat, replace (characters), (retrieve) substring, toLowerCase, toUpperCase, trim (whitespace), valueOf, compareTo, equalsIgnoreCase, endsWith, startsWith, indexOf, lastIndexOf
- Can have unique strings (space-saving): intern

valueOf safer than toString

```
public static String valueOf(Object o) {
    return (o == null) ? "null" : o.toString();
    }
```

- Purely polymorphic and safe:

```
Shape aShape = null;
…
String a = String.valueOf(aShape); // "null"
String b = aShape.toString();
                    // nullPointerException
```

== and intern on Strings

```
String one = "One";
String two = new String(one);
String three = String.valueOf(one);

System.out.println((one == two)); // "false"
System.out.println((one == three)); // "true"

one = one.intern();
two = two.intern();
System.out.println((one == two)); // "true"
```

StringBuffer

- More like strings in C (arrays of char), can be modified:
  ```
  StringBuffer strbuf = new StringBuffer("hope");
  strbuf.setCharAt(0,'c');
  ```
- Constructors: StringBuffer(String initial), StringBuffer(int capacity)
- append, insert, and reverse modify buffer and return this thus allowing for cascaded calls:
  ```
  strbuf.append(" with ").append(209);
  ```
- setCharAt, charAt,
- length, setLength, ensureCapacity, toString

StringTokenizer

- Breaks a string into a sequence of tokens,
- tokens are defined by delimiters (e.g. space)
- Implements the Enumeration protocol

```
public StringTokenizer(String s)
public StringTokenizer(String s, String delims)
public boolean hasMoreElements()
public Object nextElement()
public String nextToken()
public int countTokens() // remaining tokens
```

StringTokenizer example

```
public void readLines (DataInputStream input) throws
   IOException {
    String delims = " \t\n.,!?;:";
    for(int line = 1; true; line++) {
        String text = input.readLine();
        if (text==null) return;
        text = text.toLowerCase();
        StringTokenizer e = new StringTokenizer(text,delim);
        while( e.hasMoreElements())
            enterWord(e.nextToken(), new Integer(line));
            }}
```

Parsing String Values

- For primitive datatypes wrapper classes provide parsing from strings and back:
  ```
  String dstr = "23.7";
  Double dwrap = new Double(dstr);
  double dval = dwrap.doubleValue();
  ```
- Instead of constructor:
  ```
  double dval = Double.parseDouble("23.7");
  ```
- Similar for ints, booleans, longs, and floats