Modular programming

v Organization of programs as independent modules.
v Why? à Easier to share and reuse code to create bigger programs.
v In Java we can consider each .java file as a module.
v Each .java file contains a (public) class.

Basic class concept

v Definition of a class (Example.java file):
```
   public class Example {
// Dice
// methods
}
```
v The Example.java file must contain a public class called Example.
- We should use a nomenclature of type Person, SomeClass, SomeLongNameForClass, ...
- Java is a case-sensitive language (i.e. Example! = Example)
v This class must be declared as public

Functions

v A role
- Performs a task.
- It has zero or more input arguments. - Returns zero or an output value.
v Applications
- Scientists use mathematical functions to calculate
formulas.
- Programmers use functions to build modular programs.
- We will use them for both purposes.
v Examples
Math.random (), Math.abs (), Integer.parseInt () System.out.println (), main ()

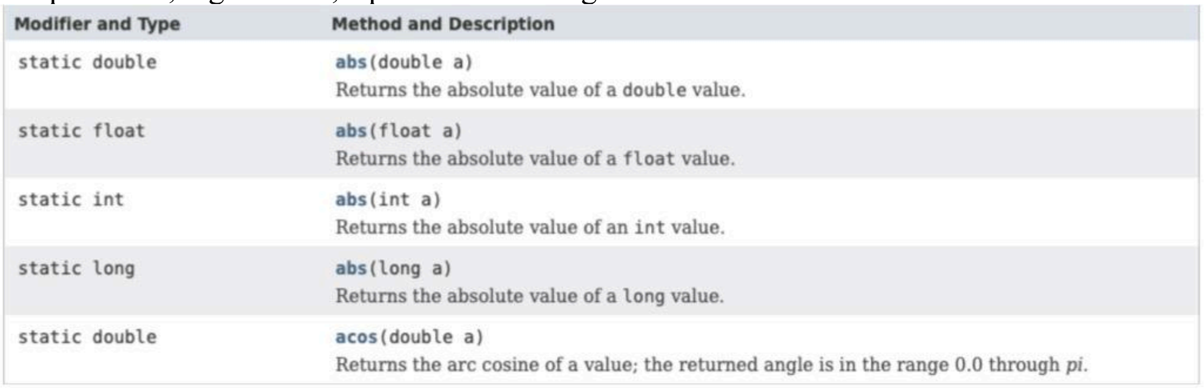
Static methods

v To implement a function (static method), we need
- Create a name
- Declaring the type and name of the argument (s) - Specifying the type for the return value
- Implement the method body
- End with return declaration
```
public static void myFunction () {System.out.println (“My Function called”);
}
public static double doisXSquare (double x) {return 2 * x * x;
}
```

---

java.lang.Math

v The Math class contains static methods for performing basic numeric operations
- exponential, logarithmic, square root and trigonometric functions.



Modifier and Type	Method and Description
static double	abs(double a) Returns the absolute value of a double value.
static float	abs(float a) Returns the absolute value of a float value.
static int	abs(int a) Returns the absolute value of an int value.
static long	abs(long a) Returns the absolute value of a long value.
static double	acos(double a) Returns the arc cosine of a value; the returned angle is in the range 0.0 through *pi*.

v General functions Math.abs ()
Math.ceil () Math.floor () Math.floorDiv () Math.min () Math.max () Math.round ()
Math.random ()
v Exponential, logarithmic functions Math.exp ()
Math.log () Math.log10 () Math.pow () Math.sqrt ()

v trigonometric functions
Math.PI Math.sin () Math.cos () Math.tan () Math.asin () Math.acos () Math.atan ()
Math.atan2 () Math.sinh () Math.cosh () Math.tanh () Math.toDegrees () Math.toRadians ()

The String class

v The java.lang.String class makes it easy to manipulate character strings.
v Example:
String s1 = "java"; // creating string by java string literal char ch [] = {'s', 't', 'r', 'i', 'n', 'g', 's'};
String s2 = new String (ch); // converting char array to string System.out.println (s1);
System.out.println (s2);
  java strings

String concatenation

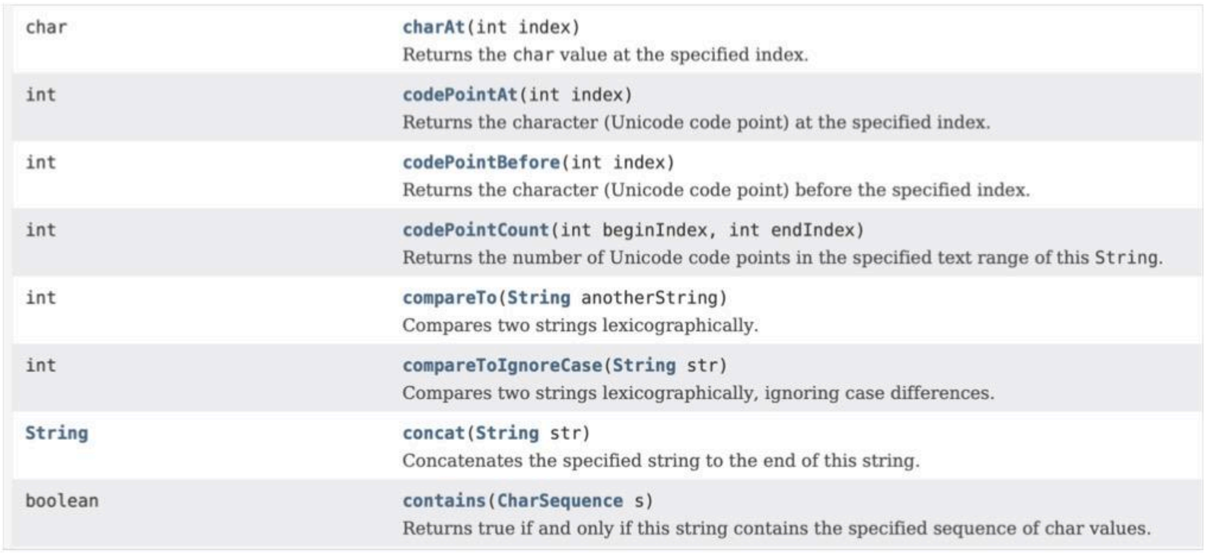
v String concatenation
String data = "feve" + "reiro"; date = 10 + date;
date + = "de" + 2019; System.out.println (data);
v Objects of type String are immutable (constants).
- All methods whose objective is to modify a String in the actually build and return a new String
- The original String remains unchanged.

v **Alternative use of type StringBuilder**

StringBuilder sb = new StringBuilder (); sb.append (10);
sb.append ("feve");
sb.append ("reiro");
sb.append ("de"); sb.append (2019);
String data = sb.toString (); System.out.println (data);
10 February 2019

String class methods

v This class has a set of methods that allow you to perform many operations on text.



char	charAt(int index) Returns the char value at the specified index.
int	codePointAt(int index) Returns the character (Unicode code point) at the specified index.
int	codePointBefore(int index) Returns the character (Unicode code point) before the specified index.
int	codePointCount(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this String.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
String	concat(String str) Concatenates the specified string to the end of this string.
boolean	contains(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.

Character length and access

v The length (number of characters) of a String can be determined using the length method.
v Accessing a character is done with the charAt (int index) method.
v Example:
String s1 = "University of Aveiro"; System.out.println (s1.length ());
for (int i = 0; i <s1.length (); i ++)
System.out.print (s1.charAt (i) + ",");

String Comparison

v Some methods
- equals, equalsIgnoreCase, compareTo
v Examples:
String s1 = "Aveiro";
String s2 = "aveiro";
System.out.println (s1.equals (s2)? "Equals": "Different"); System.out.println (s1.equalsIgnoreCase (s2)? "Equal": "Different"); System.out.println (s1.compareTo (s2));
// <0 (minor s1), 0 (equal),> 0 (major s1)

Comparison of subStrings

v We can analyze parts of a String - contains, substring, startsWith, endsWith, ...
v Examples:
String s1 = "Aveiro";
String s2 = "aveiro";
System.out.println (s1.contains ("ve")); // true System.out.println (s1.substring (1, 3)); // ve System.out.println (s1.startsWith ("ave")); // false System.out.println (s1.endsWith ("ro")); // true

---

Formatting Strings

v The format method returns a new String formatted according to format specifiers.

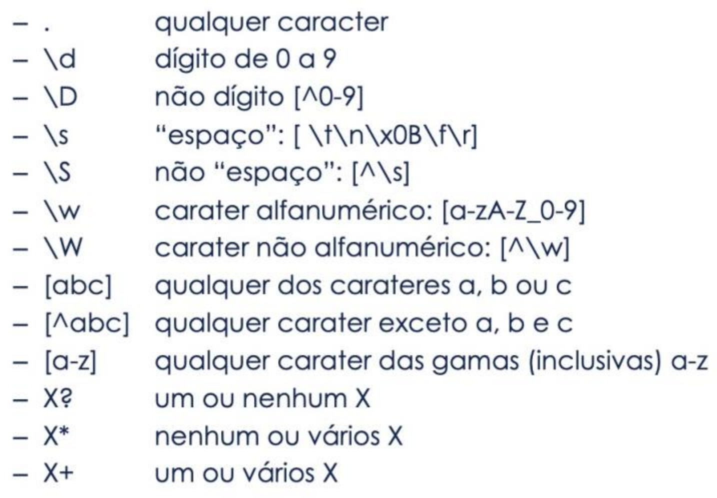
```
long seconds = 347876;
String s1 =
String.format ("% 02d hours,% 02d minutes and% 02d seconds \ n",
        seconds / 3600,
        (3600% seconds) / 60,
        seconds% 60);
System.out.println (s1);
96 hours, 37 minutes and 56 seconds
```

v System.out.printf is a method, alternative to System.out.print, that uses formatting.
v Example:

```
long seconds = 347876;
System.out.printf ("% 02d hours,% 02d minutes and% 02d seconds \ n",
        seconds / 3600,
        (3600% seconds) / 60,
        seconds% 60);
```

Regular expressions (regex)

v Allows you to define patterns that can be searched for in Strings.
- The complete list of supported constructs is described in the documentation for the java.util.regex.Pattern class.
v The matches method of the String class checks whether a String includes standard data.
v Examples:
String s1 = "123"; System.out.println (s1.matches ("\\ d {2,4}"));
// 2-4 digits in a row
s1 = "abcdefg"; System.out.println (s1.matches ("\\ w {3,}"));
// at least 3 alphanumeric characters
true true



– .	qualquer caracter
– \d	dígito de 0 a 9
– \D	não dígito [^0-9]
– \s	"espaço": [\t\n\x0B\f\r]
– \S	não "espaço": [^\s]
– \w	carater alfanumérico: [a-zA-Z_0-9]
– \W	carater não alfanumérico: [^\w]
– [abc]	qualquer dos carateres a, b ou c
– [^abc]	qualquer carater exceto a, b e c
– [a-z]	qualquer carater das gamas (inclusivas) a-z
– X?	um ou nenhum X
– X*	nenhum ou vários X
– X+	um ou vários X

Split method

v The split method separates a String into parts based on a regular expression and returns the resulting Strings vector.

What is a class?

v Classes are specifications for creating objects
v A class represents a complex data type
v Classes describe
- Types of data that make up the object (which can store)
- Methods that the object can perform (what they can do)
v Example:

```
 public class Book {
     String title;
    int pubYear;
 }
```
