05 Numbers And Strings

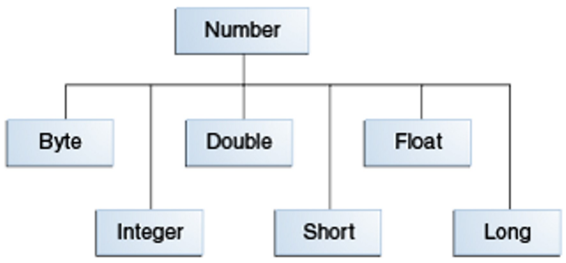
26 May 2024 16:47

Agenda :

1. Numbers
2. Questions and Exercise
3. Characters
4. Strings
5. Autoboxing and Unboxing
6. Questions and Exercise

1. Numbers

- Java platform provides *wrapper* classes for each of the primitive data types.
- Often, the wrapping is done by the compiler—if you use a primitive where an object is expected, the compiler *boxes* the primitive in its wrapper class for you. Similarly, if you use a number object when a primitive is expected, the compiler *unboxes* the object for you.

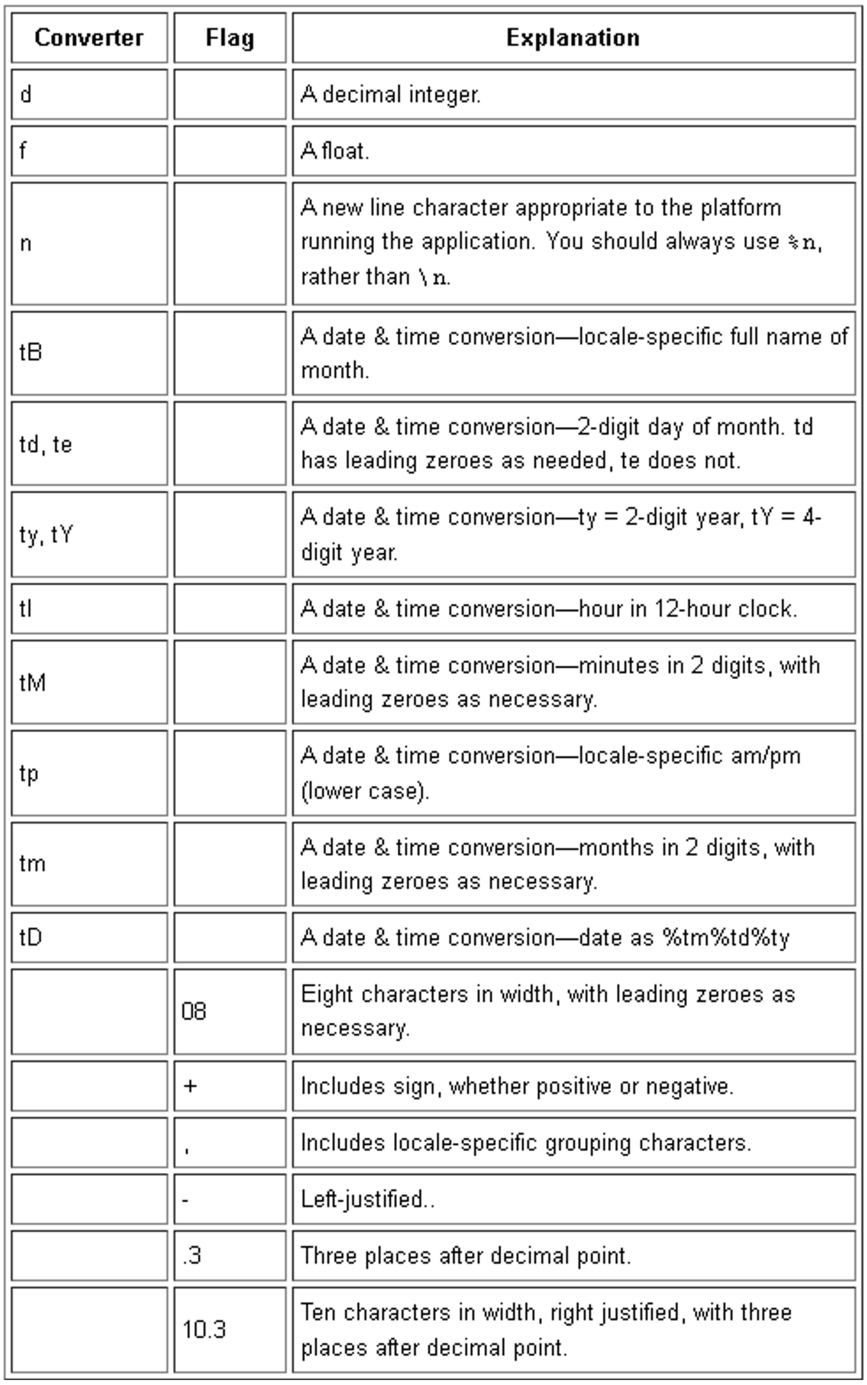


Note: There are four other subclasses of Number. BigDecimal and BigInteger are used for high-precision calculations. AtomicInteger and AtomicLong are used for multi-threaded applications.

Methods Implemented by all subclasses of Number :

https://docs.oracle.com/javase/tutorial/java/data/numberclasses.html#:~:text=Methods%20Implemented%20by%20all%20Subclasses%20of%20Number

Formatting Numeric Print Output



Converter	Flag	Explanation
d		A decimal integer.
f		A float.
n		A new line character appropriate to the platform running the application. You should always use `%n`, rather than `\n`.
tB		A date & time conversion—locale-specific full name of month.
td, te		A date & time conversion—2-digit day of month. td has leading zeroes as needed, te does not.
ty, tY		A date & time conversion—ty = 2-digit year, tY = 4-digit year.
tl		A date & time conversion—hour in 12-hour clock.
tM		A date & time conversion—minutes in 2 digits, with leading zeroes as necessary.
tp		A date & time conversion—locale-specific am/pm (lower case).
tm		A date & time conversion—months in 2 digits, with leading zeroes as necessary.
tD		A date & time conversion—date as %tm%td%ty
	08	Eight characters in width, with leading zeroes as necessary.
	+	Includes sign, whether positive or negative.
	,	Includes locale-specific grouping characters.
	-	Left-justified..
	.3	Three places after decimal point.
	10.3	Ten characters in width, right justified, with three places after decimal point.

```
import java.util.Calendar;
import java.util.Locale;
public class TestFormat {

public static void main(String[] args) {
     long n = 461012;
     System.out.format("%d%n", n);      //  -->  "461012"
     System.out.format("%08d%n", n);    //  -->  "00461012"
     System.out.format("%+8d%n", n);    //  -->  " +461012"
     System.out.format("%,8d%n", n);    // -->  " 461,012"
     System.out.format("%+,8d%n%n", n); //  -->  "+461,012"

     double pi = Math.PI;
     System.out.format("%f%n", pi);       // -->  "3.141593"
     System.out.format("%.3f%n", pi);     // -->  "3.142"
     System.out.format("%10.3f%n", pi);   // -->  "     3.142"
     System.out.format("%-10.3f%n", pi);  // -->  "3.142"
     System.out.format(Locale.FRANCE,
                 "%-10.4f%n%n", pi); // -->  "3,1416"
     Calendar c = Calendar.getInstance();
     System.out.format("%tB %te, %tY%n", c, c, c); // -->  "May 29, 2006"
     System.out.format("%tl:%tM %tp%n", c, c, c);  // -->  "2:34 am"
     System.out.format("%tD%n", c);    // -->  "05/29/06"
```

```
        }
    }
```

DecimalFormat class : https://docs.oracle.com/javase/tutorial/java/data/numberformat.html#:~:text=The%20DecimalFormat%20Class

Basic Maths Methods :
https://docs.oracle.com/javase/tutorial/java/data/beyondmath.html

2. Questions And Exercise

Questions

1. Use the API documentation to find the answers to the following questions:
   a. What Integer method can you use to convert an int into a string that expresses the number in hexadecimal? For example, what method converts the integer 65 into the string "41"?
   b. What Integer method would you use to convert a string expressed in base 5 into the equivalent int? For example, how would you convert the string "230" into the integer value 65? Show the code you would use to accomplish this task.
   c. What Double method can you use to detect whether a floating-point number has the special value Not a Number (Nan)?

2. What is the value of the following expression, and why?
```
Integer.valueOf(1).equals(Long.valueOf(1))
```

Exercises

1. Change MaxVariablesDemo to show minimum values instead of maximum values. You can delete all code related to the variables aChar and aBoolean. What is the output?

2. Create a program that reads an unspecified number of integer arguments from the command line and adds them together. For example, suppose that you enter the following:
```
java Adder 1 3 2 10
```
   The program should display 16 and then exit. The program should display an error message if the user enters only one argument. You can base your program on ValueOfDemo.

3. Create a program that is similar to the previous one but has the following differences:
   - Instead of reading integer arguments, it reads floating-point arguments.
   - It displays the sum of the arguments, using exactly two digits to the right of the decimal point.
     For example, suppose that you enter the following:
```
java FPAdder 1 1e2 3.0 4.754
```
     The program would display 108.75. Depending on your locale, the decimal point might be a comma (,) instead of a period (.)

3. Characters

- Primitive type char is used whenever we want to represent a single character.
  - Ex : **char ch='a';**
- How ever if we want to pass a object for the same, java provides a wrapper class **Character.**
  - Ex : Character ch = new Character('a');
- **The Character class is immutable, so that once it is created, a Character object cannot be changed.**

  Useful Methods in the Character Class :

  https://docs.oracle.com/javase/tutorial/java/data/characters.html#:~:text=Character%20API%20specification.-,Useful%20Methods%20in%20the%20Character%20Class,-Method

  Escape Sequence :

  https://docs.oracle.com/javase/tutorial/java/data/characters.html#:~:text=one%2Dcharacter%20string.-,Escape%20Sequences,-A%20character%20preceded

4. Strings

- Strings are the sequence of characters. The Java platform provides the String class to create and manipulate strings.

  - String str = "Hello World!!"

  - char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
    String helloString = new String(helloArray);
- Some useful methods :
  - **.length()** : returns the length of the string.
  - **.charAt(i) :** returns character present at i'th location of string.
  - **.getChars(0, len, tempCharArray, 0) :** Converts string or a portion of string into array of characters
    String palindrome = "Dot saw I was Tod";
    - int len = palindrome.length();
      char[] tempCharArray = new char[len];
  - **Str1.concat(str2) :** Used to concatenate two strings.

- Converting Strings to Numbers

  - float a = (Float.valueOf(args[0])).floatValue();
  - Int a = (Integer.**valueOf**(args[1])).IntValue();

  Or

  - float a = Float.**parseFloat**(args[0]);

- Converting Numbers to Strings

  - **Int i;**
    String s1 = ""+i;
  - **String s2 = String.valueOf(i);**
  - **String s3 = Integer.toString(i);**

- Substring Methods

  - **String substring(int beginIndex, int endIndex)**

  - **String[] split(String regex, int limit)**

  - **String trim()**

- Other Methods

  - **String toLowerCase()**

  - **String toUpperCase()**

  - **Int indexOf(int ch)**

  - **Int lastIndexOf(String str)**

  - **boolean contains(CharSequence s)**

  - **String replace(char oldChar, char newChar)**

- Comparing Strings

  - **Boolean endsWith(String suffix)**

  - **Boolean startsWith(String Prefix)**

  - **Int compareTo(String anotherString) / int compareToIgnoreCase(String str)**

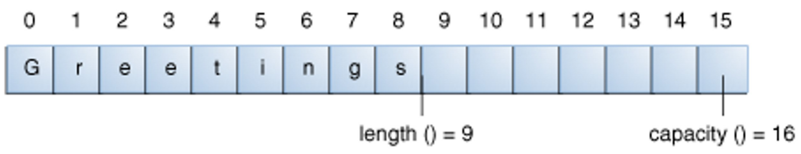
- **Boolean equals(Object anObject) / equalIgnoreCase**
- **Boolean matches(String regex)**

String Builder Class

- String builder objects are like String objects, except that they can be modified.
- Internally these objects are treated like variable length arrays that contains a sequence of characters.

```
// creates empty builder, capacity 16
StringBuilder sb = new StringBuilder();
// adds 9 character string at beginning
sb.append("Greetings");
```

will produce a string builder with a length of 9 and a capacity of 16:



Operations :

- setLength(int len)
- ensureCapacity(int cap)
- Append()
- Delete()
- Insert()
- Replace()
- setCharAt()
- Reverse()
- toString()

5. Autoboxing And Unboxing

- Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.

```
List<Integer> li = new ArrayList<>();
for (int i = 1; i < 50; i += 2)
    li.add(i);
```

- Converting a primitive value (an int, for example) into an object of the corresponding wrapper class (Integer) is called autoboxing. The Java compiler applies autoboxing when a primitive value is:
  - Passed as a parameter to a method that expects an object of the corresponding wrapper class.
  - Assigned to a variable of the corresponding wrapper class.
- Converting an object of a wrapper type (Integer) to its corresponding primitive (int) value is called unboxing. The Java compiler applies unboxing when an object of a wrapper class is:
  - Passed as a parameter to a method that expects a value of the corresponding primitive type.
  - Assigned to a variable of the corresponding primitive type.

Autoboxing and unboxing lets developers write cleaner code, making it easier to read. The following table lists the primitive types and their corresponding wrapper classes, which are used by the Java compiler for autoboxing and unboxing:

Primitive type	**Wrapper class**
boolean	Boolean
byte	Byte

char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

6. Questions and Exercise :

Questions

1. What is the initial capacity of the following string builder?
   ```
   StringBuilder sb = new StringBuilder("Able was I ere I saw Elba.");
   ```
2. Consider the following string:
   ```
   String hannah = "Did Hannah see bees? Hannah did.";
   ```
   a. What is the value displayed by the expression hannah.length()?
   b. What is the value returned by the method call hannah.charAt(12)?
   c. Write an expression that refers to the letter b in the string referred to by hannah.
3. How long is the string returned by the following expression? What is the string?
   ```
   "Was it a car or a cat I saw?".substring(9, 12)
   ```
4. In the following program, called ComputeResult, what is the value of result after each numbered line executes?
   ```
   public class ComputeResult {
      public static void main(String[] args) {
         String original = "software";
         StringBuilder result = new StringBuilder("hi");
         int index = original.indexOf('a');

   /*1*/   result.setCharAt(0, original.charAt(0));
   /*2*/   result.setCharAt(1, original.charAt(original.length()-1));
   /*3*/   result.insert(1, original.charAt(4));
   /*4*/   result.append(original.substring(1,4));
   /*5*/   result.insert(3, (original.substring(index, index+2) + " "));

   System.out.println(result);
      }
   }
   ```

Exercises

1. Show two ways to concatenate the following two strings together to get the string "Hi, mom.":
   ```
   String hi = "Hi, ";
   String mom = "mom.";
   ```
2. Write a program that computes your initials from your full name and displays them.
3. An anagram is a word or a phrase made by transposing the letters of another word or phrase; for example, "parliament" is an anagram of "partial men," and "software" is an anagram of "swear oft." Write a program that figures out whether one string is an anagram of another string. The program should ignore white space and punctuation.