I'm not robot
reCAPTCHA
Continue

Java string format null value

During work with StringBuilder you might have a strange hanging behavior() method for null values. If you hang a null value to the StringBuilder object, it would be stored as a "null" (four character strings) in the object instead of no value. Let's take an example to understand what I said above. Problem class AppendNullDemo{ public static network (String args[]){ // string array with few null elements String str[] = {" Mango ", null, " Orange ", " Banana ", null}; // Create StringBuilder StringBuilder sb = new StringBuilder(); for (String temp: str){ sb.append(temp); } // Displaying output System.out.println(sb); Exit: Mango null Banana orange null As you can see that null values are added as "null" strings in output, however we prefer not to have values for null elements. The correction of this problem is provided in the next program. Solution: Here's the solution of the problem above. class AdppendNullSolution{ public static void main(String args[]){ // string array with few null elements String str[] = {" Mango ", null, " Orange ", " Banana ", null}; // Create StringBuilder StringBuilder sb = new StringBuilder(); for (String temp: str){ // Hang only when the string is nothing if (temp!= null) { sb.append(temp); } // View of the output System.out.println(sb); Exit: Mango Banana Orange Fixed Problem: null strings are deleted from output. The reason for the problem? The reason is explained inby javadoc: public StringBuilder append(String str): Adds the specified string to this character sequence. String characters are added, in order, increasing the length of this sequence for the length of the topic. If str is null, the four characters “null” are attached. Have you tried to read and understand the documentation of the String format of Java? I found it almost impenetrable. While it includes all the information, the organization leaves something to be desired. This guide is an attempt to bring some clarity and facilitate the use of string formatting in Java. String Formatting The most common way to format a java string is using String.format(). If there was a “java sprintf” then it would be this. String output = String.format("%s = %d", "joe", 35); For the output of the formatted console, you can use printf() or the format() method of System.out and System.err PrintStreams. System.out.printf("My name is: %s%n", "joe"); Create a Formatter and connect it to a StringBuilder. The output formatted using the format() method will be attached to the StringBuilder. StringBuilder sbuf = new StringBuilder(); Formatter fmt = new Formatter(sbuf); fmt.format("PI = %f%n", Math.PI); System.out.print(sbuf.toString()); // you can continue to add data to sbuf here. Format Specifiers Here is a quick reference to all supported conversion specifications: Specification Applies to output % floating point (excluding BigDecimal) Hexof the mobile point number %b Any type of “true” if not null, “false” if no character Unicode character %d integer (incl byte, short, int, long, bigint) Decimal Integer %e decimal number at variable point in scientific notation %f decimal number at variable point %g decimal number, possibly in scientific notation depending on accuracy and value. %h any type of Hex String value from the hashCode(). %n no specific line separator of the platform. %o whole (incl. bytes, short, int, long, bigint) The number of octal%s of any type String value %t Data/Time (incl. long, Calendar, Date and TemporalAccessor) %t is the prefix for date/hour conversions. Multiple formatting flags are needed after this. See the Data/Now conversion below. %x whole (incl. byte, short, int, long, bigint) The hexagonal rope. Date and Time Formatting Notes: Using formatting characters with "%T" instead of "%t" in the table below makes the output capital. Flag Notes %tA Full name of the day of the week, such as “Sunday“, “Monday“ %ta Short name of the day of the week, for example “Sun“, “Mon“, etc. %B Full name of the month for example “January“, “February“, etc. %tb Name of the abbreviated month, for example “Jan“, “Feb“, etc. %tC Century part of the year formatted with two digits, for example “00” through “99”. % Date and time formatted with “%ta %tb %td %tT %tZ %tZ %tY” for example “Fribb 17 07:45:42 PST 2017“%tD Date formatted as “%tm/%td/%td/%ty“ %td Month Daywith two digits. for example “01” to “31“. % Day of the month formatted without a leader 0 for example “1” to “31”. %tF ISO 8601 formatted date with “%tY-%tm-%td“. % Hours of the day for the clock 24 hours, for example “00” to “23“. % equal to %tb. %TI Hours of the day for the 12-hour clock, for example “01” – “12“. % Day of the year formatted with 0 main, for example “001” to “366“. % Hours of the day for the clock 24 hours without a leader 0 for example “0” to “23“. % Hours of the day for 12 hours without a main 0 for example “1” to “12“. %tM Minute within the hour formatted a leader 0 for example “00” to “59“. %tm Mese formatted with a leader 0 for example “01” to “12“. %tN Nanosecond formatted with 9 main digits and 0, for example “000000000” to “999999999“. % Local specific “am” or “pm” marker. %tQ Milliseconds since the 1 Jan 1970 00:00:00:00:00 UTC. %tR Formatted time as 24 hours for example “%tH:%tM“. %tr Formatted time as 12 hours, e.g. “%tI:%tM:%tS %Tp“. %tS Second in the minute formatted with 2 digits, for example “00” to “60”. “60” is necessary to support second bisexuals. % Seconds from January 1, 1970 00:00:00 UTC. %tT Formatted time as 24 hours for example “%tH:%tM:%tS“. % Year formatted with 4 digits, for example “0000” to “9999“. %ty Year formatted with 2 digits, for example “00” to “99“. %tZ Abbreviation of the time zone. for example “UTC“, “PST“, etc. %tz Time Zone Offset from GMT for example “-0800“.Index An index of the subject is specified as a number ending with a "$" after "%" and selecting the argument specified in the list of arguments. String.format("%2$s", 32, "Hello"); // prints: "Hello" Formatting an Integer With the %d format specifier, you can use an argument of all integral types including byte, short, int, long and BigInteger. Default format: String.format("%d", 93); // prints 93 Specifying a width: String.format("|%20d|", 93); // prints: | 93| To the left which adjusts in the specified width: String.format("|%-20d|", 93); // prints: |93 | Pad with zero: String.format("|%020d|", 93); // prints: |00000000000