JAVA基础：Java程序的脏数据问题 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_JAVA_E5_9F_BA_E7_A1_80_c104_145283.htm 脏数据(Out-of-date data)，指过时的数据。 如果在您的Java程序中存在脏数据，将或多或少地给软件系统带来一些问题，如：无法实时地应用已经发生改变的配置，软件系统出现一些莫名其妙的、难以重现的、后果严重的错误等等。尽量避免脏数据的存在是非常有价值的。本文希望能在这方面给同行们一点帮助。 Fragment 1. 缓存技术的脏数据问题 /** * A report printer is used to print a report. * * @version 1.0 9/9/2003 * @author Bill */ public class ReportPrinter { /** * Constructs a ReportPrinter instance. */ public ReportPrinter() { // do something... } /** * Prints a printable. * * @param printable the specified printable object */ public void print(Printable printable) { Graphics g = getGraphics(). g.setFont(getReportFont(printable.getFont()). printable.print(g). } /** * Returns the corresponding report font of a java font. * * @param javaFont the specified java font * @return the corresponding report font */ private Font getReportFont(font javaFont) { Font reportFont = fontMap.get(javaFont). if(reportFont == null) { reportFont = loadFont(javaFont). fontMap.put(javaFont, reportFont). } return reportFont. } /** * Loads the corresponding report font of a java font. * * @param javaFont the specified java font * @param the corresponding report font */ protected static Font loadFont(Font javaFont) { Font reportFont = null. // do something... return reportFont. } /** * The font map(java

font->report font). */ private static HashMap fontMap = new HashMap(). } Fragment 1中，由于装载一个java font所对应的report font开销较大，使用了缓存技术来避免这种开销。这是一种常见的提高性能的方式，而且在一般情况下运行良好。但是Fragment 1的设计与实现可能是不完备的，因为极有可能一个java font所对应的report font在系统启动之后发生变化，在这种变化发生之后，只有重启软件系统才能装载之，这常常是最终用户的抱怨之一。更可怕的是，类似的这种脏数据的存在还可能带来其它严重的、无法想象的后果。