the bug detected :

If the denominator to a division or modulo operation is zero it would result in a fatal error.

When working with double or float, no fatal error will be raised, but it will lead to unusual result and should be avoided anyway.

This rule supports primitive int, long, double, float as well as BigDecimal and BigInteger.

suggested not yet confirmed solution is to make an exception for denominator ==0 .

the old code :

private ClientAnchorDetail calculateColumnLocation(HSSFSheet sheet, int startingColumn, double reqImageWidthMM) {

ClientAnchorDetail anchorDetail;

double totalWidthMM = 0.0D;

double colWidthMM = 0.0D;

double overlapMM;

double coordinatePositionsPerMM;

int toColumn = startingColumn;

int inset;

while (totalWidthMM < reqImageWidthMM) {

colWidthMM = ConvertImageUnits.widthUnits2Millimetres((short)(sheet.getColumnWidth(toColumn)));

totalWidthMM += (colWidthMM + ConvertImageUnits.CELL\_BORDER\_WIDTH\_MILLIMETRES);

toColumn++;

}

toColumn--;

if((int)totalWidthMM == (int)reqImageWidthMM) {

anchorDetail = new ClientAnchorDetail(startingColumn, toColumn, ConvertImageUnits.TOTAL\_COLUMN\_COORDINATE\_POSITIONS);

} else {

overlapMM = reqImageWidthMM - (totalWidthMM - colWidthMM);

if(overlapMM < 0) {

overlapMM = 0.0D;

}

coordinatePositionsPerMM = ConvertImageUnits.TOTAL\_COLUMN\_COORDINATE\_POSITIONS / colWidthMM;

Make sure "colWidthMM" can't be zero before doing this division.

inset = (int)(coordinatePositionsPerMM \* overlapMM);

anchorDetail = new ClientAnchorDetail(startingColumn, toColumn, inset);

}

return(anchorDetail);

}

private ClientAnchorDetail fitImageToRows(HSSFSheet sheet, int rowNumber, double reqImageHeightMM, int resizeBehaviour) {

double rowCoordinatesPerMM;

int pictureHeightCoordinates;

the new code:

private ClientAnchorDetail calculateColumnLocation(HSSFSheet sheet, int startingColumn,double reqImageWidthMM) {

ClientAnchorDetail anchorDetail;

double totalWidthMM = 0.0D;

double colWidthMM = 0.0D;

double overlapMM;

double coordinatePositionsPerMM;

int toColumn = startingColumn;

int inset;

while (totalWidthMM < reqImageWidthMM) {

colWidthMM = ConvertImageUnits.widthUnits2Millimetres((short)(sheet.getColumnWidth(toColumn)));

totalWidthMM += (colWidthMM + ConvertImageUnits.CELL\_BORDER\_WIDTH\_MILLIMETRES);

toColumn++;

}

toColumn--;

if((int)totalWidthMM == (int)reqImageWidthMM) {

anchorDetail = new ClientAnchorDetail(startingColumn, toColumn, ConvertImageUnits.TOTAL\_COLUMN\_COORDINATE\_POSITIONS);

} else {

overlapMM = reqImageWidthMM - (totalWidthMM - colWidthMM);

if(overlapMM < 0) {

overlapMM = 0.0D;

}

//To make sure "colWidthMM" can't be zero before doing this division.

try

{

coordinatePositionsPerMM = ConvertImageUnits.TOTAL\_COLUMN\_COORDINATE\_POSITIONS / (int)colWidthMM;

inset = (int)(coordinatePositionsPerMM \* overlapMM);

anchorDetail = new ClientAnchorDetail(startingColumn, toColumn, inset);

}

catch (ArithmeticException e) {

System.out.println("ArithmeticException => " +"division by zero!");

inset =0;

anchorDetail = new ClientAnchorDetail(startingColumn, toColumn, inset);

}

}return(anchorDetail);

}

Motivation :

the motivation behind fixing the error is to ensure that the code run as expected . and controlling user input by adding try&catch method to the code shown . as entering 0 as a denominator will result in fatal logical error .

correcting this logical error will result in more consistency and preventing error will cause the program to be more efficient . as a team we ensure that.