Hello,

I've reviewed your questions and created examples along with detailed explanations for each of them. These examples include code snippets to demonstrate the solutions. The code has been written in Java, using iText version 8.0.5, and is provided as a compressed archive for your convenience.
If you're using a different version of iText or need further assistance, feel free to reach out—I'll be happy to help!

## Question 1:

Is there a way to adjust the opacity, size and text colour of the watermark so that it doesn't interfere with existing content?

*Adjusting the opacity, size, and color of a watermark text to ensure it doesn't interfere with existing content is not only possible but also straightforward with iText. This powerful library provides tools and methods that make the process simple and flexible.*
*The watermark text can be created as a* `Paragraph` *object, which is then rendered on the* `PdfCanvas` *with its appearance customized through properties like font size, color, and rotation. To control transparency, the* `PdfExtGState` *class is used, ensuring that the opacity affects only the watermark, leaving the rest of the content untouched. Additionally, the use of* `saveState()` *and* `restoreState()` *methods isolates the watermark's properties, preventing any unintended changes to other graphical elements in the document.*
*In the following sections, we'll demonstrate how to achieve this step by step using iText 8.0.5 in Java, with examples that showcase how to configure and apply these properties effectively.*

### *1. Create the watermark text*

*To create the watermark text, we use the* `Paragraph` *class. This object encapsulates the text and its visual properties, such as font size, color, and rotation.*

### *Creating the Paragraph Object:*

*We create the watermark text as an instance of the* `Paragraph` *class and define its properties.*
*The constructor takes the watermark text as a parameter, and additional methods allow us to configure its appearance.*
*The font size is adjusted using the* `setFontSize()` *method, and the color is set with* `setFontColor()`*. These methods take values like the size in points (50 in the example) and a predefined constant for the color (*`ColorConstants.GRAY`*).*

### *Explained Parameters:*

- `"Watermark Example"`: *The actual text to be used as the watermark.*
- `ColorConstants.GRAY`*: A predefined constant for gray color in the iText library.*
- `50`*: Font size in points, adjustable to meet specific design needs.*
- `Math.toRadians(45)`*: Converts the rotation angle from degrees to radians, as the* `setRotationAngle()` *method requires radians. (There will be more information about the rotation in question 3)*

```
Paragraph watermarkText = new Paragraph( text: "Watermark Example")
        .setFontColor(ColorConstants.GRAY)
        .setFontSize(50)
        .setRotationAngle(Math.toRadians(45) );
```

***Note**: The* `Paragraph` *object is later passed to the* `Canvas.showTextAligned()` *method for rendering on the page.*

***2. Adjust the opacity***

*Opacity is not a property of the* `Paragraph` *itself but is applied to the graphical context of the page using the* `PdfExtGState` *class. This allows us to control the transparency of the watermark text.*

***Creating the*** `PdfExtGState` ***Object**:*

- *We instantiate a* `PdfExtGState` *object to define the opacity level for the text.*
- *The* `setFillOpacity()` *method is used to set the opacity, where the value ranges from 0 (fully transparent) to 1 (fully opaque).*

***Applying the Opacity to the*** `PdfCanvas`*:*

- *The* `PdfExtGState` *is applied to the* `PdfCanvas` *using the* `setExtGState()` *method.*
- *Before making any changes, the current graphic state is saved using* `saveState()`*. This ensures that the opacity setting will only affect the watermark and not any other content.*
- *After adding the watermark text, the original graphic state is restored with* `restoreState()`*.*

```
float opacity = 0.5f;

under.saveState();
PdfExtGState gs1 = new PdfExtGState();
gs1.setFillOpacity(opacity);
under.setExtGState(gs1);

canvas.showTextAligned(
        watermarkText,
        x, y,
        TextAlignment.CENTER, VerticalAlignment.TOP
);

under.restoreState();
```

***Explained Parameters**:*

- `0.5f`*: Sets the opacity level to 50%. This value can be adjusted to suit different needs.*
- `under`*: The* `PdfCanvas` *object used to draw the watermark on the page.*

***3. Details About Canvas Creation (*`PdfCanvas`*)***

*The* `PdfCanvas` *is the graphical context used to draw directly on the PDF. For a watermark, a* `PdfCanvas` *is created for each page of the document.*

***Using*** `newContentStreamBefore()`*:*

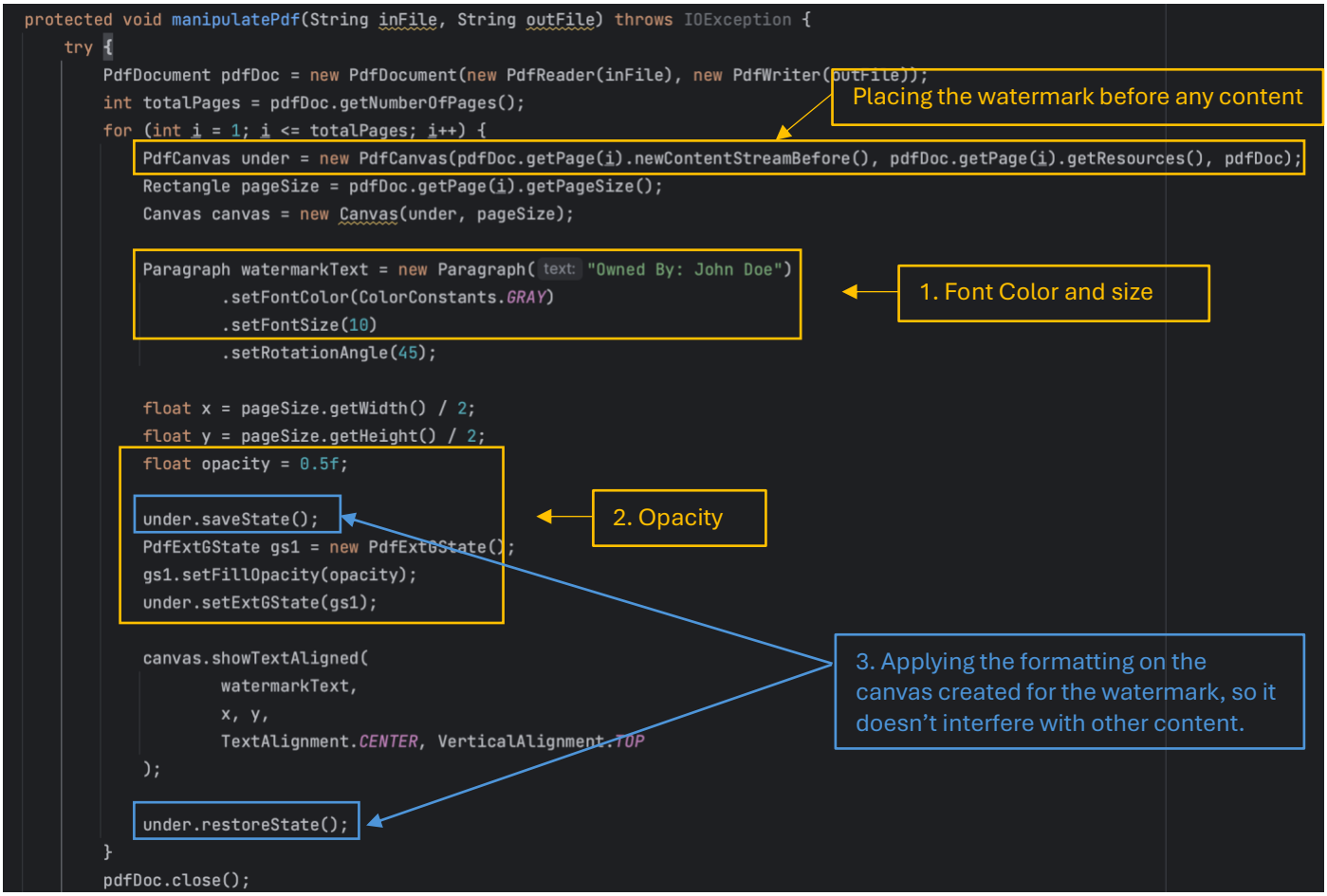
- *The method* `newContentStreamBefore()` *ensures that the watermark content is rendered* ***before*** *the main content of the page.*
- *This placement prevents the watermark from obstructing the existing text or images.*

```
PdfCanvas under = new PdfCanvas(pdfDoc.getPage(i).newContentStreamBefore(), pdfDoc.getPage(i).getResources(), pdfDoc);
```

***Explained Parameters**:*

- `pdfDoc.getPage(i).newContentStreamBefore()`*: Creates a content stream before the main page content.*
- `pdfDoc.getPage(i).getResources()`*: Retrieves the resources (e.g., fonts, colors) associated with the page.*
- `pdfDoc`*: The PDF document being manipulated.*

*Below, you can see the manipulatePdf method, which demonstrates the watermark formatting process, including setting up the content stream, configuring opacity, and applying the watermark to the PDF pages.*


protected void manipulatePdf(String inFile, String outFile) throws IOException {
try {
PdfDocument pdfDoc = new PdfDocument(new PdfReader(inFile), new PdfWriter(outFile));
int totalPages = pdfDoc.getNumberOfPages();
for (int i = 1; i <= totalPages; i++) {
PdfCanvas under = new PdfCanvas(pdfDoc.getPage(i).newContentStreamBefore(), pdfDoc.getPage(i).getResources(), pdfDoc);
Rectangle pageSize = pdfDoc.getPage(i).getPageSize();
Canvas canvas = new Canvas(under, pageSize);
Paragraph watermarkText = new Paragraph( text: "Owned By: John Doe")
.setFontColor(ColorConstants.GRAY)
.setFontSize(10)
.setRotationAngle(45);
float x = pageSize.getWidth() / 2;
float y = pageSize.getHeight() / 2;
float opacity = 0.5f;
under.saveState();
PdfExtGState gs1 = new PdfExtGState();
gs1.setFillOpacity(opacity);
under.setExtGState(gs1);
canvas.showTextAligned(
watermarkText,
x, y,
TextAlignment.CENTER, VerticalAlignment.TOP
);
under.restoreState();
}
pdfDoc.close();
Placing the watermark before any content
1. Font Color and size
2. Opacity
3. Applying the formatting on the canvas created for the watermark, so it doesn't interfere with other content.

**Question 2:**
How can I rotate text along the center point to make it appear diagonal across the page at a 45° angle? I tried adding a rotation parameter, but the text is rotated too much.

*To rotate text diagonally across the page at a 45° angle, you can use the* `.setRotationAngle()` *method in iText 8. This method allows you to rotate text at precise angles and is particularly useful for creating diagonal watermarks or other visually striking design elements in PDF documents.*
*It's important to note that the angle must be passed in radians, which can be easily converted from degrees using* `Math.toRadians()`*. By combining this method with proper text alignment and placement, you can ensure the text is positioned and rotated exactly as intended.*

*To rotate text diagonally across the page at a 45° angle, you can follow this example:*

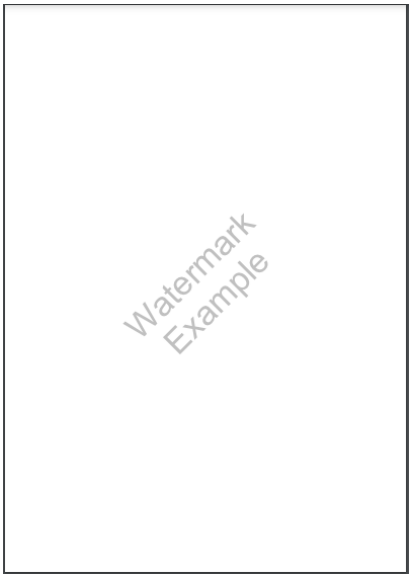
```
Paragraph watermarkText = new Paragraph( text: "Watermark Example")
        .setFontColor(ColorConstants.GRAY)
        .setFontSize(50)
        .setRotationAngle(Math.toRadians(45) );

float x = pageSize.getWidth() / 2;
float y = pageSize.getHeight() / 2;

float opacity = 0.5f;

under.saveState();
PdfExtGState gs1 = new PdfExtGState();
gs1.setFillOpacity(opacity);
under.setExtGState(gs1);

canvas.showTextAligned(
        watermarkText,
        x, y,
        TextAlignment.CENTER, VerticalAlignment.MIDDLE
);
```

Accepts the rotation in radians

***Important Details***

- *Anchor Point for Rotation:The text rotates around the point specified in* `showTextAligned()`*. In this example, the text rotates around the center of the page.*
- *Coordinates Remain the Same:Even with rotation, the x and y coordinates remain unchanged and represent the anchor point.*

*Output example at 45° angle:*

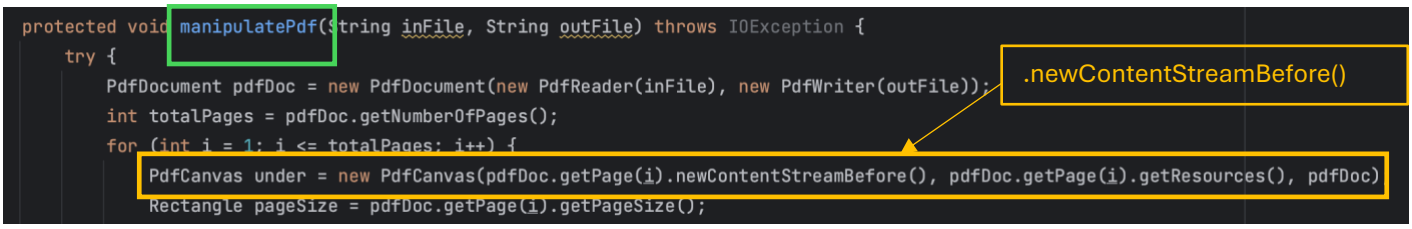



*Output example at 75° angle:*

## Question 3:

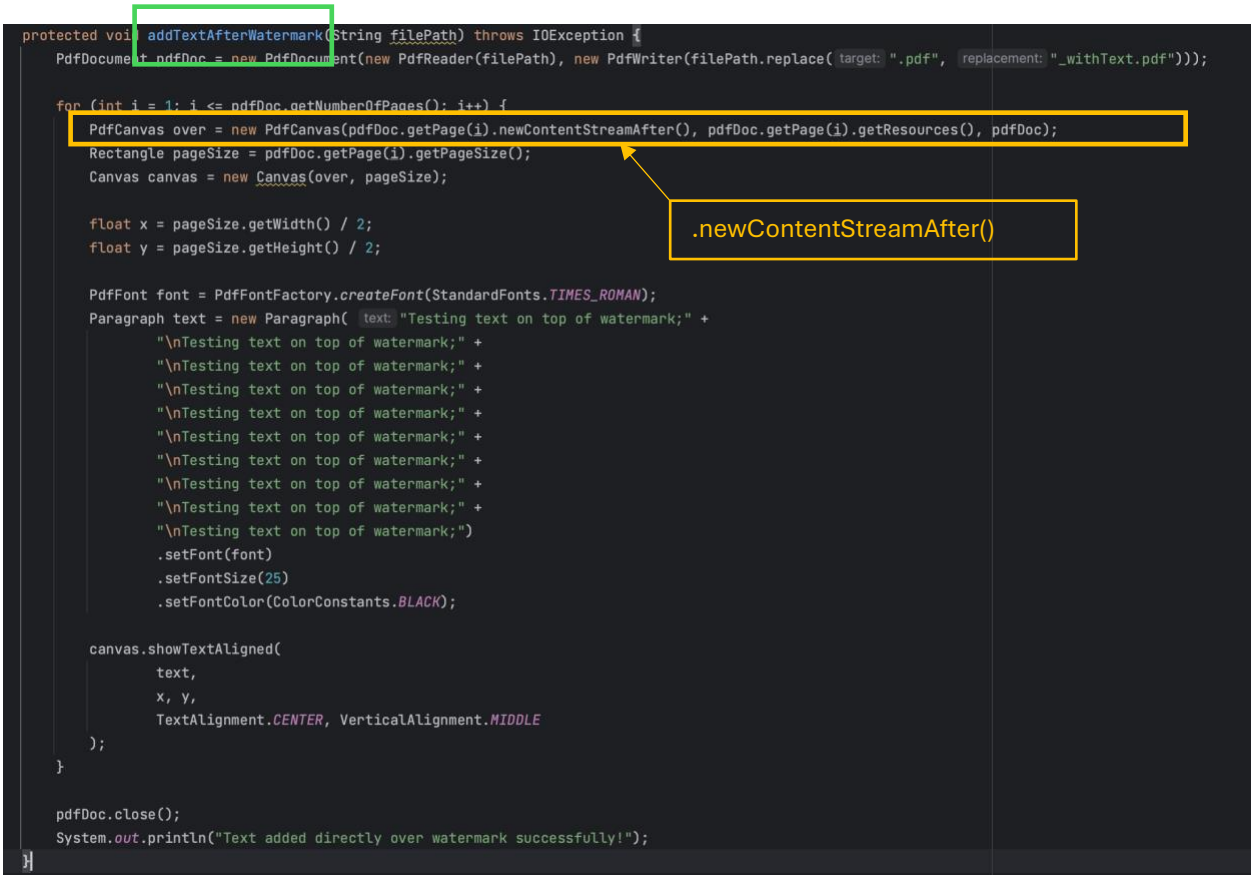
Can the watermark be placed beneath the existing content instead of over it?

*Yes, with iText, it's possible to control whether the watermark appears beneath or over the existing content by using specific methods. To place the watermark beneath the existing content, you can use the* `newContentStreamBefore()` *method. This method ensures that the watermark is added before any other content on the page. In the example, I created a* `PdfCanvas` *and named it under to indicate that this canvas is used to render the watermark below the content.*




```
protected void manipulatePdf(String inFile, String outFile) throws IOException {
    try {
        PdfDocument pdfDoc = new PdfDocument(new PdfReader(inFile), new PdfWriter(outFile));
        int totalPages = pdfDoc.getNumberOfPages();
        for (int i = 1; i <= totalPages; i++) {
            PdfCanvas under = new PdfCanvas(pdfDoc.getPage(i).newContentStreamBefore(), pdfDoc.getPage(i).getResources(), pdfDoc)
            Rectangle pageSize = pdfDoc.getPage(i).getPageSize();
```

*Additionally, I included a method in the sample code to add text on top of the watermark. In this case, the* `newContentStreamAfter()` *method is used to ensure that the new text is rendered over the watermark. This flexibility allows you to precisely control the layering of elements in the PDF.*




```
protected void addTextAfterWatermark(String filePath) throws IOException {
    PdfDocument pdfDoc = new PdfDocument(new PdfReader(filePath), new PdfWriter(filePath.replace( target: ".pdf",  replacement: "_withText.pdf")));

    for (int i = 1; i <= pdfDoc.getNumberOfPages(); i++) {
        PdfCanvas over = new PdfCanvas(pdfDoc.getPage(i).newContentStreamAfter(), pdfDoc.getPage(i).getResources(), pdfDoc);
        Rectangle pageSize = pdfDoc.getPage(i).getPageSize();
        Canvas canvas = new Canvas(over, pageSize);

        float x = pageSize.getWidth() / 2;
        float y = pageSize.getHeight() / 2;

        PdfFont font = PdfFontFactory.createFont(StandardFonts.TIMES_ROMAN);
        Paragraph text = new Paragraph( text: "Testing text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;" +
                "\nTesting text on top of watermark;")
                .setFont(font)
                .setFontSize(25)
                .setFontColor(ColorConstants.BLACK);

        canvas.showTextAligned(
                text,
                x, y,
                TextAlignment.CENTER, VerticalAlignment.MIDDLE
        );
    }

    pdfDoc.close();
    System.out.println("Text added directly over watermark successfully!");
}
```

## Question 4:

Can the watermarking be automated so it will draw the watermark whenever a new page is started or ended? Additionally, We'd like to position this watermark in the center of the page, regardless of what size the page may have.

*Yes, watermarking can be automated to ensure the watermark is drawn whenever a new page is started or added.*
*In the initial example presented in this document, the watermark was applied to each page using a* `for loop` *that iterates through all the pages of the document. This approach allows positioning of the watermark at the center of each page, regardless of the page size.*
*However, this method requires manually iterating through the pages, which is less scalable for applications generating a larger number of pages.*

***A Better Solution: Using an Event Handler***

*An* ***Event Handler*** *is a more scalable solution that automates watermarking by triggering the watermark addition whenever a new page is created or completed. This approach eliminates the need for manual iteration and ensures the watermark is consistently applied across all pages.*
*The* ***formatting structure*** *of the watermark remains the same as in the* `for loop` *example. The center position of the watermark (x and y coordinates), its opacity, rotation, font size, and color are all configured in the same way. However, instead of iterating through pages, the* `handleEvent()` *method of the event handler takes care of applying the watermark dynamically.*

***How the Event Handler Works***

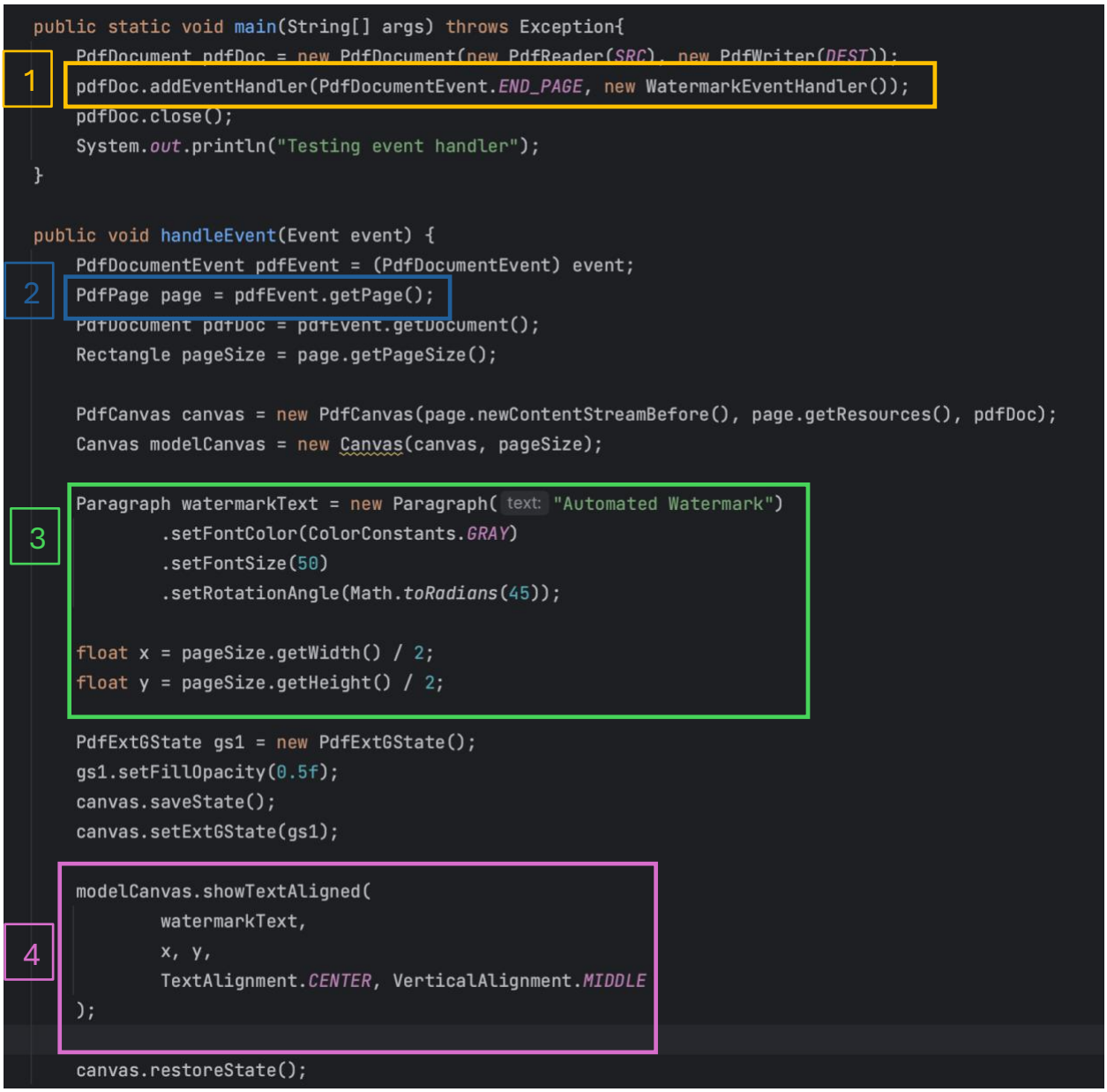
1. ***Triggering the Event****:*
   *The* `addEventHandler` *method associates the* `WatermarkEventHandler` *class with the* `PdfDocumentEvent.END_PAGE` *event. This ensures that the* `handleEvent()` *method is executed for every page at the time it is finalized.*

2. ***Accessing Page Details****:*
   *Inside* `handleEvent()`*, the current page is accessed using* `PdfDocumentEvent.getPage()` *and its size is determined with* `getPageSize()`*.*

3. ***Watermark Formatting****:*
   *The watermark formatting (font size, color, opacity, and rotation) is set up in the same way as in the* `for loop` *example.*
   *The center position is calculated for every page*

### 4. *Adding the Watermark:*

*The watermark is applied to the page using a* `PdfCanvas` *and aligned with* `showTextAligned()`.




```
public static void main(String[] args) throws Exception{
    PdfDocument pdfDoc = new PdfDocument(new PdfReader(SRC), new PdfWriter(DEST));
    pdfDoc.addEventHandler(PdfDocumentEvent.END_PAGE, new WatermarkEventHandler());
    pdfDoc.close();
    System.out.println("Testing event handler");
}

public void handleEvent(Event event) {
    PdfDocumentEvent pdfEvent = (PdfDocumentEvent) event;
    PdfPage page = pdfEvent.getPage();
    PdfDocument pdfDoc = pdfEvent.getDocument();
    Rectangle pageSize = page.getPageSize();

    PdfCanvas canvas = new PdfCanvas(page.newContentStreamBefore(), page.getResources(), pdfDoc);
    Canvas modelCanvas = new Canvas(canvas, pageSize);

    Paragraph watermarkText = new Paragraph( text: "Automated Watermark")
            .setFontColor(ColorConstants.GRAY)
            .setFontSize(50)
            .setRotationAngle(Math.toRadians(45));

    float x = pageSize.getWidth() / 2;
    float y = pageSize.getHeight() / 2;

    PdfExtGState gs1 = new PdfExtGState();
    gs1.setFillOpacity(0.5f);
    canvas.saveState();
    canvas.setExtGState(gs1);

    modelCanvas.showTextAligned(
            watermarkText,
            x, y,
            TextAlignment.CENTER, VerticalAlignment.MIDDLE
    );

    canvas.restoreState();
```