APACHE POI PPT - PRESENTATION

Generally, we use MS-PowerPoint to create presentations. Now let us see how to create presentations using Java. After completion of this chapter, you will be able to create new MS-PowerPoint presentations and open existing PPTs with your Java program.

Creating Empty Presentation

To create an empty presentation, you have to instantiate the **XMLSlideShow** class of the *org.poi.xslf.usermodel* package:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Save the changes to a PPT document using the **FileOutputStream** class:

```
File file=new File("C://POIPPT//Examples//example1.pptx");
FileOutputStream out = new FileOutputStream(file);
ppt.write(out);
```

Given below is the complete program to create a blank MS-PowerPoint presentation.

```
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class CreatePresentation {

   public static void main(String args[]) throws IOException{

      //creating a new empty slide show
      XMLSlideShow ppt = new XMLSlideShow();

      //creating an FileOutputStream object
      File file =new File("example1.pptx");
      FileOutputStream out = new FileOutputStream(file);

      //saving the changes to a file
      ppt.write(out);
      System.out.println("Presentation created successfully");
      out.close()
   }
}
```

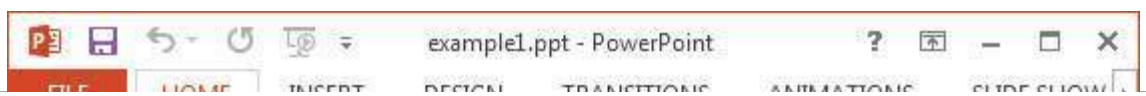
Save the above Java code as **CreatePresentation.java**, and then compile and execute it from the command prompt as follows:

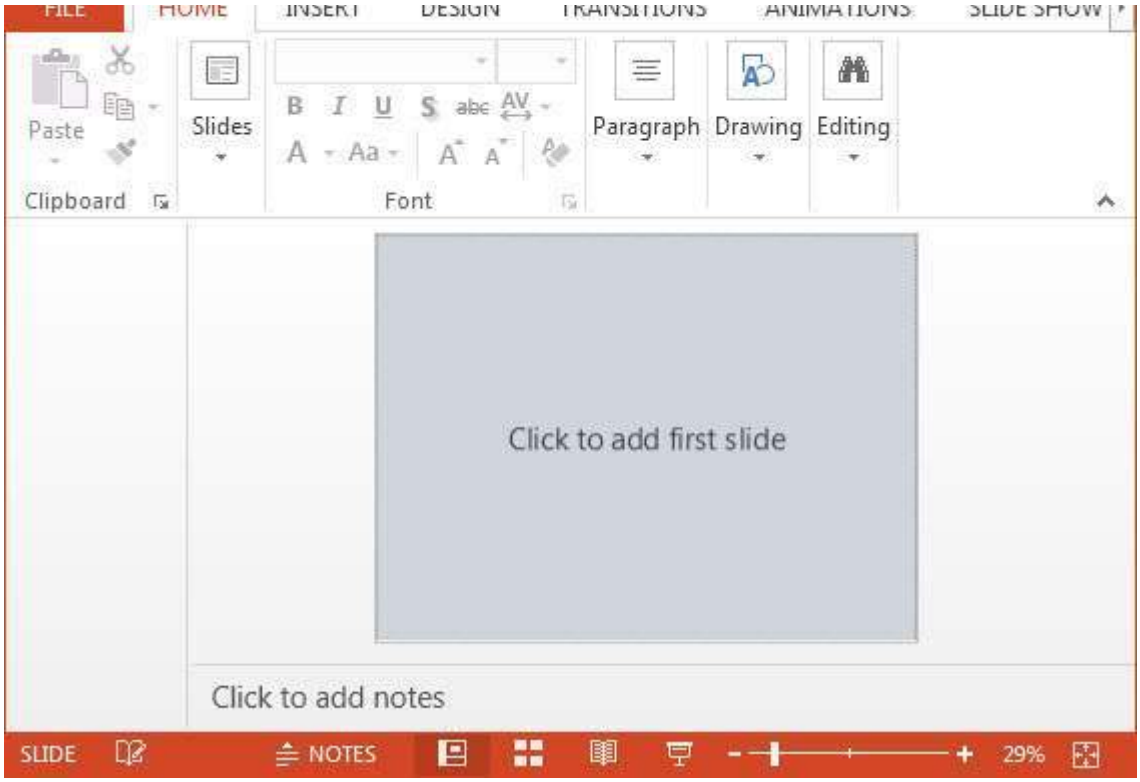
```
$javac CreatePresentation.java
$java CreatePresentation
```

If your system environment is configured with the POI library, it will compile and execute to generate a blank PPT file named **example1.pptx** in your current directory and display the following output on the command prompt:

```
Presentation created successfully
```

The blank PowerPoint document appears as follows:





Editing an Existing Presentation

To open an existing presentation, instantiate the **XMLSlideShow** class and pass the **FileInputStream** object of the file to be edited, as an argument to the **XMLSlideShow** constructor.

```
File file=new File("C://POIPPT//Examples//example1.pptx");
FileInputstream inputstream =new FileInputStream(file);
XMLSlideShow ppt = new XMLSlideShow(inputstream);
```

You can add slides to a presentation using the **createSlide** method of the XMLSlideShow class which is in the *org.poi.xslf.usermodel* package.

```
XSLFSlide slide1= ppt.createSlide();
```

Given below is the complete program to open and add slides to an existing PPT:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class EditPresentation {

   public static void main(String ar[]) throws IOException{

      //opening an existing slide show
      File file = new File("example1.pptx");
      FileInputStream inputstream=new FileInputStream(file);
      XMLSlideShow ppt = new XMLSlideShow(inputstream);

      //adding slides to the slodeshow
      XSLFSlide slide1 = ppt.createSlide();
      XSLFSlide slide2 = ppt.createSlide();

      //saving the changes
      FileOutputStream out = new FileOutputStream(file);
```

```
        ppt.write(out);

        System.out.println("Presentation edited successfully");
        out.close();
    }
}
```

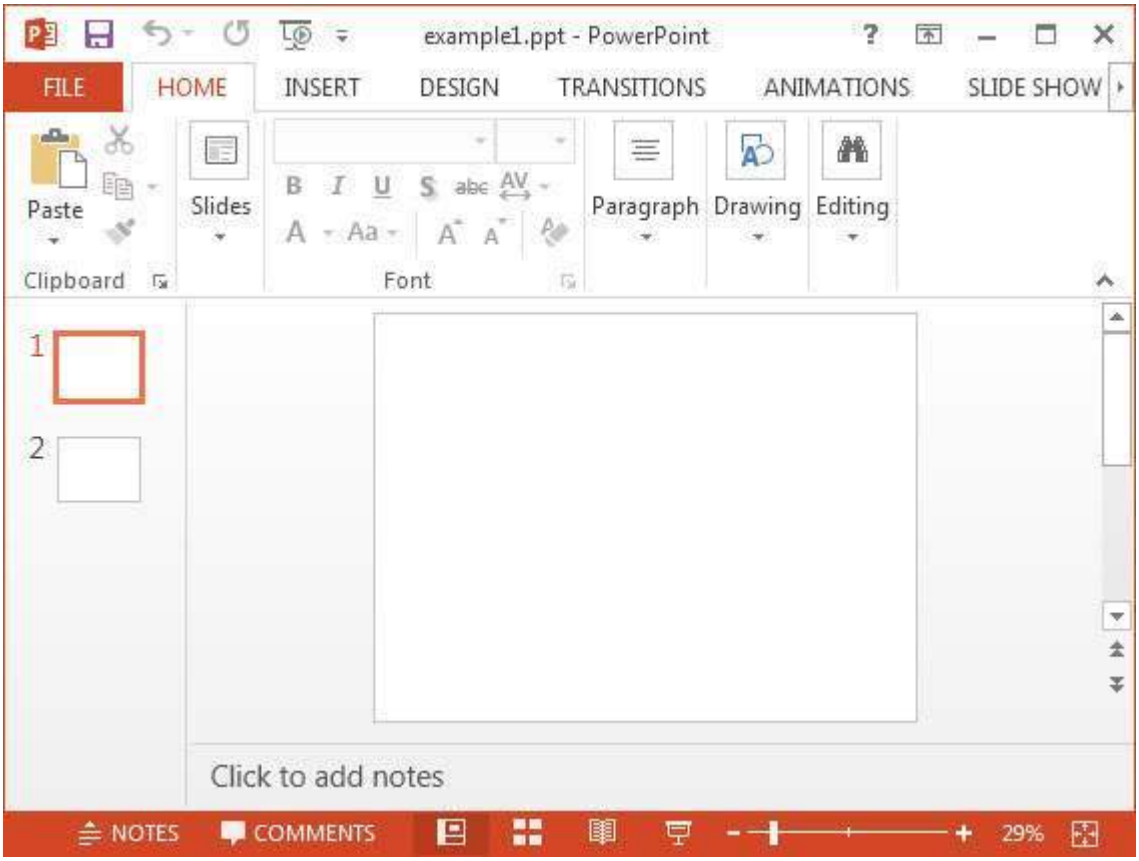
Save the above Java code as **EditPresentation.java**, and then compile and execute it from the command prompt as follows:

```
$javac EditPresentation.java
$java EditPresentation
```

It will compile and execute to generate the following output:

```
slides successfully added
```

The output PPT document with newly added slides looks as follows:



After adding slides to a PPT, you can add, perform, read, and write operations on the slides.