MODUL 6
FORM DAN DIALOG

41. MEMBUAT FORM TURUNAN DARI KELAS QDIALOG

Dalam PyQt, form dibeakan menjadi dialog dan main window. Dialog diturunkan dari kelas QDialog dan main wondow diturunkan dari kelas QMainWindow. Form main window secara default memiliki menubar, toolbar, statusbar, dan widget. Forn utama dapat dibuat dengan kelas QMainWindow, QWidget maupun QDialog.

A. Metode accept() dan reject()

Metode accet() dan reject() adalah dua metode penting yang sering digunakan. Metode accet() akan menutup form dengan mengembalikan nilai dari QDialog.Accepted. Sedangkan reject() akan menutup form dengan mengembalikan nilai QDialog.Rejected. Kedua metode ini dapat digunakan untuk menendai respon yang dilakukan oleh user pada form dialog yang ditamplkan. Misal adanya form dengan tombol ok dan cancel, maka anda dapat mendeteksi tombol yang ditekan oleh user dengan menggunakan metode accept() dan reject(). Tuliskan kode program dibawah ini untuk implementasi metode accept() dan reject().

```
#######################################################
# Nama file: DemoQDialog.py
#######################################################
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class DialogForm(QDialog):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(300, 100)
        self.move(320, 280)
        self.setWindowTitle('Dialog')
```

```
        self.label = QLabel('Form Kedua (Dialog)')
        self.okButton = QPushButton('OK')
        self.cancelButton = QPushButton('Batal')
        hbox = QHBoxLayout()
        hbox.addStretch()
        hbox.addWidget(self.okButton)
        hbox.addWidget(self.cancelButton)
        layout = QVBoxLayout()
        layout.addWidget(self.label)
        layout.addLayout(hbox)
        self.setLayout(layout)
        self.okButton.clicked.connect(self.accept)
        self.cancelButton.clicked.connect(self.reject)
class MainForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(350, 100)
        self.move(300, 300)
        self.setWindowTitle('Demo QDialog.accept() dan
QDialog.reject()')
        self.label = QLabel('Form Utama')
        self.showDialogButton = QPushButton('Tampilkan
Dialog')
        layout = QVBoxLayout()
        layout.addWidget(self.label)
        layout.addWidget(self.showDialogButton)
        self.setLayout(layout)

self.showDialogButton.clicked.connect(self.showDialogB
uttonClick)
```

```
    def showDialogButtonClick(self):
        form = DialogForm()
        if form.exec_() == QDialog.Accepted:
            QMessageBox.information(self,
              'Informasi', 'Anda memilih tombol OK')
        else:   # QDialog.Rejected
            QMessageBox.information(self,
              'Informasi', 'Anda memilih tombol Batal')
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

Tuliakan kode program dibawah ini pada computer anda untuk menerapkan konset accet() dan reject() untuk memasukkan nilai pada form.

```
#############################################################
# Nama file: DemoQDialog1.py
#############################################################
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class AddForm(QDialog):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(350, 130)
        self.move(320, 280)
        self.setWindowTitle('Tambah Data')
```

```
        self.label1 = QLabel('Bahasa Pemrograman')
        self.languageEdit = QLineEdit()
        self.label2 = QLabel('Nama Pencipta')
        self.nameEdit = QLineEdit()
        grid = QGridLayout()
        grid.addWidget(self.label1, 0, 0)
        grid.addWidget(self.languageEdit, 0, 1)
        grid.addWidget(self.label2, 1, 0)
        grid.addWidget(self.nameEdit, 1, 1)
        self.okButton = QPushButton('OK')
        self.cancelButton = QPushButton('Batal')
        hbox = QHBoxLayout()
        hbox.addStretch()
        hbox.addWidget(self.okButton)
        hbox.addWidget(self.cancelButton)
        layout = QVBoxLayout()
        layout.addLayout(grid)
        layout.addLayout(hbox)
        self.setLayout(layout)
        self.okButton.clicked.connect(self.accept)
        self.cancelButton.clicked.connect(self.reject)
class MainForm(QWidget):
    lastRecordNumber = -1
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(450, 300)
        self.move(300, 300)
        self.setWindowTitle('Demo QDialog.accept() dan
QDialog.reject()')
```

```
        self.tableWidget = QTableWidget()
        self.tableWidget.setRowCount(0)
        self.setColumnAndHeaders()
        self.addButton = QPushButton('Tambah')
        self.deleteButton = QPushButton('Hapus')
        self.exitButton = QPushButton('Keluar')
        vbox = QVBoxLayout()
        vbox.addWidget(self.addButton)
        vbox.addWidget(self.deleteButton)
        vbox.addStretch()
        vbox.addWidget(self.exitButton)
        layout = QHBoxLayout()
        layout.addWidget(self.tableWidget)
        layout.addLayout(vbox)
        self.setLayout(layout)
 self.addButton.clicked.connect(self.addButtonClick)

self.deleteButton.clicked.connect(self.deleteButtonCli
ck)
 self.exitButton.clicked.connect(self.exitButtonClick)
   def setColumnAndHeaders(self):
      self.tableWidget.setColumnCount(2)
      columnHeaders = ['Bahasa Pemrograman', 'Nama
Pencipta']

self.tableWidget.setHorizontalHeaderLabels(columnHeade
rs)
   def addRow(self, row, itemLabels=[]):
      for i in range(2):
         item = QTableWidgetItem()
         item.setText(itemLabels[i])
         self.tableWidget.setItem(row, i, item)
```

```
    def addButtonClick(self):
        if MainForm.lastRecordNumber ==
self.tableWidget.rowCount()-1:
            self.tableWidget.setRowCount(
              self.tableWidget.rowCount()+1)
        form = AddForm()
        if form.exec_() == QDialog.Accepted:
            MainForm.lastRecordNumber += 1
            language = form.languageEdit.text()
            name = form.nameEdit.text()
            data = [language, name]
            self.addRow(MainForm.lastRecordNumber, data)
    def deleteButtonClick(self):
        tableData = []
        for i in range(0, self.tableWidget.rowCount()):
            language = self.tableWidget.item(i, 0).text()
            name = self.tableWidget.item(i, 1).text()
            tableData.append([language, name])
        row = self.tableWidget.currentRow()
        del tableData[row]
        MainForm.lastRecordNumber -= 1
        self.tableWidget.clear()
        self.setColumnAndHeaders()
        self.tableWidget.setRowCount(len(tableData))
        for i in range(0, len(tableData)):
            data = tableData[i]
            self.addRow(i, data)
    def exitButtonClick(self):
        self.close()
if __name__ == '__main__':
    a = QApplication(sys.argv)
```

```
    form = MainForm()
    form.show()
    a.exec_()
```

B. Modal dan non-modal

Dialog yang ditampilkan dengan mengggunakan metode exec_() akan selalu bersifat model.Selain menggunakan metopde tersebut anda juga dapat menggunakan metode show(). Jika anda menggunakan show(), anda dapat menentukan dialog yang akan ditampilkan bersifat modal atupun non-modal. Jika bersifat model amak form utama tidaka akn dapat diakses sebelum form keduanya ditutup. Tuliskan kode program di bawah ini untuk mengimplementasikan kelas modal dan non-modal.

```
#############################################################
# Nama file: DemoQDialog2.py
#############################################################
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class DialogForm(QDialog):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(300, 100)
        self.move(320, 280)
        self.setWindowTitle('Dialog')
        self.label = QLabel('')
        self.closeButton = QPushButton('Tutup')
        hbox = QHBoxLayout()
        hbox.addStretch()
        hbox.addWidget(self.closeButton)
        layout = QVBoxLayout()
```

```
        layout.addWidget(self.label)
        layout.addLayout(hbox)
        self.setLayout(layout)
        self.closeButton.clicked.connect(self.close)
class MainForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(350, 100)
        self.move(300, 300)
        self.setWindowTitle('Demo QDialog.setModal()')
        self.label = QLabel('Tuliskan teks pada kotak di
bawah ' +
          'ketika dialog ditampilkan.')
        self.lineEdit = QLineEdit()
        self.showModalDialogButton =
QPushButton('Modal')
        self.showModelessDialogButton =
QPushButton('Non-Modal')
        hbox = QHBoxLayout()
        hbox.addWidget(self.showModalDialogButton)
        hbox.addWidget(self.showModelessDialogButton)
        layout = QVBoxLayout()
        layout.addWidget(self.label)
        layout.addWidget(self.lineEdit)
        layout.addLayout(hbox)
        self.setLayout(layout)

self.showModalDialogButton.clicked.connect(self.showMo
dalDialogButtonClick)
```

```
self.showModelessDialogButton.clicked.connect(self.sho
wModelessDialogButtonClick)
    def showModalDialogButtonClick(self):
        self.form = DialogForm()
        self.form.label.setText('Dialog bersifat modal')
        self.form.setModal(True)
        self.form.show()
    def showModelessDialogButtonClick(self):
        self.form = DialogForm()
        self.form.label.setText('Dialog bersifat non-
modal (modeless)')
        self.form.setModal(False)
        self.form.show()
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

42. MENGGUNAKAN KELAS QMESSAGEBOX

Kelas QMessageBox digunakan untuk menampilkan informasi ke user, baik pesan peringatan, informasi ppproses maupun keterangan lainnya yang relevan. Dalam kelas QMessageBox terdapat lima metode statis yang sering digunkan antara lain.

Metode	Fungsi
about()	Digunakan untuk menyampaikan pesan informasi tentang program
Critical()	Digunakan untuk menampilkan pesan kesalahan ketika user melakukan suatu penyimpangan.
Information()	Digunakan untuk menampilkan pesan informasi khusus

Question()	Digunakan untuk menampilkan pesan konformasi, seperti penghapusan file, dll
Warning()	Digunakan untuk menyampaikan pesan peringatan

Tuliskan kode program dibawah ini untuk implementasi dari kelas QMessageBox

```
#############################################################
# Nama file: DemoQMessageBox.py
#############################################################
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class MainForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(450, 100)
        self.move(300, 300)
        self.setWindowTitle('Demo QMessageBox')
        self.aboutButton = QPushButton('About')
        self.criticalButton = QPushButton('Critical')
        self.informationButton = QPushButton('Information')
        self.questionButton = QPushButton('Question')
        self.warningButton = QPushButton('Warning')
        layout = QHBoxLayout()
        layout.addWidget(self.aboutButton)
        layout.addWidget(self.criticalButton)
        layout.addWidget(self.informationButton)
        layout.addWidget(self.questionButton)
```

```
        layout.addWidget(self.warningButton)
        self.setLayout(layout)
 self.aboutButton.clicked.connect(self.aboutButtonClick)

self.criticalButton.clicked.connect(self.criticalButtonCl
ick)

self.informationButton.clicked.connect(self.informationBu
ttonClick)

self.questionButton.clicked.connect(self.questionButtonCl
ick)

self.warningButton.clicked.connect(self.warningButtonClic
k)
    def aboutButtonClick(self):
        QMessageBox.about(self, 'Tentang Program',
          'Video Recorder\n' +
          'Versi 1.0.0\n' +
          'Hak cipta (C) 2016 PyQt Lover Team')
    def criticalButtonClick(self):
        QMessageBox.critical(self, 'Kesalahan',
          'File settings.conf tidak ditemukan.')
    def informationButtonClick(self):
        QMessageBox.information(self, 'Informasi',
          'Proses upload file ke server telah berhasil
dilakukan.')
    def questionButtonClick(self):
        fileName = 'settings.conf'
        response = QMessageBox.question(self, 'Konfirmasi',
          'Anda yakin akan menghapus file %s?' % fileName)
        if response == QMessageBox.Yes:
           QMessageBox.about(self, 'Respon',
```

```
            'Anda memilih tombol Yes')
        else:
            QMessageBox.about(self, 'Respon',
              'Anda memilih tombol No')
    def warningButtonClick(self):
        QMessageBox.warning(self, 'Peringatan',
          'Operasi ini akan menghapus semua data di dalam
disk Anda!')
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

43. MENGGUNAKAN KELAS QFILEDIALOG

Pada dasrnya kelas QFileDialog menyediakan fungsi secara umum untuk operasi yang berhubungan dengan file, seperti memilih atupun menympan file.

A. Dialog untuk memilih file

Untuk menamplkan dialog memilih file dalam python dapat menggunakan metode statis getOpenFileName() dari kelas QFileDialog. Tuliskan kode program berikt pada computer anda untuk implementasi metode getOpenFileName().

```
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class MainForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
```

```
        self.resize(500, 450)
        self.move(300, 300)
        self.setWindowTitle('Demo
QFileDialog.getOpenFileName()')
        self.textEdit = QTextEdit()
        self.openButton = QPushButton('Buka')
        hbox = QHBoxLayout()
        hbox.addWidget(self.openButton)
        hbox.addStretch()
        self.fileLabel = QLabel('Nama file: ')
        layout = QVBoxLayout()
        layout.addWidget(self.textEdit)
        layout.addLayout(hbox)
        layout.addWidget(self.fileLabel)
        self.setLayout(layout)
 self.openButton.clicked.connect(self.openButtonClick)
   def openButtonClick(self):
        import os
        fileName = QFileDialog.getOpenFileName(self,
          'Pilih file', os.curdir,
          'Python Code (*.py);; Ruby Code (*.rb)',
          '*.py')
        if not fileName[0]: return
        self.fileLabel.setText('Nama file: ' +
fileName[0])
        fileHandle = QFile(fileName[0])
        if not fileHandle.open(QIODevice.ReadOnly):
return
        stream = QTextStream(fileHandle)
        self.textEdit.setPlainText(stream.readAll())
        fileHandle.close()
```

```
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

B. Dialog untuk menyimpan file

Sedangkan untuk menyimpan file, adna dapat menggunakan getSaveFileName(). Tuliskan kode program di bawh ini ntuk implementasi metode tersebut

```
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class MainForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
    def setupUi(self):
        self.resize(500, 450)
        self.move(300, 300)
        self.setWindowTitle('Demo
QFileDialog.getSaveFileName()')
        self.textEdit = QTextEdit()
        self.saveButton = QPushButton('Simpan')
        hbox = QHBoxLayout()
        hbox.addWidget(self.saveButton)
        hbox.addStretch()
        self.fileLabel = QLabel('Nama file: ')
        layout = QVBoxLayout()
        layout.addWidget(self.textEdit)
        layout.addLayout(hbox)
```

```
        layout.addWidget(self.fileLabel)
        self.setLayout(layout)
 self.saveButton.clicked.connect(self.saveButtonClick)
   def saveButtonClick(self):
        import os
        fileName = QFileDialog.getSaveFileName(self,
          'Simpan file', os.curdir,
          'Python Code (*.py);; Ruby Code (*.rb)',
          '*.py')
        if not fileName[0]: return
        self.fileLabel.setText('Nama file: ' +
fileName[0])
        fileHandle = QFile(fileName[0])
        if not fileHandle.open(QIODevice.WriteOnly):
return
        stream = QTextStream(fileHandle)
        stream << self.textEdit.document().toPlainText()
        stream.flush()
        fileHandle.close()
if __name__ == '__main__':
   a = QApplication(sys.argv)
   form = MainForm()
   form.show()
   a.exec_()
```

44. MENGGUNAKAN KELAS QFONTDIALOG

Kelas QFontDialog dalam python digunakan untuk menampilkan dialog font. Diamana anada dapat memilih jenis font, ukuran font, dan style font tertentu. Tuliskan kode program berikut untuk implementasi kelas QFontDialog

```
import sys
```

```
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class MainForm(QWidget):
   def __init__(self):
      super().__init__()
      self.setupUi()
   def setupUi(self):
      self.resize(500, 450)
      self.move(300, 300)
      self.setWindowTitle('Demo QFontDialog')
      self.textEdit = QTextEdit()
      self.fontButton = QPushButton('Font')
      hbox = QHBoxLayout()
      hbox.addWidget(self.fontButton)
      hbox.addStretch()
      layout = QVBoxLayout()
      layout.addWidget(self.textEdit)
      layout.addLayout(hbox)
      self.setLayout(layout)
 self.fontButton.clicked.connect(self.fontButtonClick)
   def fontButtonClick(self):
      fontTuple = QFontDialog.getFont(QFont('Sans Serif',
11), self, 'Pilih font')
      if fontTuple[0]:
         self.textEdit.setCurrentFont(fontTuple[0])
if __name__ == '__main__':
   a = QApplication(sys.argv)
   form = MainForm()
   form.show()
   a.exec_()
```

45. MENGGUNAKAN KELAS QCOLORDIALOG

Kelas QFontDialog dalam python digunakan untuk menampilkan dialog font. Diamana anada dapat memilih jenis font, ukuran font, dan style font tertentu. Tuliskan kode program berikut untuk implementasi kelas QFontDialog.

```
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
class MainForm(QWidget):
   def __init__(self):
      super().__init__()
      self.setupUi()
   def setupUi(self):
      self.resize(500, 450)
      self.move(300, 300)
      self.setWindowTitle('Demo QFontDialog')
      self.textEdit = QTextEdit()
      self.fontButton = QPushButton('Font')
      hbox = QHBoxLayout()
      hbox.addWidget(self.fontButton)
      hbox.addStretch()
      layout = QVBoxLayout()
      layout.addWidget(self.textEdit)
      layout.addLayout(hbox)
      self.setLayout(layout)
 self.fontButton.clicked.connect(self.fontButtonClick)
   def fontButtonClick(self):
      fontTuple = QFontDialog.getFont(QFont('Sans Serif',
11), self, 'Pilih font')
```

```
        if fontTuple[0]:
            self.textEdit.setCurrentFont(fontTuple[0])
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

46. MEMBUAT FORM TURUNAN DARI KELAS QMAINWINDOWS

Kelas QMain Window pada pyQt berfungsi untuk membuat berbagai macam widget seperti menu, toolbar maupuan statusbar. Pada dasarnya kelas ini berfungsi untuk membuat tampilan GUI sebagaimana tampilan GUI pada umumnya. Sebagai implementasi buatlah program teks editor pada computer anda sesuai dengan kode program dibawah ini, dana Analisa perintah-perintah untuk membuat toolbar, menu, maupun aksi.

```
import sys, os
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
PROGRAM_NAME = 'PyQt Editor'
class MainForm(QMainWindow):
    def __init__(self):
        super().__init__()
        self.currentFileName = ''
        self.setupUi()
    def setupUi(self):
        self.resize(550, 450)
        self.move(300, 300)
        self.setWindowTitle(PROGRAM_NAME + ' - Untitled')
        # inisialisasi teks pada statusbar
```

```
        self.statusBar().showMessage('Ketikkan teks yang
Anda inginkan')

        # mendapatkan objek menubar

        menubar = self.menuBar()

        # membuat menu File dan menempatkannya ke dalam
menubar

        fileMenu = menubar.addMenu('&File')

        # membuat aksi untuk menu File

        fileNewAction = QAction(QIcon('icons/New.png'),
'&New', self)

        fileNewAction.setShortcut('Ctrl+N')

        fileNewAction.setStatusTip('Buat teks baru')


fileNewAction.triggered.connect(self.fileNewActionTrigger
ed)

        fileMenu.addAction(fileNewAction)

        fileOpenAction = QAction(QIcon('icons/Open.png'),
'&Open...', self)

        fileOpenAction.setShortcut('Ctrl+O')

        fileOpenAction.setStatusTip('Buka file')


fileOpenAction.triggered.connect(self.fileOpenActionTrigg
ered)

        fileMenu.addAction(fileOpenAction)

        fileMenu.addSeparator()

        fileSaveAction = QAction(QIcon('icons/Save.png'),
'&Save', self)

        fileSaveAction.setShortcut('Ctrl+S')

        fileSaveAction.setStatusTip('Simpan teks ke file')


fileSaveAction.triggered.connect(self.fileSaveActionTrigg
ered)

        fileMenu.addAction(fileSaveAction)
```

```
        fileSaveAsAction = QAction(QIcon(None), 'Save
&As...', self)
        fileSaveAsAction.setStatusTip('Simpan teks ke file
lain')

fileSaveAsAction.triggered.connect(self.fileSaveAsActionT
riggered)
        fileMenu.addAction(fileSaveAsAction)
        fileMenu.addSeparator()
        fileExitAction = QAction(QIcon(None), 'Exit', self)
        fileExitAction.setShortcut('Ctrl+Q')
        fileExitAction.setStatusTip('Buat dokumen baru')

fileExitAction.triggered.connect(self.fileExitActionTrigg
ered)
        fileMenu.addAction(fileExitAction)
        # membuat menu Edit dan menempatkannya ke dalam
menubar
        editMenu = menubar.addMenu('&Edit')
        # membuat aksi untuk menu Edit
        editCutAction = QAction(QIcon('icons/Cut.png'),
'C&ut', self)
        editCutAction.setShortcut('Ctrl+X')
        editCutAction.setStatusTip('Potong teks')

editCutAction.triggered.connect(self.editCutActionTrigger
ed)
        editMenu.addAction(editCutAction)
        editCopyAction = QAction(QIcon('icons/Copy.png'),
'&Copy', self)
        editCopyAction.setShortcut('Ctrl+C')
        editCopyAction.setStatusTip('Salin teks')
```

```
editCopyAction.triggered.connect(self.editCopyActionTrigg
ered)
      editMenu.addAction(editCopyAction)
      editMenu.addSeparator()
      editPasteAction = QAction(QIcon('icons/Paste.png'),
'&Paste', self)
      editPasteAction.setShortcut('Ctrl+V')
      editPasteAction.setStatusTip('Tempel teks (yang
telah dipotong/disalin)')

editPasteAction.triggered.connect(self.editPasteActionTri
ggered)
      editMenu.addAction(editPasteAction)
      # membuat menu Format dan menempatkannya ke dalam
menubar
      formatMenu = menubar.addMenu('F&ormat')
      # membuat aksi untuk menu Format
      formatFontAction = QAction(QIcon(None), 'F&ont...',
self)
      formatFontAction.setStatusTip(
        'Menentukan jenis dan ukuran huruf pada teks yang
disorot')

formatFontAction.triggered.connect(self.formatFontActionT
riggered)
      formatMenu.addAction(formatFontAction)
      # membuat toolbar
      toolbar = self.addToolBar('')
      toolbar.addAction(fileNewAction)
      toolbar.addAction(fileOpenAction)
      toolbar.addAction(fileSaveAction)
      toolbar.addSeparator()
      toolbar.addAction(editCutAction)
```

```
        toolbar.addAction(editCopyAction)
        toolbar.addAction(editPasteAction)
        # membuat objek QTextEdit dan menempatkannya ke
dalam pusat widget
        self.textEdit = QTextEdit()
        self.setCentralWidget(self.textEdit)


    def confirmation(self):
        if self.textEdit.document().isModified():
            response = QMessageBox.question(self,
'Konfirmasi',
                        'Teks telah dimodifikasi. Simpan?')
            if response == QMessageBox.Yes:
                self.fileSaveActionTriggered()
    def fileNewActionTriggered(self):
        self.confirmation()
        self.textEdit.document().clear()
        self.currentFileName = ''
        self.setWindowTitle(PROGRAM_NAME + ' - Untitled')
    def fileOpenActionTriggered(self):
        self.confirmation()
        fileName = QFileDialog.getOpenFileName(self,
          'Pilih file', os.curdir,
          'File Teks (*.txt)',
          '*.txt')
        if not fileName[0]: return
        self.currentFileName = fileName[0]
        self.setWindowTitle(PROGRAM_NAME + ' - ' +
self.currentFileName)
        fileHandle = QFile(fileName[0])
        if not fileHandle.open(QIODevice.ReadOnly): return
```

```
        stream = QTextStream(fileHandle)
        self.textEdit.setPlainText(stream.readAll())
        fileHandle.close()
    def writeToFile(self):
        fileHandle = QFile(self.currentFileName)
        if not fileHandle.open(QIODevice.WriteOnly): return
        stream = QTextStream(fileHandle)
        stream << self.textEdit.document().toPlainText()
        stream.flush()
        fileHandle.close()
        self.textEdit.document().setModified(False)
    def fileSaveActionTriggered(self):
        if self.currentFileName == '':
            # mengeksekusi aksi Save As
            self.fileSaveAsActionTriggered()
        else:
            self.writeToFile()
    def fileSaveAsActionTriggered(self):
        fileName = QFileDialog.getSaveFileName(self,
          'Simpan file', os.curdir,
          'File Teks (*.txt)',
          '*.txt')
        if not fileName[0]: return
        self.currentFileName = fileName[0]
        self.setWindowTitle(PROGRAM_NAME + ' - ' +
self.currentFileName)
        self.writeToFile()
    def fileExitActionTriggered(self):
        sys.exit(0)
    def editCutActionTriggered(self):
        self.textEdit.cut()
```

```
    def editCopyActionTriggered(self):
        self.textEdit.copy()
    def editPasteActionTriggered(self):
        self.textEdit.paste()
    def formatFontActionTriggered(self):
        fontTuple = QFontDialog.getFont(
          QFont('Sans Serif', 11),
          self, 'Pilih font')
        if fontTuple[0]:
            self.textEdit.setCurrentFont(fontTuple[0])
if __name__ == '__main__':
    a = QApplication(sys.argv)
    form = MainForm()
    form.show()
    a.exec_()
```

.