Uygulama Yöneticisi Projesi Raporu

February 2, 2021

Yazar: Fatih Selim YAKAR

1 Kullanılan Programlar ve Teknolojiler

- Java 11.0.9
- JavaFX 15.0.1
- JavaFXSceneBuilder2.0
- Eclipse 4.11
- Javadoc
- Latex
- UML Generator
- Linux(Ubuntu 20.04.1 LTS)

2 Problem/Proje Tanımı

Bu projede Java programlama dili ve JavaFX kullanıcı arayüzü geliştirme iskeleti kullanılarak uygulama yöneticisi adında bir uygulama geliştirilmesi beklenilmektedir.Geliştirilecek uygulamanın, kullanıcı arayüzü üzerinden aldığı girdiler sayesinde kullanıcının istediği uygulamaları ilgili parametrelere göre çalıştırması ve uygulama arayüzündeki tabloya process'in bilgilerinin eklenmesi beklenilmektedir. Tabloda çalıştılan process'in ID'si, process ismi, process parametreleri, başlama zamanından bu yana geçen zaman bilgisi gösterilmelidir. Ayrıca başlatılan processleri sonlandırmak için kullanıcı arayüzü sağlanmalı ve sonlanması halinde tablodan silinmesi beklenilmektedir.

3 Geliştirme Ortamı

Geliştirme ortamı olarak Ubuntu 20.04.1 LTS — 2.7GHz - 128GB 2.7GHz çift çekirdek - Intel Core i5 işlemci özelliği barındıran bilgisayarda Eclipse 4.11 IDE'si üzerinde Java,
JavaFX,JavaFXSceneBuilder,Javadoc kullanılmıştır.

4 Gereksinimler

4.1 Fonksiyonel Gereksinimler

- Gerekli parametreler girilip "Start process" tuşuna basıldığında girilmiş parametreler temizlenmeli, bu parametrelere göre yeni bir process satırı oluşturulmalı ve tabloya PID, varsa argümanlar, process ismi, ne kadar süredir çalıştığı gibi bilgiler eklenmelidir. Ayrıca süre anlık olarak güncellenmelidir.
- Herhangi bir process'in kill butonuna basıldığında tablodaki process ile ilgili olan bilgiler silinmeli ve altındaki diğer process tarafından yeri alınmalıdır. Tablodan silinen process ayrıca gerçekte de kapanmalıdır.

4.2 Fonksiyonel Olmayan Gereksinimler

- Kullanıcı arayüzü belirtilen şekilde ve düzende olmalıdır.
- Programın açılmasında ve kapanmasında gecikme yaşatmamalıdır.
- Geliştirme ortamı olan Ubuntu 20.04.1 LTS — 2.7GHz - 128GB 2.7GHz çift çekirdek - Intel Core i5 işlemci özelliği barındıran bilgisayarda çalışmalıdır.
- Hem Eclipse üzerinde hem terminal üzerinde çalışabilir olmalıdır.

5 Tasarım

- Öncelikle belirtilen uygulamanın tasarımının yapılması ve bir .fxml dosyası oluşması amacıyla eclipse SceneBuilder kullanıldı.Burda uygulamanın arayüzü oluşturuldu fakat tablonun satır ve sutünları java dosyasındaki TableProcess model sınıfını kullanmak amacı ile doldurulmadı.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xm
   <children>
      <Button id="start_process_button" layoutX="248.0" layoutY="157.0" mnemonicParsing="false" text="Start Process" />
      <Label layoutX="133.0" layoutY="61.0" text="Directory:" />
      <Label layoutX="133.0" layoutY="94.0" text="Executable:" />
      <Label layoutX="133.0" layoutY="125.0" text="Arguments:" />
      <TextField id="directory_box" layoutX="214.0" layoutY="56.0" prefHeight="25.0" prefWidth="217.0" />
      <TextField id="executable_box" layoutX="215.0" layoutY="89.0" prefHeight="25.0" prefWidth="217.0" />
      <TextField id="argument_box" layoutX="215.0" layoutY="120.0" prefHeight="25.0" prefWidth="217.0" />
      <TableView id="table" layoutX="27.0" layoutY="186.0" prefHeight="200.0" prefWidth="553.0">
        <columns>
        </columns>
      </TableView>
   </children>
</Pane>
```

- Komut çalıştırmak yani process oluşturmak için Runtime.getRuntime().exec() metodu kullanarak runProgram() adlı private metod yazıldı. Böylece terminal üzerinden komut çalıştırmaya benzer bir çalıştırma sistemi yapıldı. Bunun yanında process'i kapatmak içinde yine runProgram() metodu "kill -15 [PID]" komutu ile kullanıldı.

```
/**
 * This method creates a new process by running the commandline command using java's Runtime.getRuntime (). Exec method.
 * @param directory This is the executable's directory.
 * @param executable This is executable name.
 * @return long Returns the created process's PID.
 */
private long runProgram(String directory,String executable,String arguments) throws IOException {
    Process process = Runtime.getRuntime().exec("./"+executable+" "+arguments ,null, new File(directory));
    return process.pid();
}
```

- Programın çalışma zamanını ölçme ise JavaFX'den Timeline sınıfı ile oluşturulan bir timer ile sürekli saniyede bir olcak şekilde ilgili satırı güncelleyerek saniye alanını değiştirme yoluyla sağlandı.

```
/* Adds or updates timerRows with row integer value */
int row=rowSize;
if(timerRows.size()==row)
    timerRows.add(row);
else
    timerRows.set(row, row);

/* Creates the timeline timer in running duration cell */
process.setTimer(new Timeline(new KeyFrame(Duration.seconds(1), e -> {
    TableProcess temp_process=process;
    temp_process.setRunningDuration(integerToClock(clockToInteger(temp_process.getRunningDuration())+1));
    table.getItems().set(timerRows.get(row), temp_process);
    //System.out.println("timer");

})));

/* starts the timer and increases the rowSize*/
process.startTimer();
++rowSize;
```

- SceneBuilder ile tasarım yapıldıktan sonra javaFX'in Application sınıfından türeyen bir sınıf yapıldı. Sınıfın private değişkenleri olarak ise table,rowSize ve timerRows adı altında üç değişken kullanıldı.

Table değişkeni tarafından uygulamanın içindeki dinamik tablo tutuldu,rowSize tarafından bu tablonun kaç satırdan oluştuğu, timerRows tarafından ise timerlar ve process satırları arasındaki ilişki yönetildi.

```
import javafx.beans.property.SimpleStringProperty;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.util.Callback;
import javafx.util.Duration;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableCell;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;


public class Main extends Application {

    /**
     * Data structure holding the table in the application.
     */
    private TableView<TableProcess> table = new TableView<TableProcess>();
    /**
     * Holds the current row size in table.
     */
    private int rowSize=0;
    /**
     * It is an arraylist where the values of timers are kept.
     */
    private ArrayList<Integer>timerRows=new ArrayList<Integer>();

    /**
     * This method creates a new process by running the commandline command using java's Runtime.getRuntime (). Exec method.
     * @param directory This is the executable's directory.
     * @param executable This is executable name.
```

- Tablonun her process satırını ifade etmek amacıyla ana sınıfın yanında TableProcess adı verilen bir statik alt sınıf oluşturuldu. Bu sınıf her satırın tutacağı PID,process ismi,arguman ve zamanları string tipinde tutuldu.Metod olarak ise sadece getter ve setterları içeriyordu.
- Genel olarak tüm işlemler oluşturulan ana sınıf içindeki start metodu içinde yapıldı. Öncelikle .fxml tipinde dosyayı yükleyip ona göre bir scene oluşturuldu.
- .fxlm tarafından oluşturulan start process butonu lookup metodu ile alınıp buton üzerinde basınca işlem yapmasını sağlayacak şekilde setOnAction metodu ile yeni işlevler eklendi böylece butona basınca hem parameteleri alıp tabloya process ekleyebildi hemde yazılmış runProgram private metodu ile process'i çalıştırabildi. Ayrıca saniyenin akmasını sağlayan kodda bu kısımda yazıldı.

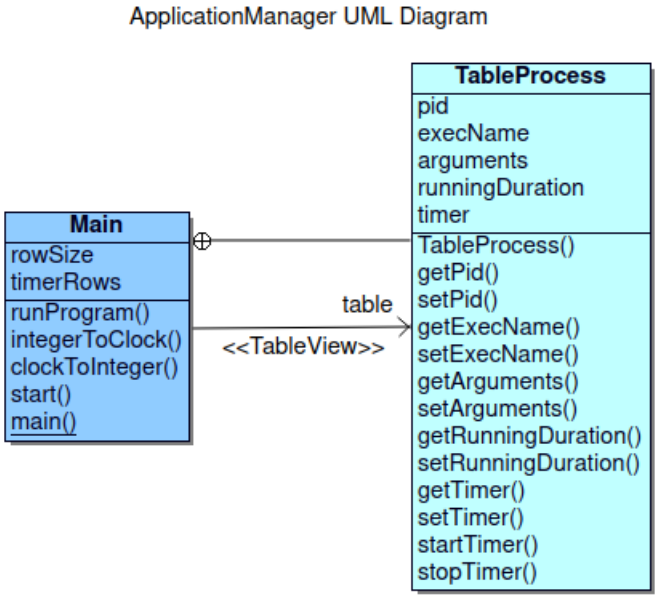
```
/* When the star process button is pressed, it adds new process blocks to the table and creates that process with the runProgram method.*/
but.setOnAction(event -> {

    /* Gets the current textfield parameters in the boxes */
    TextField directory=(TextField)scene.lookup("#directory_box");
    TextField executable=(TextField)scene.lookup("#executable_box");
    TextField arguments=(TextField)scene.lookup("#argument_box");

    System.out.println(directory.getText()+" "+executable.getText()+" "+arguments.getText());

    /* Runs the process and gets the PID of process */
    long pid;
    try {
        pid=runProgram(directory.getText(),executable.getText(),arguments.getText());
    } catch (IOException e) {
        e.printStackTrace();
        pid=-1;
    }

    /* Creates a TableProcess by the passed parameters */
    TableProcess process = new TableProcess(String.valueOf(pid), executable.getText(), arguments.getText(),"0:0:0");

    /* Adds the process in the table */
    table.getItems().add(process);
    //TableColumn<TableProcess, ?> process_col = table.getColumns().get(3);
    //System.out.println(process_col.getCellData(rowSize));

    /* Adds or updates timerRows with row integer value */
    int row=rowSize;
    if(timerRows.size()==row)
        timerRows.add(row);
    else
        timerRows.set(row, row);

    /* Creates the timeline timer in running duration cell */
    process.setTimer(new Timeline(new KeyFrame(Duration.seconds(1), e -> {
        TableProcess temp_process=process;
        temp_process.setRunningDuration(integerToClock(clockToInteger(temp_process.getRunningDuration())+1));
        table.getItems().set(timerRows.get(row), temp_process);
        //System.out.println("timer");

    })));

    /* starts the timer and increases the rowSize*/
    process.startTimer();
    ++rowSize;

    /* Removes the parameter boxes */
    directory.clear();
    executable.clear();
    arguments.clear();

});
```

- Kill butonunun yapılabilmesi için ise öncelikle geçici bir buton oluşturup bu butonun işlevi üzerinde değişiklik yapılarak setOnAction metodu içine tablodan process'i silme ve runProgram ile halihazırda çalışan process sonlandırıldı.

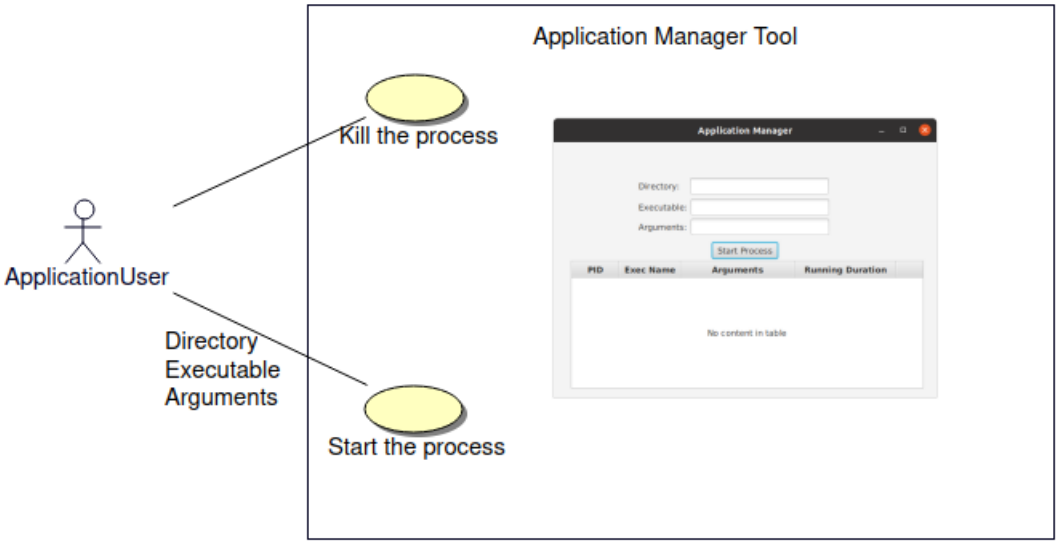
```
btn.setOnAction(event -> {
    /* gets the current process */
    TableProcess process = getTableView().getItems().get(getIndex());

    System.out.println("Killed "+process.getPid() + " " + process.getExecName());

    /* Executes the kill command(SIGTERM signal) */
    try {
        runProgram("/usr/bin","./kill","-15 "+process.getPid());
    } catch (IOException e) {
        e.printStackTrace();
    }

    /* removes the current process in the table and updates timers */
    table.getItems().remove(process);
    for(int i=getIndex();i<timerRows.size();++i) {
        if(timerRows.get(i)>0)
            timerRows.set(i, timerRows.get(i)-1);
    }
    /* Stops the own timer and decreases the rowSize */
    process.stopTimer();
    --rowSize;
});
setGraphic(btn);
setText(null);
```

6 Diyagramlar

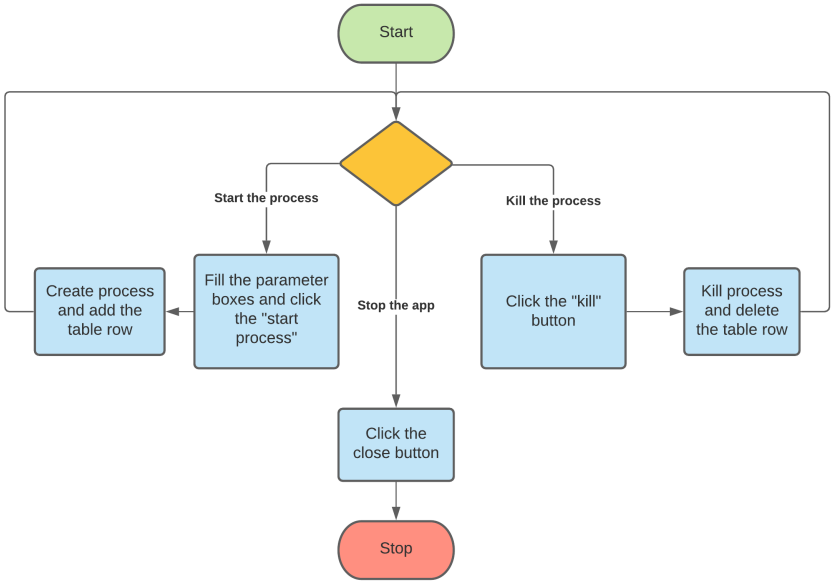
6.1 UML Diyagramı



6.2 Kullanım Durumları Diyagramı(Use Case Diagram)



6.3 Akış Diyagramı(Flow Diagram)



7 Çalıştırma Talimatları

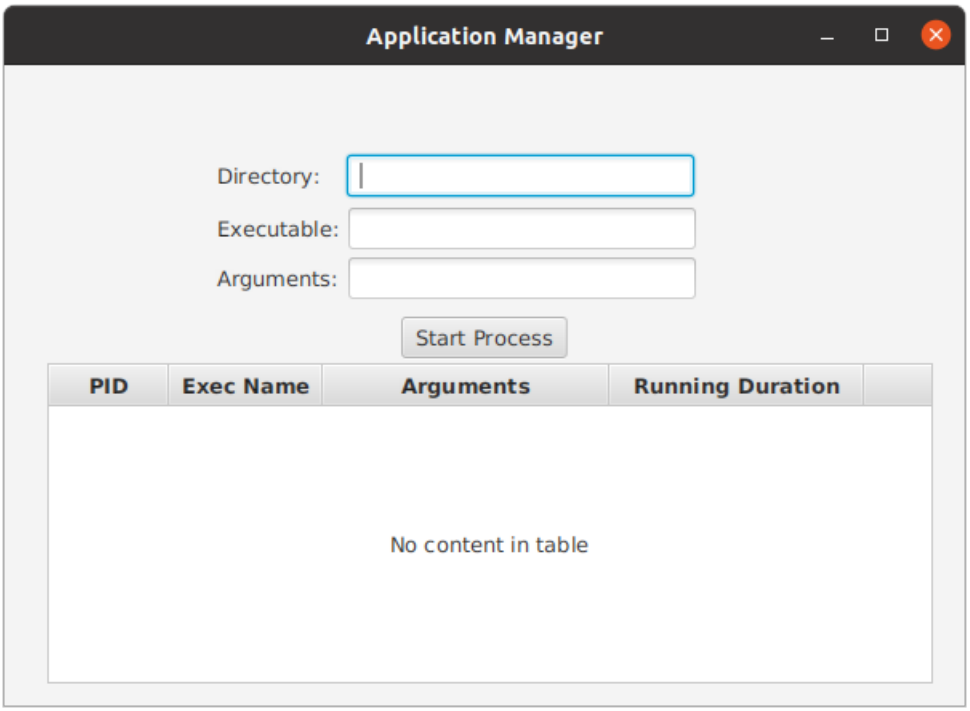
7.1 Eclipse 4.11 ile

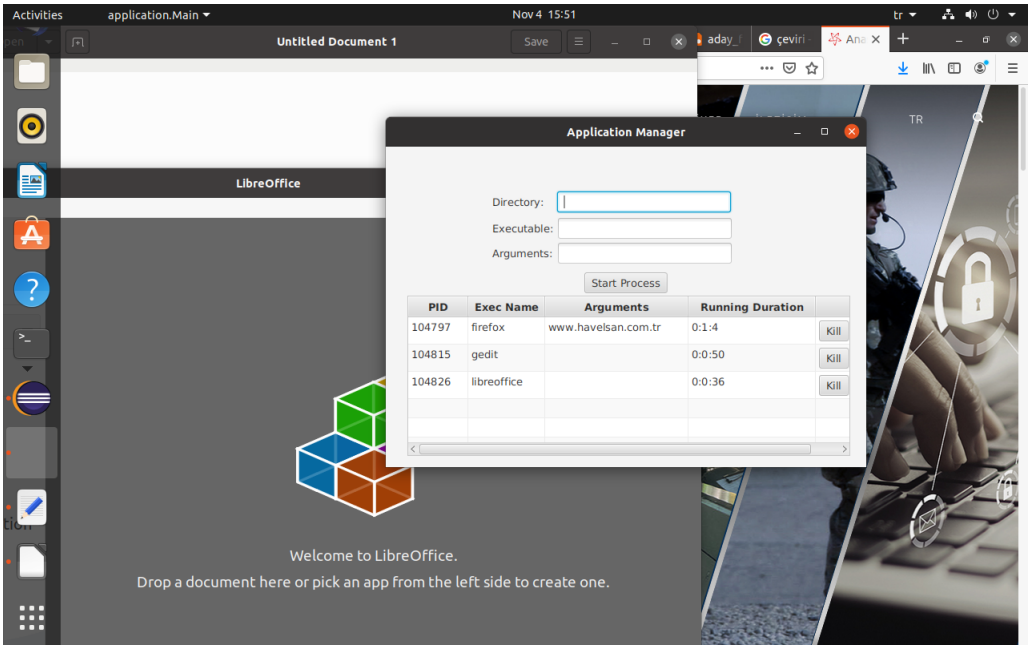
- Gereklilikler, yani JavaFX 15.0.1 ve Java 11.0.9 Eclipse 4.11 IDE'sine yüklü ve entegre ise, sadece sağlanan eclipse projesini açıp ardından "run" tuşuna basmak yeterlidir.
- Eğer gereklilikler yüklü veya entegre değilse raporla beraber sağlanan kurulum dokümanı yardımı ile kurunuz.

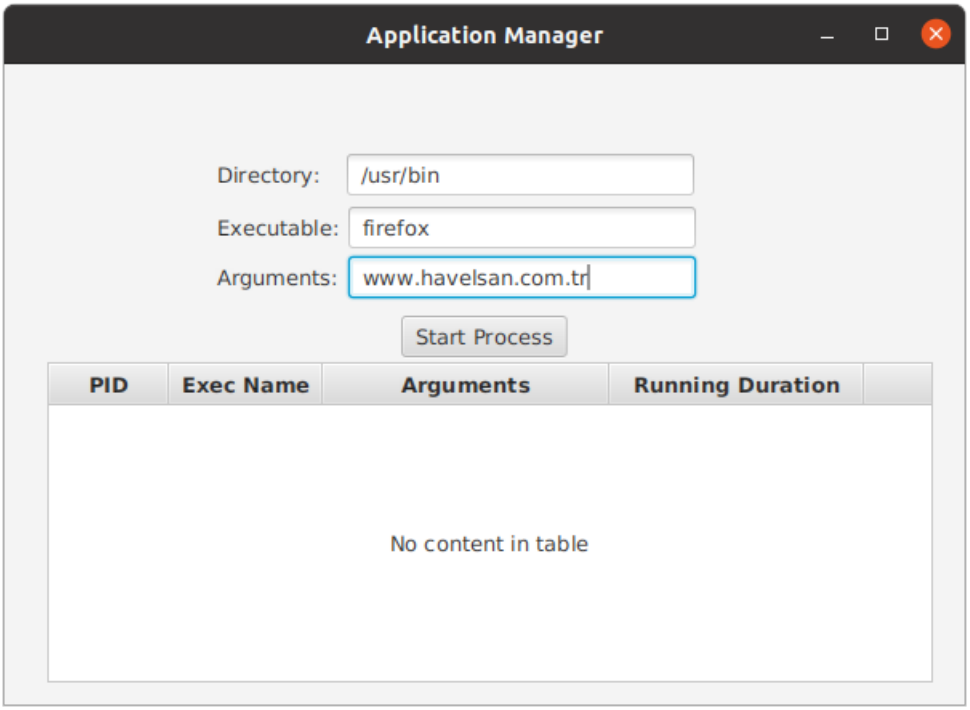
7.2 Linux(Ubuntu) Terminal ile

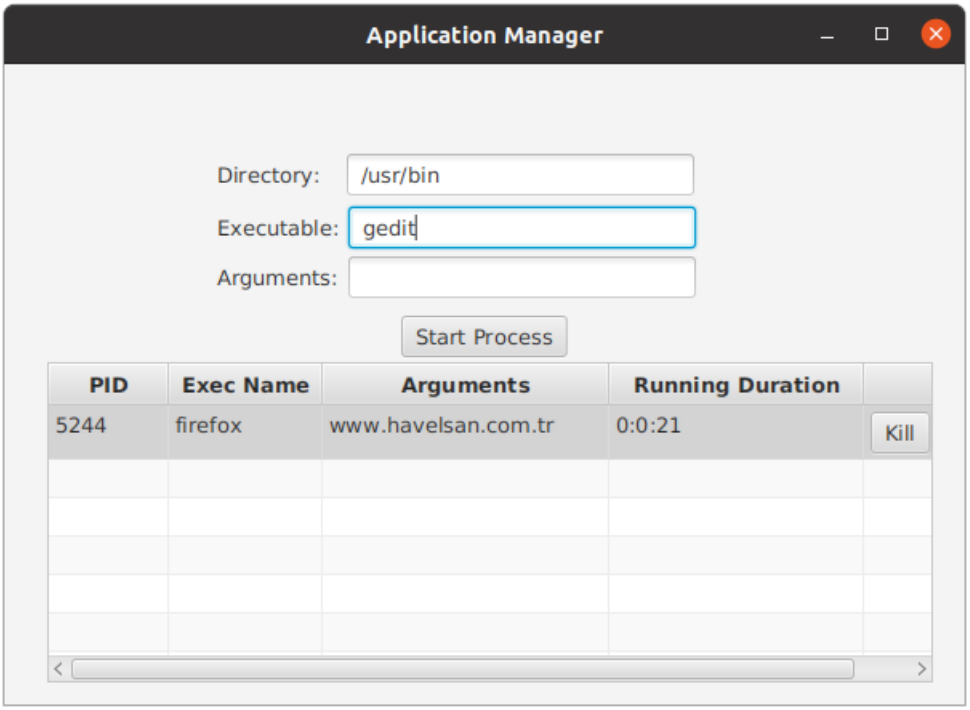
- Java 11.0.9 kurulu ise sağlanan kaynak dosyalası içinde şu komutları çalıştırmak yeterlidir:
  - **"javac -cp lib/*:. Main.java"**
  - **"java –module-path lib –add-modules javafx.controls,javafx.fxml -cp .:* Main"**
- Eğer gereklilikler yükle veya entegre değilse raporla beraber sağlanan kurulum dokümanı yardımı ile kurunuz.

8 Çalışma Ekran Görüntüleri






Application Manager
Directory:
/usr/bin
Executable:
firefox
Arguments:
www.havelsan.com.tr
Start Process
PID
Exec Name
Arguments
Running Duration
No content in table


Application Manager
Directory:
/usr/bin
Executable:
gedit
Arguments:
Start Process
PID
Exec Name
Arguments
Running Duration
5244
firefox
www.havelsan.com.tr
0:0:21
Kill

9 Çalışma Videosu

- `https://youtu.be/V_TZMLtRd7w`
- Alternatif bağlantı:

  `https://drive.google.com/file/d/1WO9pf6fLE2JWpoFKKhetBXrfNP0u2P9d/view?usp=sharing`

10 Kaynaklar

- https://openjfx.io/openjfx-docs/
- https://lucid.app/lucidchart
- https://linuxize.com/post/how-to-install-the-latest-eclipse-ide-on-ubuntu-18-04/
- https://www.baeldung.com/java-home-on-windows-7-8-10-mac-os-x-linux
- https://gluonhq.com/products/javafx/
- https://www.how2shout.com/how-to/install-eclipse-linux-ubuntu-using-terminal.html
- https://www.oracle.com/java/technologies/javafx-scene-builder-source-code.html
- https://www.overleaf.com