Funkcja znajdująca wszystkie dyski na komputerze:

```
  Disposable drives = getDrives()
          .subscribe(s -> {
             Text text = new Text(s.toString()+"\n");
             text.setFill(Color.BLUE);
             text.fontProperty().setValue(text.getFont().font(20));
             dirPrint.getChildren().add(text);

             JavaFxObservable.eventsOf(text,  MouseEvent.MOUSE_CLICKED)
                   .subscribe(
                         s1 -> {
                            path.setText(s.toString());
                         }
                   );
          });

}
```

```
public Observable<File> getDrives() {
   return Observable.fromArray(File.listRoots());
}
```

Funkcja skanująca pliki i foldery w danej ścieżce

```
Observable<String> scanPathObs(String path) throws IOException {
   return Observable.create(emitter -> {
      try {
         Files.list(Paths.get(path))
               .parallel()
               .filter(s -> s.toFile().isDirectory() || s.toFile().isFile())
               .map(Path::toString)
               .map(s -> s + "\n")
               .forEach(emitter::onNext);
         emitter.onComplete();
      } catch (IOException e) {
         emitter.onError(e);
      }
   });
}
```

Funkcja inicjalizująca (wywoływana na starcie programu)

```
@FXML
void initialize() {
   path.setText("C:\\");

   JavaFxObservable.valuesOf(path.textProperty())
         .subscribe(
               s -> {
                  try {
                     print.getChildren().clear();

                     scanPathObs(s)
                           .subscribeOn(io.reactivex.schedulers.Schedulers.newThread())
                           .observeOn((JavaFxScheduler.platform()))
                           .subscribe(
                                 s1 -> {
                                    String url = s1;
                                    Text text = new Text(url);
                                    if (!url.contains(".")) {
                                       text.setFill(Color.BLUE);
                                       JavaFxObservable.eventsOf(text, MouseEvent.MOUSE_CLICKED)
                                             .subscribe(
                                                   s2 -> {
                                                      path.setText(url);
                                                   }
                                             );
```

```
                                    }else{
                                      JavaFxObservable.eventsOf(text, MouseEvent.MOUSE_CLICKED)
                                                .subscribe(
                                                      s2 -> {
                                                         try {
                                       Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + url.trim());
                                                   } catch (IOException e) {
                                                      e.printStackTrace();
                                                   }
                                                }
                                         );
}
```

```
                                    text.fontProperty().setValue(text.getFont().font(20));
                                    print.getChildren().add(text);
                                 }
                           );

                     } catch (IOException e) {
                        e.printStackTrace();
                     }
                  }
```

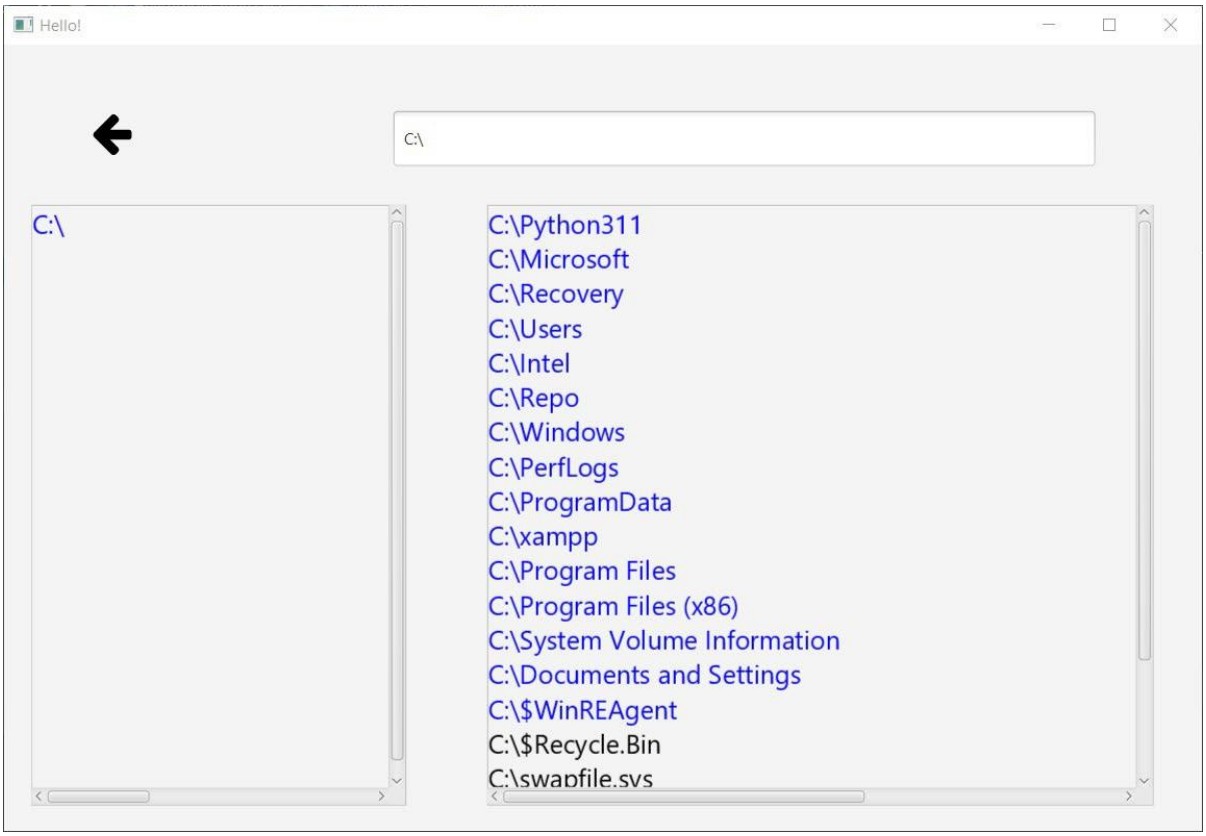
Program przy każdej zmianie ścieżki, na nowo ładuje wszystkie pliki i foldery w aktualnej ścieżce i wyświetla je. W przypadku, kiedy w ścieżce znajduje się folder, zamiast tylko wyświetlenia, zmieniam mu kolor na niebieski by był bardziej „interaktywny” i dodaje do niego Observable, który sprawdza czy został kliknięty. W przypadku gdy został kliknięty, następuje zmiana ścieżki do klikniętego folderu. W przypadku kliknięcia na plik, zostanie on otwarty domyślnie zarejestrowanym programem w Windowsie

W przypadku kliknięcia przez użytkownika ikony cofnięcia, następuje cofnięcie się ścieżki do poprzedniej wersji

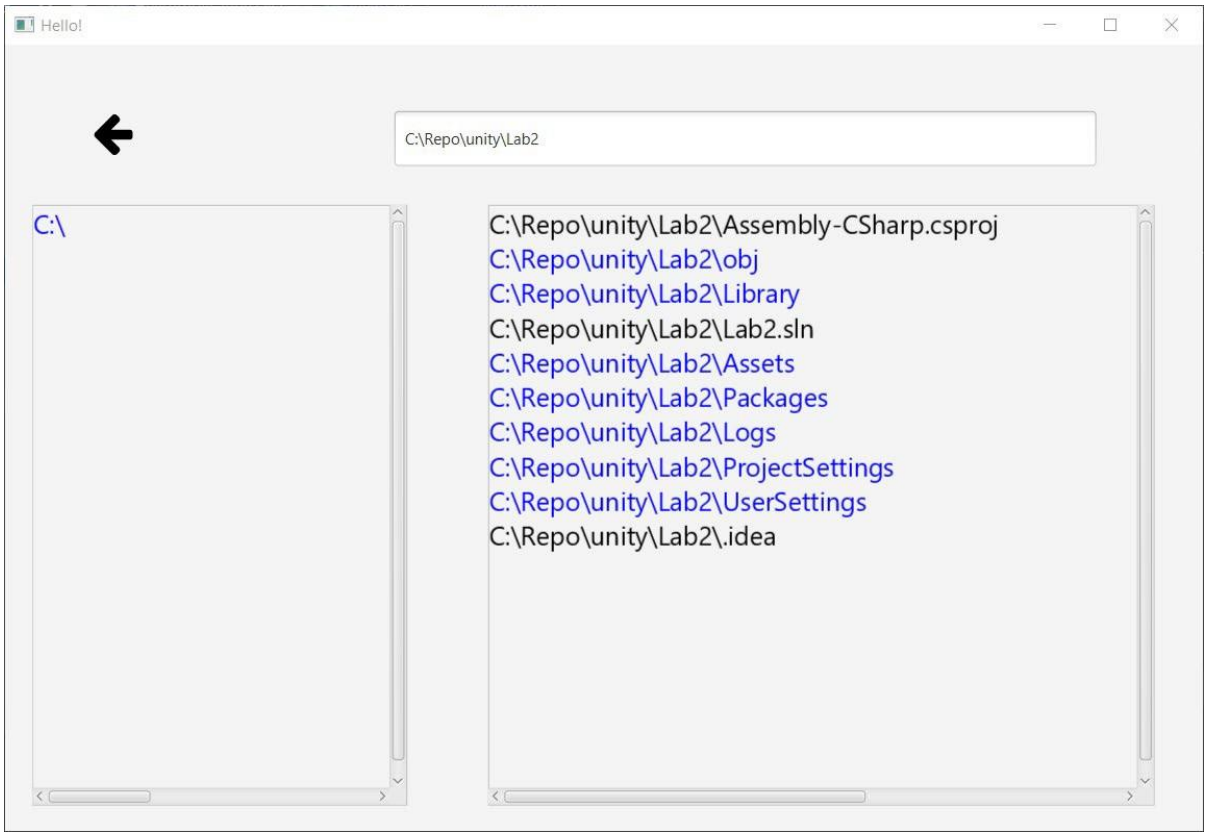
```
JavaFxObservable.eventsOf(left, MouseEvent.MOUSE_CLICKED)
    .subscribe(
        s -> {
            print.getChildren().clear();
            String[] split = path.getText().split("\\\\");
            String newPath = "";
            for (int i = 0; i < split.length - 1; i++) {
                newPath += split[i] + "\\";
            }
            path.setText(newPath);

        }
    );
```

Wygląd programu na starcie:



Wejście w głąb losowego folderu:



Wnioski:

Wykorzystanie JavyRx ma naprawdę szerokie spektrum. Istnieje masa bibliotek wykorzystujących Rx, w tym przypadku, wykorzystałem JavaFxRx, która jest rozszerzeniem o Rx dla zwykłej JavyFx. Posiada ona bardzo wygodne, gotowe metody dla JavyFx jak np. JavaFxObservable.eventsOf , która powoduje obserwacje danego elementu JavyFx.