Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №6

з дисципліни
«Об’єктно-орієнтоване програмування»

Виконав:	Перевірив:
студент групи ІМ-31 Литвиненко Сергій Андрійович номер у списку групи: 11	Порєв В. М.

Київ 2024

Варіант завдання

Програма Lab6

1. Користувач вводить значення n, Min, Max у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації.

Програма Object2

1. Створює вектор n дробових (double) чисел у діапазоні Min – Max

2. Показує числові значення у декількох стовпчиках та рядках у власному головному вікні

3. Записує дані в Clipboard Windows у текстовому форматі

Програма Object3

1. Зчитує дані з Clipboard Windows

2. Відображає графік y=f(x) у власному головному вікні Значення y – це значення вектора, x – індекси елементів. Графік, як в математиці – лінія, що проходить через точки (x,y) в порядку зростання x; осі координат з підписами числових значень x, y.

Вихідний текст програм

File: ./programs/generator/Main.java

```
import javafx.application.Application;

import javafx.stage.Stage;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;


public class Main extends Application {


  private final String pathToView =
"./resources/Main.fxml";

  private final String title = "Generator";


  public static void main(final String[] args) {

    launch(args);

  }


  @Override

  public void start(Stage stage) throws Exception {

    final Parent root =
FXMLLoader.load(getClass().getResource(pathToView));

    stage.setScene(new Scene(root));

    stage.setTitle(title);

    stage.show();

  }
```

```
}



File: ./programs/generator/controllers/MainController.java
package controllers;

import javafx.fxml.FXML;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import java.util.Map;
import org.json.JSONObject;
import javafx.scene.input.Clipboard;
import javafx.scene.input.ClipboardContent;
import javafx.application.Platform;
import javafx.beans.property.SimpleDoubleProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import listeners.InputStreamListener;

class Number {
  private final SimpleIntegerProperty index;
  private final SimpleDoubleProperty number;

  public Number(int index, double number) {
    this.index = new SimpleIntegerProperty(index);
    this.number = new SimpleDoubleProperty(number);
  }
  public SimpleIntegerProperty getIndex() { return
index; }
  public SimpleDoubleProperty getNumber() { return
number; }
}

public class MainController {
  @FXML
  private TableView<Number> tableView;

  @FXML
  private TableColumn<Number, Integer> columnIndex;

  @FXML
  private TableColumn<Number, Double> columnNumber;

  private double random(double min, double max) {
    return min + Math.random() * (max - min);
  }

  @FXML
  private void initialize() {
    columnIndex.setCellValueFactory((cellData) ->
cellData.getValue().getIndex().asObject());
```

```
    columnNumber.setCellValueFactory((cellData) ->
cellData.getValue().getNumber().asObject());

    final ObservableList<Number> numbers =
FXCollections.observableArrayList();

    tableView.setItems(numbers);

    final var listener =
InputStreamListener.getInstance();

    listener.on("data", (json) -> {

      numbers.clear();

      final var min = json.getDouble("min");

      final var max = json.getDouble("max");

      final var n = json.getInt("n");

      for (int index = 0; index < n; index++) {

        numbers.add(new Number(index, random(min, max)));

      }

      Platform.runLater(() -> {

        final var data = numbers.stream().map((num) ->
num.getNumber().getValue()).toList();

        final var clipboard =
Clipboard.getSystemClipboard();

        final var content = new ClipboardContent();

        final var clipboardContent = new
JSONObject(Map.of("data", data, "max", max, "min", min));

        content.putString(clipboardContent.toString());

        clipboard.clear();

        clipboard.setContent(content);

        final var message = Map.of(
          "receiver", "function",
          "service", "data"
        );
        System.out.print(new
JSONObject(message).toString());
      });
    });
  }
}
```

File: ./programs/generator/resources/Main.fxml

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

<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>

<TableView fx:id="tableView" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="500.0" prefWidth="300.0"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/22"
fx:controller="controllers.MainController">
  <columns>
    <TableColumn fx:id="columnIndex" prefWidth="150.0"
text="Index" />
    <TableColumn fx:id="columnNumber" prefWidth="150.0"
text="Number" />
```

```
  </columns>
</TableView>
```

File: ./programs/generator/listeners/InputStreamListener.java

```java
package listeners;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CompletableFuture;
import org.json.JSONObject;

import java.util.function.Consumer;

public class InputStreamListener {
  private static InputStreamListener instance = null;
  private static final Map<String,
List<Consumer<JSONObject>>> listeners = new HashMap<>();

  public static InputStreamListener getInstance() {
    if (instance != null) return instance;
    instance = new InputStreamListener();
    CompletableFuture.runAsync(() -> {
      final var isReader = new
InputStreamReader(System.in);
      try (final var reader = new
BufferedReader(isReader)) {
        while (true) {
          final var line = reader.readLine();
          if (line == null) continue;
          final var json = new JSONObject(line);
          final var service = json.getString("service");
          final var data = json.has("data") ?
json.getJSONObject("data") : new JSONObject();
          instance.emit(service, data);
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    });
    return instance;
  }

  public InputStreamListener on(final String eventName,
final Consumer<JSONObject> listener) {
    final var exists = listeners.containsKey(eventName);
    if (exists) listeners.get(eventName).add(listener);
    else listeners.put(eventName, List.of(listener));
    return this;
  }

  private InputStreamListener emit(final String eventName,
final JSONObject json) {
    final var exists = listeners.containsKey(eventName);
    if (!exists) return this;
    for (final var listener: listeners.get(eventName)) {
      listener.accept(json);
    }
    return this;
  }
}
```

File: ./programs/main/controllers/MenuController.java

```
package controllers;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
public class MenuController {
  final private String pathToDialg =
"../resources/Dialog.fxml";
  final private String title = "Enter data";

  @FXML
  private BorderPane borderPane;

  @FXML
  private void close() {
    Platform.exit();
  }

  @FXML
  private void start() throws Exception {
    final var root = borderPane.getScene().getWindow();
    final var gui =
(Parent)FXMLLoader.load(getClass().getResource(pathToDialg
));
    final var scene = new Scene(gui);
    final var stage = new Stage();
    stage.setScene(scene);
    stage.initOwner(root);
    stage.initModality(Modality.WINDOW_MODAL);
    stage.setTitle(title);
    stage.show();
  }
```

```
}


File: ./programs/main/controllers/DialogController.java
package controllers;

import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import org.json.JSONObject;

public class DialogController {

  @FXML
  private AnchorPane anchorPane;

  @FXML
  private TextField nField;

  @FXML
  private TextField minField;

  @FXML
  private TextField maxField;
  @FXML
  private void cancel() {
    final var window =
(Stage)anchorPane.getScene().getWindow();
    window.close();
  }

  @FXML
  private void ok() {
    final var n = nField.getText();
    final var min = minField.getText();
    final var max = maxField.getText();
    final var isNumber = "[+-]?\\d+(\\.\\d+)?";
    final var isNumbers = (
      n.matches("^\\d+$") &&
      min.matches(isNumber) &&
      max.matches(isNumber)
    );
    if (!isNumbers) return;
    if (Double.parseDouble(min) >=
Double.parseDouble(max)) return;
    nField.setText("");
    minField.setText("");
    maxField.setText("");
    try {
      final var json = new JSONObject();
      json.put("service", "data");
      json.put("receiver", "generator");
      final var data = new JSONObject();
      data.put("n", n);
      data.put("min", min);
      data.put("max", max);
      json.put("data", data);
      System.out.print(json.toString());
      cancel();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
File: ./programs/main/resources/Main.fxml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.control.Label?>
```

```
<BorderPane fx:id="borderPane" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controllers.MenuController">
  <top>
    <MenuBar BorderPane.alignment="CENTER">
      <menus>
         <Menu mnemonicParsing="false" text="File">
          <items>
            <MenuItem mnemonicParsing="false"
onAction="#close" text="Close" />
          </items>
        </Menu>
        <Menu>
          <graphic>
            <Label text="Start" onMouseClicked="#start"/>
          </graphic>
        </Menu>
      </menus>
    </MenuBar>
  </top>
</BorderPane>
```

File: ./programs/main/resources/Dialog.fxml

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

```
<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<AnchorPane fx:id="anchorPane" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="237.0" prefWidth="496.0"
xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controllers.DialogController">

   <children>

      <Text layoutX="168.0" layoutY="49.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Enter Data"
wrappingWidth="159.0654296875">

         <font>

            <Font size="29.0" />

         </font>

      </Text>

      <Text layoutX="80.0" layoutY="107.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="N"
wrappingWidth="24.0654296875">

         <font>

            <Font size="29.0" />

         </font>

      </Text>
      <Text layoutX="221.0" layoutY="107.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Min"
wrappingWidth="52.0654296875">
         <font>
            <Font size="29.0" />
         </font>
      </Text>
      <Text layoutX="373.0" layoutY="107.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Max"
wrappingWidth="62.0654296875">
         <font>
            <Font size="29.0" />
         </font>
      </Text>
      <TextField fx:id="nField" layoutX="22.0"
layoutY="150.0" prefHeight="24.0" prefWidth="140.0" />
      <TextField fx:id="minField" layoutX="178.0"
layoutY="150.0" prefHeight="24.0" prefWidth="140.0" />
      <TextField fx:id="maxField" layoutX="334.0"
layoutY="150.0" prefHeight="24.0" prefWidth="140.0" />
      <Button onAction="#ok" layoutX="401.0"
layoutY="188.0" mnemonicParsing="false" prefHeight="35.0"
prefWidth="69.0" text="Ok" />
      <Button onAction="#cancel" layoutX="327.0"
layoutY="188.0" mnemonicParsing="false" prefHeight="35.0"
prefWidth="69.0" text="Cancel" />
   </children>
</AnchorPane>
```

```
File: ./programs/main/Main.java
import javafx.application.Application;
import javafx.application.Platform;
import javafx.stage.Stage;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import listeners.InputStreamListener;

public class Main extends Application {

  private final double width = 900;
  private final double height = 900;
  private final String pathToView =
"./resources/Main.fxml";
  private final String title = "Lab6";

  public static void main(final String[] args) {
    launch(args);
  }

  @Override
  public void start(Stage stage) throws Exception {
    final BorderPane root =
FXMLLoader.load(getClass().getResource(pathToView));
```

```
    stage.setScene(new Scene(root));

    stage.setWidth(width);

    stage.setHeight(height);

    stage.setTitle(title);

    stage.show();

    final var listener =
InputStreamListener.getInstance();

    listener.on("close", (_) ->
Platform.runLater(stage::close));

  }

}
```

File: ./programs/main/listeners/InputStreamListener.java

```
package listeners;


import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.concurrent.CompletableFuture;

import org.json.JSONObject;


import java.util.function.Consumer;
```

```
public class InputStreamListener {
  private static InputStreamListener instance = null;
  private static final Map<String,
List<Consumer<JSONObject>>> listeners = new HashMap<>();

  public static InputStreamListener getInstance() {
    if (instance != null) return instance;
    instance = new InputStreamListener();
    CompletableFuture.runAsync(() -> {
      final var isReader = new
InputStreamReader(System.in);
      try (final var reader = new
BufferedReader(isReader)) {
        while (true) {
          final var line = reader.readLine();
          if (line == null) continue;
          final var json = new JSONObject(line);
          final var service = json.getString("service");
          final var data = json.has("data") ?
json.getJSONObject("data") : new JSONObject();
          instance.emit(service, data);
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    });
    return instance;
  }

  public InputStreamListener on(final String eventName,
final Consumer<JSONObject> listener) {
    final var exists = listeners.containsKey(eventName);
    if (exists) listeners.get(eventName).add(listener);
    else listeners.put(eventName, List.of(listener));
    return this;
  }

  private InputStreamListener emit(final String eventName,
final JSONObject json) {
    final var exists = listeners.containsKey(eventName);
    if (!exists) return this;
    for (final var listener: listeners.get(eventName)) {
      listener.accept(json);
    }
    return this;
  }
}
```

File: ./programs/function/Main.java

```
import javafx.application.Application;
import javafx.application.Platform;
import javafx.stage.Stage;
import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import listeners.InputStreamListener;


public class Main extends Application {

  private final String pathToView =
"./resources/Main.fxml";

  private final String title = "Function";


  public static void main(final String[] args) {

    launch(args);

  }


  @Override

  public void start(Stage stage) throws Exception {

    final Parent root =
FXMLLoader.load(getClass().getResource(pathToView));

    stage.setScene(new Scene(root));

    stage.setTitle(title);

    stage.show();

    final var listener =
InputStreamListener.getInstance();

    listener.on("close", (_) ->
Platform.runLater(stage::close));

  }

}
```

File: ./programs/function/resources/Main.fxml

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

<?import javafx.scene.canvas.Canvas?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity"
minHeight="-Infinity" minWidth="-Infinity"
xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controllers.MainController">
  <children>
    <Canvas fx:id="canvas" width="800" height="600" />
  </children>
</AnchorPane>
```

File: ./programs/function/controllers/MainController.java

```java
package controllers;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.input.Clipboard;
import javafx.util.Pair;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.util.ArrayList;

import org.json.JSONObject;
import listeners.InputStreamListener;

public class MainController {
  @FXML
  private Canvas canvas;
  private final double padding = 20;
  private final double ticksLength = 7;
  private final int yTiks = 10;
  private final double dotWidth = 6;

  private JSONObject getJson(final String source) {
    try {
      return new JSONObject(source);
    } catch (Exception e) {
      return null;
    }
  }

  public static String truncate(double input) {
    DecimalFormat decimalFormat = new
DecimalFormat("##.##");

    decimalFormat.setRoundingMode(RoundingMode.DOWN);

    String formatResult = decimalFormat.format(input);

    return formatResult;

  }


  private void drawYTicks(final GraphicsContext context,
int maxY) {

    final var height = canvas.getHeight();

    final var center = height / 2;

    final var heightLength = center - padding;

    final var step = heightLength / yTiks;

    for (int index = 0; index < yTiks; index++) {

      final var down = center + (index + 1) * step;

      final var up = center - (index + 1) * step;

      final var cost = Math.abs(maxY / (double)yTiks *
(index + 1));

      context.strokeLine(padding - ticksLength, up,
padding + ticksLength, up);

      context.strokeLine(padding - ticksLength, down,
padding + ticksLength, down);

      context.strokeText(truncate(cost), padding +
ticksLength, up);

      context.strokeText(truncate(-cost), padding +
ticksLength, down);

    }

  }
  private void drawXTicks(final GraphicsContext context,
int maxX) {
    final var width = canvas.getWidth();
    final var height = canvas.getHeight();
    final var widthLength = width - 2 * padding;
    final var step = widthLength / (maxX - 1);
    final var xHeight = height / 2;
    for (int index = 0; index < maxX; index++) {
      final var position = index * step + padding;
      context.strokeLine(position, xHeight - ticksLength,
position, xHeight + ticksLength);
      context.strokeText(String.valueOf(index), position,
xHeight + ticksLength * 2);
    }
  }

  private void drawAxes(int maxX, int maxY) {
    final var width = canvas.getWidth();
    final var height = canvas.getHeight();
    final var context = canvas.getGraphicsContext2D();
    context.strokeLine(padding / 2, height / 2, width -
padding / 2, height / 2);
    context.strokeLine(padding, padding / 2, padding,
height - padding / 2);
    context.strokeText("y", padding / 2, padding / 2);
    context.strokeText("x", width - padding / 2, height /
2);
    drawXTicks(context, maxX);
    drawYTicks(context, maxY);
  }

  private void curwe(final ArrayList<Pair<Double, Double>>
points) {
    final var context = canvas.getGraphicsContext2D();
    final var p = points.getFirst();
    context.strokeOval(p.getKey() - dotWidth / 2,
p.getValue()- dotWidth / 2, dotWidth, dotWidth);
    for (int index = 0; index < points.size() - 1;
index++) {
      final var first = points.get(index);
      final var second = points.get(index + 1);
      final var x1 = first.getKey();
      final var y1 = first.getValue();
      final var x2 = second.getKey();
      final var y2 = second.getValue();
      context.strokeOval(x2 - dotWidth / 2, y2 - dotWidth
/ 2, dotWidth, dotWidth);
      context.strokeLine(x1, y1, x2, y2);
    }
  }

  @FXML
  private void initialize() {
    final var listener =
InputStreamListener.getInstance();
    listener.on("data", (_) -> {
      Platform.runLater(() -> {
        final var clipboard =
Clipboard.getSystemClipboard();
        final var content = clipboard.getString();
        final var json = getJson(content);
        if (json == null || !json.has("data")) return;
        final var data = json.getJSONArray("data");
        final var min = json.getDouble("min");
        final var max = json.getDouble("max");
        final var points = new ArrayList<Double>();
        for (final var point: data) {
          final var number =
Double.valueOf(point.toString());
          points.add(number);
        }
        final var absMax = Math.max(Math.abs(min),
Math.abs(max));
        final var width = canvas.getWidth();
        final var height = canvas.getHeight();
        canvas.getGraphicsContext2D().clearRect(0, 0,
canvas.getWidth(), canvas.getHeight());
        drawAxes(points.size(), (int)Math.ceil(absMax));
        final var widthLength = width - 2 * padding;
        final var step = widthLength / (points.size() -
1);

        final var normalisedPoints = new
ArrayList<Pair<Double, Double>>();

        final var minimum = height - padding + (min +
absMax) * (2 * padding - height) / (2 * absMax);

        final var maximum = height - padding + (max +
absMax) * (2 * padding - height) / (2 * absMax);

        for (int index = 0; index < points.size();
index++) {

          final var y = points.get(index);

          final var xPos = index * step + padding;

          final var yPos = minimum + (y - min) * (maximum
- minimum) / (max - min);

          final var point = new Pair<>(xPos, yPos);

          normalisedPoints.add(point);

        }

        curwe(normalisedPoints);

      });

    });

  }

}
```

File: ./programs/function/listeners/InputStreamListener.ja
va

```
package listeners;
```

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CompletableFuture;
import org.json.JSONObject;

import java.util.function.Consumer;

public class InputStreamListener {
  private static InputStreamListener instance = null;
  private static final Map<String,
List<Consumer<JSONObject>>> listeners = new HashMap<>();

  public static InputStreamListener getInstance() {
    if (instance != null) return instance;
    instance = new InputStreamListener();
    CompletableFuture.runAsync(() -> {
      final var isReader = new
InputStreamReader(System.in);
      try (final var reader = new
BufferedReader(isReader)) {
        while (true) {
          final var line = reader.readLine();
          if (line == null) continue;
          final var json = new JSONObject(line);
          final var service = json.getString("service");
          final var data = json.has("data") ?
json.getJSONObject("data") : new JSONObject();
          instance.emit(service, data);
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    });
    return instance;
  }

  public InputStreamListener on(final String eventName,
final Consumer<JSONObject> listener) {
    final var exists = listeners.containsKey(eventName);
    if (exists) listeners.get(eventName).add(listener);
    else listeners.put(eventName, List.of(listener));
    return this;
  }

  private InputStreamListener emit(final String eventName,
final JSONObject json) {
    final var exists = listeners.containsKey(eventName);
    if (!exists) return this;
    for (final var listener: listeners.get(eventName)) {
      listener.accept(json);
    }
    return this;
  }
}
```

File: ./main.js

```
'use strict';

const config = require('./config.json');
const find = require('./find.js');
const exists = require('./exists.js');
const fsp = require('node:fs/promises');
const execute = require('./execute.js')(config);
const compile = require('./compile.js')(config, find);
const moveResources =
require('./moveResources.js')(config, exists);
const events = require('node:events');

const pipe = (...functions) => {
  const next = (value, index = 0) => {
    if (index >= functions.length) return value;
    const answer = functions[index](value);
    const callback = (arg) => next(arg, index + 1);
    return answer.then ? answer.then(callback) :
callback(answer);
  };
  return (value) => next(value);
};

const compileProject = async () => {
  const programs = await
fsp.readdir(config.programsFolder);
  const compiles = programs.map(compile);
  const copies = programs.map(moveResources);
  await Promise.all(compiles);
  await Promise.all(copies);
};

const manageProcesses = async (project) => {
  const processes = new Map();
  const manager = async (name) => {
    const subprocess = await execute(name);
    processes.set(name, subprocess);
    subprocess.stderr.pipe(process.stderr);
    subprocess.stdout.setEncoding('utf-8');
    subprocess.stdout.setDefaultEncoding('utf-8');
    const subprocesses = new Set();
    subprocess.stdout.on('data', async (chunk) => {
      const { service, receiver, data } =
JSON.parse(chunk);
      if (!processes.has(receiver)) {
        subprocesses.add(await manager(receiver));
      }
      const subprocess = processes.get(receiver);
      const message = JSON.stringify({ service, data });
      subprocess.stdin.write(message + '\n');
    });
    subprocess.once('close', () => {
      processes.delete(name);
      for (const subprocess of subprocesses) {
        subprocess.kill();
      }
    });
    return subprocess;
  };
  return manager(project);
};

const main = pipe(
  compileProject,
  () => manageProcesses(config.mainProject),
  (mainProcess) => events.once(mainProcess, 'close'),
  () => fsp.rm(config.target, { recursive: true, force:
true })
);

main();
```

File: ./find.js

```
'use strict';

const path = require('node:path');
const fsp = require('node:fs/promises');

const find = async (folder, pattern) => {
  const result = new Set();
  const files = await fsp.readdir(folder, { withFileTypes:
true });
  for (const file of files) {
    const { name: filename } = file;
    const fullpath = path.join(folder, filename);
    if (file.isDirectory()) {
      const subset = await find(fullpath, pattern);
      for (const file of subset) result.add(file);
      continue;
    }
    const match = filename.match(pattern);
    if (match) result.add(fullpath);
  }
  return result;
};

module.exports = find;
```

File: ./compile.js

```
'use strict';

const path = require('node:path');
const { once } = require('node:events');
const child_process = require('node:child_process');

module.exports = (config, find) => async (program) => {
  const fullpath = path.join(config.programsFolder,
program);
  const targetFolder = path.join(config.target, program);
  const files = await find(fullpath, '.java$');
  const args = ['-d',
targetFolder, ...config.libs, ...files];
  const subprocess = child_process.spawn('javac', args,
{ stdio: 'inherit' });
  return once(subprocess, 'exit').then(([status]) =>
status);
};
```

File: ./config.json

```
{
  "mainProject": "main",
  "mainFile": "Main",
  "libs": [
    "--module-
path=/home/serhii/programming/code/java/libs/javafx/lib:/h
ome/serhii/programming/code/java/libs/javax/lib/",
    "--add-modules=javafx.controls,javafx.fxml,org.json"
  ],
  "target": "bin",
  "programsFolder": "programs",
  "resourcesFolder": "resources"
}
```

```
File: ./moveResources.js
'use strict';

const path = require('node:path');
const fsp = require('node:fs/promises');

module.exports = (config, exists) => {
  const { programsFolder, target, resourcesFolder } =
config;
  return async (programName) => {
    const oldpath = path.join(programsFolder, programName,
resourcesFolder);
    const newpath = path.join(target, programName,
resourcesFolder);
    const present = await exists(oldpath);
    if (!present) return;
    return fsp.cp(oldpath, newpath, { recursive: true,
force: true });
  };
};
```

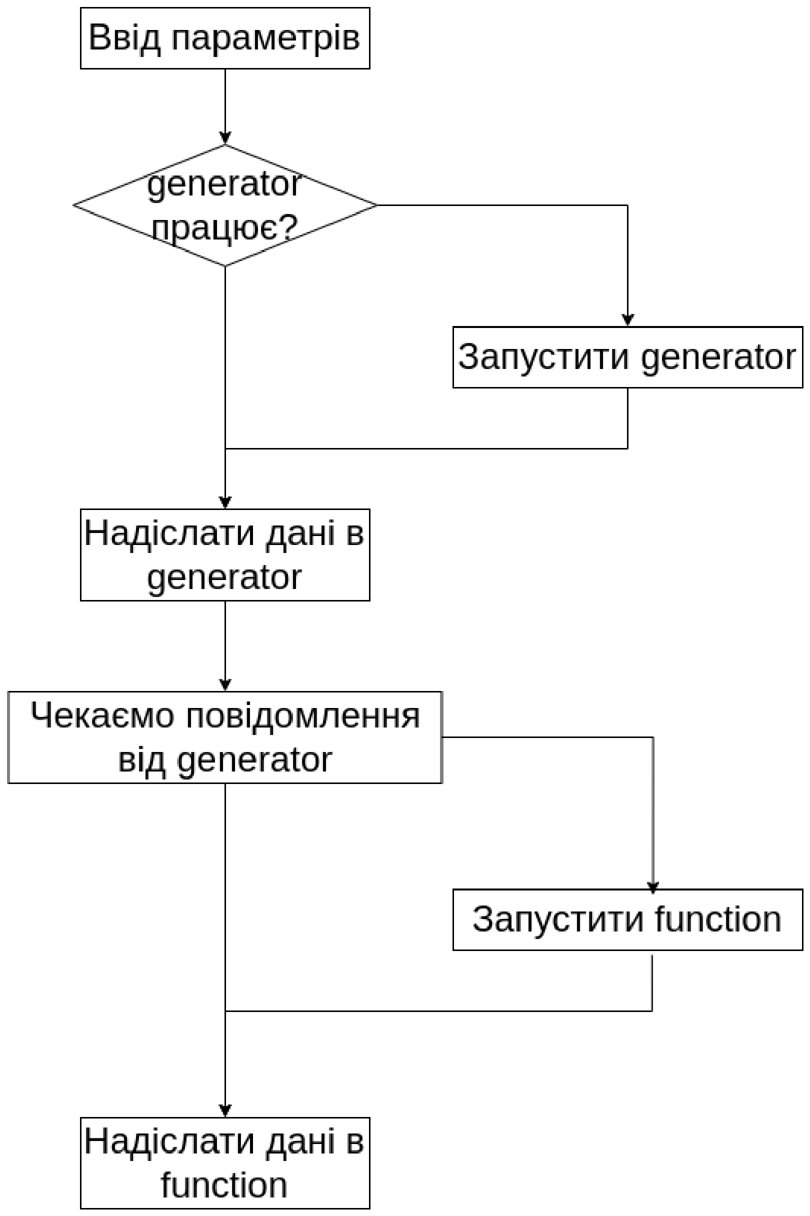
File: ./execute.js

```
'use strict';

const path = require('node:path');
const { once } = require('node:events');
const child_process = require('node:child_process');

module.exports = (config) => {
  const { target, mainProject, mainFile, libs } = config;
  return (project = mainProject) => {
    const fullpath = path.join(target, project);
    const args = ['-cp', fullpath, ...libs, mainFile];
    const process = child_process.spawn('java', args);
    return once(process, 'spawn').then(() => process);
  };
};
```

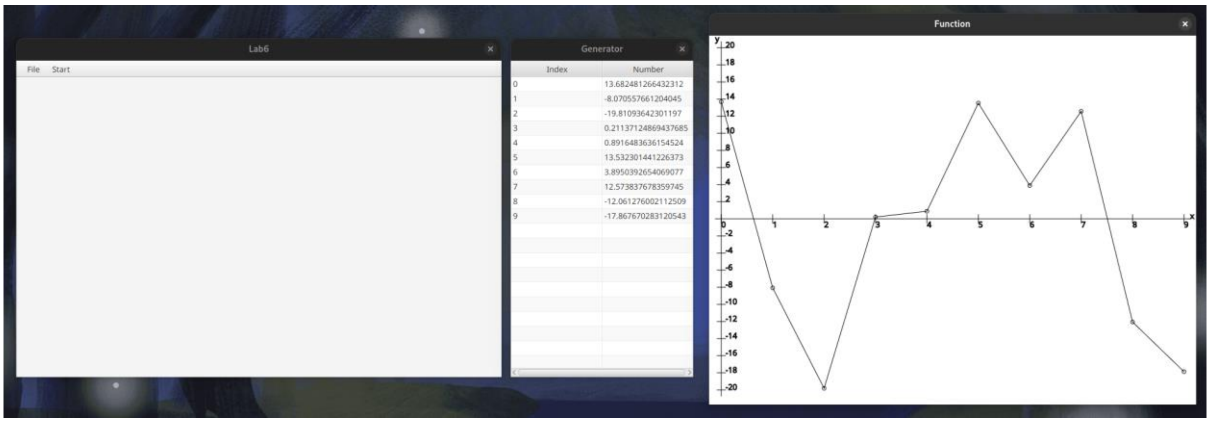
File: ./exists.js

```
'use strict';

const fsp = require('node:fs/promises');

module.exports = (file) => fsp.access(file).then(() =>
true, () => false);
```

Схема послідовності надсилання-обробки повідомлень


Ввід параметрів
generator працює?
Запустити generator
Надіслати дані в generator
Чекаємо повідомлення від generator
Запустити function
Надіслати дані в function

Скріншоти виконання



Index	Number
0	13.682481266432312
1	-8.070557661204045
2	-19.81093642301197
3	0.21137124869437685
4	0.8916483636154524
5	13.532301441226373
6	3.8950392654069077
7	12.573837678359745
8	-12.061276002112509
9	-17.867670283120543

Висновки

Під час виконання лабораторної роботи я здобув навички використання інкапсуляції, абстрактних типів, успадкування та поліморфізму, вичвив патерни Singleton та Observer, навчився обмінюватися повідомленнями між процесами за допомогою stdio потоків та покращав свої навички програмування у Java SDK та Node.js середовищі.