LAPORAN RESMI

Tugas Relasi Class Diagram

Mata Kuliah: Praktek Pemrograman Berbasis Objek

Disusun oleh:

Bayu Hadi Leksana (3122500046)

2 D3 Teknik Informatika B

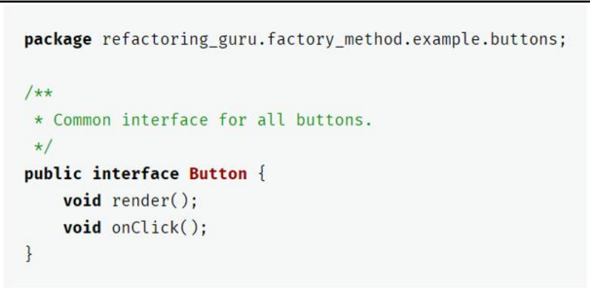
Dosen Pengampu: Andhik Ampuh Yunanto S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

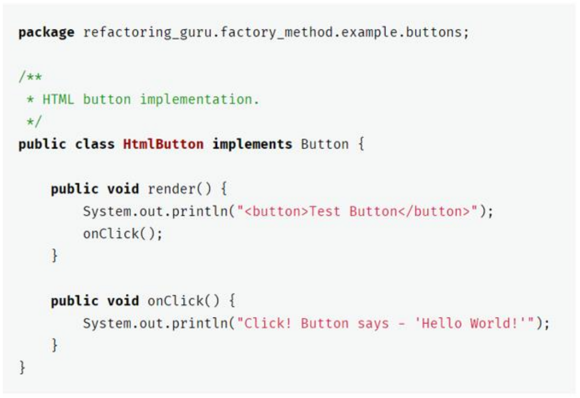
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

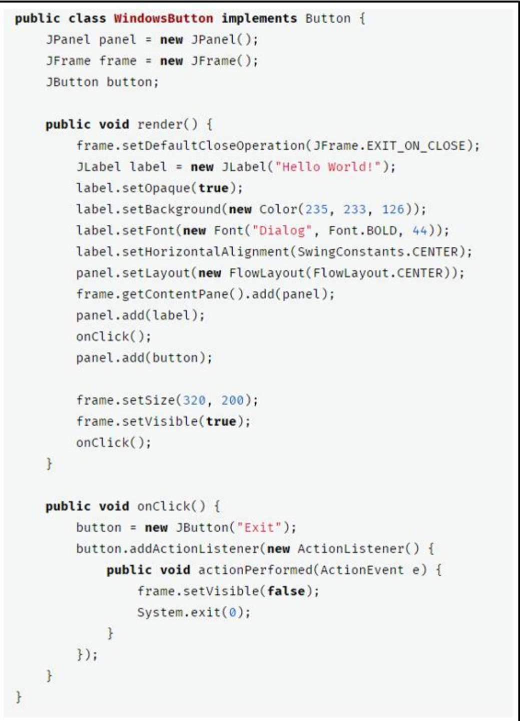
Kode:



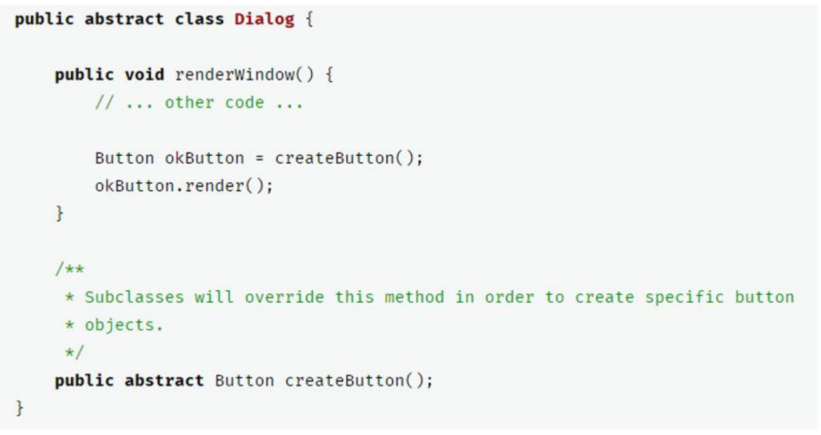
```java
package refactoring_guru.factory_method.example.buttons;

/**
 * Common interface for all buttons.
 */
public interface Button {
    void render();
    void onClick();
}
```



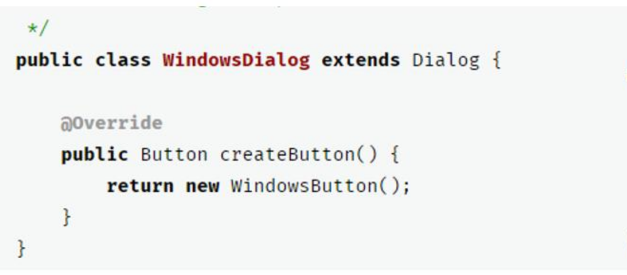
```java
package refactoring_guru.factory_method.example.buttons;

/**
 * HTML button implementation.
 */
public class HtmlButton implements Button {

    public void render() {
        System.out.println("<button>Test Button</button>");
        onClick();
    }

    public void onClick() {
        System.out.println("Click! Button says - 'Hello World!'");
    }
}
```



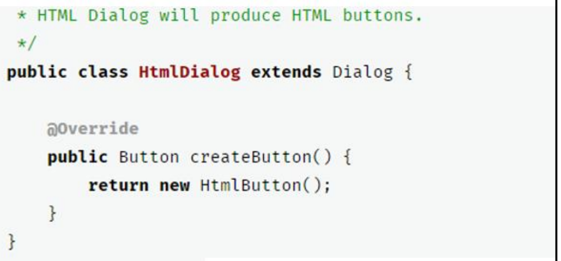
```java
public class WindowsButton implements Button {
    JPanel panel = new JPanel();
    JFrame frame = new JFrame();
    JButton button;

    public void render() {
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World!");
        label.setOpaque(true);
        label.setBackground(new Color(235, 233, 126));
        label.setFont(new Font("Dialog", Font.BOLD, 44));
        label.setHorizontalAlignment(SwingConstants.CENTER);
        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
        frame.getContentPane().add(panel);
        panel.add(label);
        onClick();
        panel.add(button);

        frame.setSize(320, 200);
        frame.setVisible(true);
        onClick();
    }

    public void onClick() {
        button = new JButton("Exit");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                frame.setVisible(false);
                System.exit(0);
            }
        });
    }
}
```



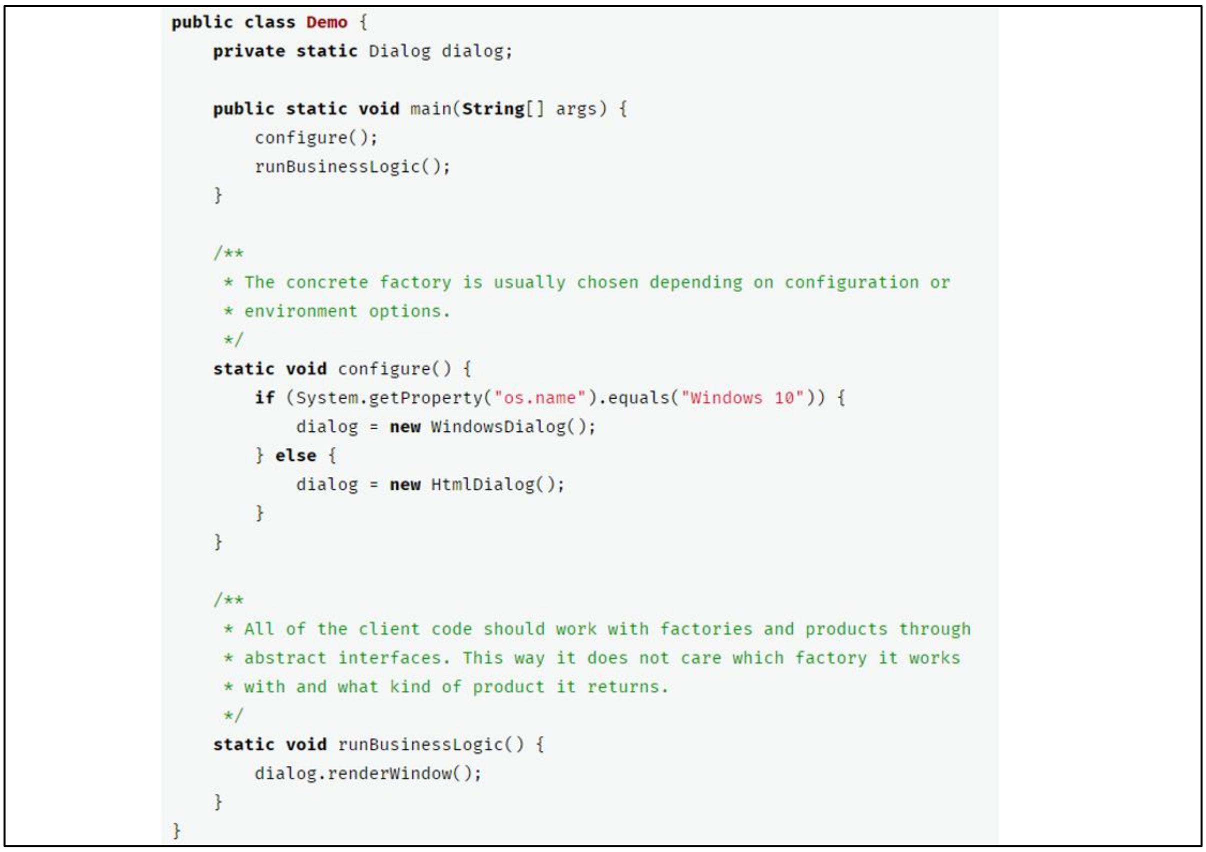
```java
public abstract class Dialog {

    public void renderWindow() {
        // ... other code ...

        Button okButton = createButton();
        okButton.render();
    }

    /**
     * Subclasses will override this method in order to create specific button
     * objects.
     */
    public abstract Button createButton();
}
```



```java
 */
public class WindowsDialog extends Dialog {

    @Override
    public Button createButton() {
        return new WindowsButton();
    }
}
```

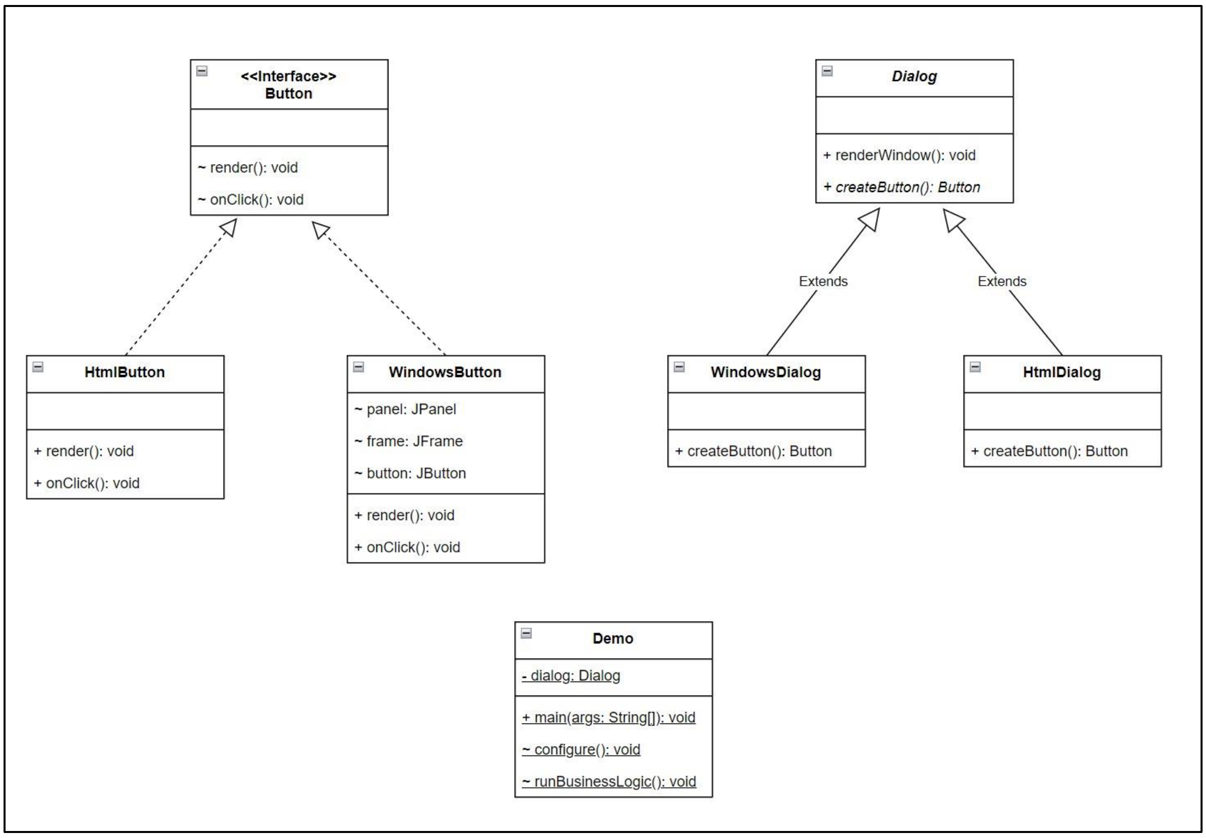


```java
 * HTML Dialog will produce HTML buttons.
 */
public class HtmlDialog extends Dialog {

    @Override
    public Button createButton() {
        return new HtmlButton();
    }
}
```



```java
public class Demo {
    private static Dialog dialog;

    public static void main(String[] args) {
        configure();
        runBusinessLogic();
    }

    /**
     * The concrete factory is usually chosen depending on configuration or
     * environment options.
     */
    static void configure() {
        if (System.getProperty("os.name").equals("Windows 10")) {
            dialog = new WindowsDialog();
        } else {
            dialog = new HtmlDialog();
        }
    }

    /**
     * All of the client code should work with factories and products through
     * abstract interfaces. This way it does not care which factory it works
     * with and what kind of product it returns.
     */
    static void runBusinessLogic() {
        dialog.renderWindow();
    }
}
```

Class Diagram:



<<Interface>>
Button
~ render(): void
~ onClick(): void
Dialog
+ renderWindow(): void
+ createButton(): Button
Extends
Extends
HtmlButton
+ render(): void
+ onClick(): void
WindowsButton
~ panel: JPanel
~ frame: JFrame
~ button: JButton
+ render(): void
+ onClick(): void
WindowsDialog
+ createButton(): Button
HtmlDialog
+ createButton(): Button
Demo
- dialog: Dialog
+ main(args: String[]): void
~ configure(): void
~ runBusinessLogic(): void