# Assignment 2

| Name | Student ID | Course | University |
| --- | --- | --- | --- |
| Andrii Yatsura | 19387 | Introduction to Programming | Wrocław Academy of Business |

# Question 1

| | Class | Object |
| --- | --- | --- |
| **1.** | Weekdays | Friday |
| **2.** | Language | Java |
| **3.** | Bird | Eagle |
| **4.** | Employee | Bob |
| **5.** | Student | Alice |

# Question 2

The following Java code defines a method called `randomColor` that generates a random color using RGB values and returns a `Color` object. The main method demonstrates its usage by printing a randomly generated color.

```
package Assignment_02;

import java.util.Random;

class Color {
    private int red;
    private int green;
```

```
    private int blue;

    public Color(int red, int green, int blue) {
        this.red = validateColorValue(red);
        this.green = validateColorValue(green);
        this.blue = validateColorValue(blue);
    }

    private int validateColorValue(int value) {
        return Math.min(255, Math.max(0, value));
    }

    @Override
    public String toString() {
        return ("Color(red=" + red
                    + ", green=" + green
                    + ", blue=" + blue + ')');
    }
}

public class Theoretical_01 {
    public static Color randomColor() {
        Random rand = new Random();
        int red = rand.nextInt(256);
        int green = rand.nextInt(256);
        int blue = rand.nextInt(256);

        return new Color(red, green, blue);
    }

    public static void main(String[] args) {
        Color randomColor = randomColor();

        System.out.println("Random Color: " + randomColor.toString());
    }
}
```

# Question 3

This Java code defines an interface `Visible` with two methods: `makeVisible` and `makeInvisible`. The class `ExampleClass` implements this interface, providing functionality to make an object visible or invisible. Each method returning boolean depending on whether the

change happened, so if the object's visibility changed after invoking the method, it returns `true`, and otherwise, if the visibility stayed the same, `false`.

```
public interface Visible {
    boolean makeVisible();
    boolean makeInvisible();
}

public class ExampleClass implements Visible {
    private boolean isVisible;

    public ExampleClass() {
        isVisible = true;
    }

    @Override
    public boolean makeVisible() {
        if (!isVisible) {
            isVisible = true;
            return true;
        }
        return false;
    }

    @Override
    public boolean makeInvisible() {
        if (isVisible) {
            isVisible = false;
            return true;
        }
        return false;
    }
}
```