

For our upcoming assignments and projects, we will be using the Visual Paradigm tool to create and work with class diagrams. You can access and utilize the tool through the following link: [Visual Paradigm Class Diagram Tool](#).

Q1: Create a simple bank account management system in Java. The system should have the following requirements:

1. Classes:

- BankAccount: Represents a bank account with attributes for account number, account balance, and customer name.
- Bank: Manages multiple bank accounts and provides methods to add accounts, display account details, and perform deposit and withdrawal operations.

2. Methods:

- BankAccount:
 - Constructor to initialize account details.
 - Methods to deposit and withdraw money.
- Bank:
 - Method to add a new account.
 - Method to display details of all accounts.
 - Method to perform deposit and withdrawal operations based on account number.

3. Main Program:

- Create a bank and add multiple bank accounts.
- Perform various deposit and withdrawal operations.
- Display the details of all bank accounts.

Solution:

```
class BankAccount {  
    private int accountNum;  
    private double accountBalance;  
    private String customerName;  
  
    public BankAccount(int accountNum, double accountBalance, String customerName) {  
        this.accountNum = accountNum;  
        this.accountBalance = accountBalance;  
        this.customerName = customerName;  
    }  
  
    public void deposit(double amount) {  
        accountBalance += amount;  
    }  
  
    public void withdraw(double amount) {  
        if (accountBalance >= amount) {  
            accountBalance -= amount;  
        } else {  
            System.out.println("Insufficient balance!");  
        }  
    }  
  
    public String getDetails() {  
        return "Account Number: " + accountNum + ", Customer Name: " + customerName + ", Balance:  
" + accountBalance;  
    }  
}
```

```
class Bank {  
    private BankAccount[] accounts;  
    private int accountCount;  
    public Bank(int size) {  
        accounts = new BankAccount[size];  
        accountCount = 0;  
    }  
    public void addAccount(BankAccount account) {  
        if (accountCount < accounts.length) {  
            accounts[accountCount++] = account;  
        } else { System.out.println("Bank is full! Cannot add more accounts.");  
        }  
    }  
    public void displayAccounts() {  
        for (int i = 0; i < accountCount; i++) {  
            System.out.println(accounts[i].getDetails());  
        }  
    }  
    public void deposit(int accountNum, double amount) {  
        for (int i = 0; i < accountCount; i++) {  
            if (accounts[i].getDetails().contains("Account Number: " + accountNum)) {  
                accounts[i].deposit(amount);  
                return;  
            }  
        }  
        System.out.println("Account not found!");  
    }  
}
```

```
public void withdraw (int accountNum, double amount) {  
    for (int i = 0; i < accountCount; i++) {  
        if (accounts[i].getDetails().contains("Account Number: " + accountNum)) {  
            accounts[i].withdraw(amount);  
            return; }  
        System.out.println("Account not found!");  
    }  
}  
public class BankDemo {  
    public static void main(String[] args) {  
        Bank bank = new Bank(10);  
        BankAccount account1 = new BankAccount(101, 1000, "Alice");  
        BankAccount account2 = new BankAccount(102, 2000, "Bob");  
        BankAccount account3 = new BankAccount(103, 3000, "Charlie");  
        bank.addAccount(account1);  
        bank.addAccount(account2);  
        bank.addAccount(account3);  
        bank.displayAccounts();  
        System.out.println("\nDepositing $500 into account 101");  
        bank.deposit(101, 500);  
        System.out.println("\nWithdrawing $300 from account 102");  
        bank.withdraw(102, 300);  
        System.out.println("\nUpdated account details:");  
        bank.displayAccounts();  
    }  
}
```

Q2: Write a Java program consisting of two classes, Box and BoxDemo, that demonstrates the following:

1. Define a Box class with three public double attributes: width, height, and depth.
2. Include a default constructor and a parameterized constructor in the Box class.
3. Include a method getVolume in the Box class that calculates and returns the volume of the box.
4. In the BoxDemo class, perform the following:
 - Create an instance of Box using the default constructor and set its dimensions.
 - Create another instance of Box using the parameterized constructor.
 - Assign the second instance to a new Box reference.
 - Modify the depth of the new reference.
 - Print the depth of the original second instance to show that it has been updated.
 - Calculate and print the volume of the updated second instance.

Solution

```
class Box {  
    public double width;  
    public double height;  
    public double depth;  
  
    Default constructor  
    Box() {}  
  
    Parameterized constructor  
    Box(double w, double h, double d) {  
        width = w;  
        height = h;  
        depth = d;  
    }  
  
    Method to calculate volume  
    double getVolume() {  
        double volume = width * height * depth;  
        return volume;  
    }  
}
```

```
class BoxDemo {  
    public static void main(String[] args) {  
  
        Creating an instance using the default constructor  
  
        Box box1 = new Box();  
  
        box1.width = 2;  
  
        box1.height = 4;  
  
        box1.depth = 6;  
  
  
        Creating an instance using the parameterized constructor  
  
        Box box2 = new Box(4, 10, 12);  
  
  
        Assigning the second instance to a new reference  
  
        Box box3 = box2;  
  
  
        Modifying the depth of the new reference  
  
        box3.depth = 14;  
  
  
        Printing the depth of the original second instance  
  
        System.out.println(box2.depth); // Output: 14  
  
  
        Calculating and printing the volume of the updated second instance  
  
        double volume = box2.getVolume();  
  
        System.out.println("Volume of box2: " + volume); // Output: 560.0  
  
    }  
}
```

Q3: You are required to create a Java program that performs the following tasks using the `java.util.Random` class:

- Generate 5 random integers between 1 and 50.
- Store these random integers in an array.
- Calculate and print the sum and average of these integers.

Provide a detailed implementation of the program.

Solution

```
import java.util.Random;

public class RandomNumberExample {

    public static void main(String[] args) {

Create an instance of Random

        Random random = new Random();

Array to store random integers

        int[] randomNumbers = new int[5];

Generate 5 random integers between 1 and 50

        for (int i = 0; i < 5; i++) {

            randomNumbers[i] = random.nextInt(50) + 1;

            System.out.println("Random number [" + (i + 1) + "]: " + randomNumbers[i]);

        }

Calculate the sum of the random integers

        int sum = 0;

        for (int num : randomNumbers) {

            sum += num;

        }

        System.out.println("Sum of random numbers: " + sum);

Calculate the average of the random integers

        double average = (double) sum / randomNumbers.length;

        System.out.println("Average of random numbers: " + average);

    }


}
```

Lab Work (10 Points): Create a Java class named Point that contains the following:

1. Two private data members x and y of type Integer (Wrapper class) representing the coordinates of the point. (1 Point)
2. A constructor that takes two parameters to initialize the x and y coordinates. Provide a default constructor that initializes the point at the origin (0, 0). (1 Point)
3. A method setLocation(int x, int y) that sets the coordinates of the point. (1 Point)
4. A method translate(int dx, int dy) that adjusts the coordinates of the point by the given amounts. (1 Point)
5. A method distance(Point p) that calculates the distance from this point to another point p. Use java.lang.Math to calculate the distance. (1 Point)
6. Use java.util.Scanner to input the coordinates of the points from the user. . (1 Point)
7. Use Random class to generate random coordinates for a point. (1 Point)
8. Illustrate the difference between reference data types and primitive types: (2 Points)
 - o Demonstrate how modifying a primitive type variable (e.g., int) affects its value and does not impact other variables (by value).
 - o Show how modifying a reference type variable (e.g., a Point object) affects the object and any other references to it (by reference).

Additionally, provide a test class to demonstrate the functionality of the Point class, including: (1 Point)

1. Creating instances using both constructors.
2. Using setLocation and translate methods.
3. Calculating the distance between two points.
4. Using Scanner to input point coordinates from the user.
5. Generating and using random coordinates for points.
6. Illustrating the difference between reference data types and primitive types with appropriate examples.

	Point
- x: int - y: int	
+ Point() + Point(x: int, y: int) + setLocation(x: int, y: int): void + translate(dx: int, dy: int): void + distance(p: Point): double	