

# Assignment 1

Contributor: Nguyen Quang Huy - 1677664

## Problem 1

1. What are the differences between procedural programming and object oriented programming?

Figure 1: Problem 1 Task Requirement

### Core Concepts

- **Procedural Programming:** Focuses on functions and procedures to execute a sequence of tasks.
- **OOP:** Centers on objects, which encapsulate data and behaviors (methods).

### Structure

- **Procedural:** Organized in a linear flow of functions.
- **OOP:** Organized around classes and objects.

### Data Handling

- **Procedural:** Data and functions are separate; data is passed to functions.
- **OOP:** Data and functions are bundled; methods operate on object data.

### Key Features

- **Encapsulation:** OOP promotes data hiding and access control; procedural programming has minimal encapsulation.
- **Inheritance:** OOP supports inheritance for code reuse; procedural does not.
- **Polymorphism:** OOP allows method overloading and overriding; procedural has limited support.

### Use Cases

- **Procedural:** Best for simple scripts and tasks.
- **OOP:** Ideal for complex applications requiring modularity and reusability.

## Problem 2

2. Find an example in the real world where the Object Technology is applied with success.

Figure 2: Problem 2 Task Requirement

- **Banking Systems:** Many banking applications use Java for secure, scalable, and maintainable systems, managing accounts, transactions, and user interactions.
- **E-commerce Platforms:** Platforms like eBay and Amazon utilize Java to handle extensive product catalogs, user data, and transactions effectively.
- **Healthcare Systems:** Java is used in healthcare applications for managing patient records, scheduling, and billing, ensuring data integrity and security.

## Problem 3

3. Setup the environment for compiling and running Java programs. Follows the guide in the section 1.10 Test-Driving a Java Application in the text book Java How to Program 9th edition, run the ATMCaseStudy and take the snapshot of the screen in each step.

Figure 3: Problem 3 Task Requirement

```
C:\Users\May\OneDrive\Desktop\ATM- java ATMCaseStudy
Picked up JAVA_TOOL_OPTIONS: D:\jdk-11.0.2\bin\java.exe;D:\jdk-11.0.2\bin\java.exe;D:\jdk-11.0.2\bin\java.exe
Welcome!

Please enter your account number: 12345
Enter your PIN: 54321

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
- Available balance: $1,000.00
- Total balance: $1,200.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 2

Withdrawal Menu:
1 - $0
2 - $40
3 - $60
4 - $100
5 - $200
6 - Cancel transaction

Choose a withdrawal amount: 4

Please take your cash now.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
- Available balance: $900.00
- Total balance: $1,100.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 4

Exiting the system...

Thank you! Goodbye!

Welcome!

Please enter your account number: 
```

Figure 4: Problem 3 Runtime Output

## Assignment 2

Contributor: Nguyen Quang Huy - 1677664

## 2.1 DisplayInitials.java

1. Write a program displaying on the screen two first letters of your name. For example: Thủy → Display TH:

TTTTTTTT	H	H
T	H	H
T	HHHHHHHH	
T	H	H
T	H	H

Figure 5: 2.1 Task Requirement

```

1 import javax.swing.JOptionPane;
2
3 public class DisplayIntegers {
4     public static void main(String[] args) {
5         String firstInput = JOptionPane.showInputDialog("Hay nhap so thu 1:");
6         String secondInput = JOptionPane.showInputDialog("Hay nhap so thu 2:");
7
8         int firstNumber = Integer.parseInt(firstInput);
9         int secondNumber = Integer.parseInt(secondInput);
10
11         JOptionPane.showMessageDialog(null, "So thu 1: " + firstNumber);
12         JOptionPane.showMessageDialog(null, "So thu 2: " + secondNumber);
13     }
14 }

```

1: DisplayInitials.java

## 2.2 (Ex17 - Book) IngredientAdjuster.java

## 17. Ingredient Adjuster

A cookie recipe calls for the following ingredients:

- 1.5 cups of sugar
- 1 cup of butter
- 2.75 cups of flour

The recipe produces 48 cookies with these amounts of the ingredients. Write a program that asks the user how many cookies he or she wants to make, and then displays the number of cups of each ingredient needed for the specified number of cookies.

Figure 6: IngredientAdjuster Task Requirement

```

1 import java.util.Scanner;

3 public class IngredientAdjuster {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);

7         final double sugarPerCookie = 1.5 / 48;
8         final double butterPerCookie = 1.0 / 48;
9         final double flourPerCookie = 2.75 / 48;

11        System.out.print("Enter the number of cookies you want to make: ");
12        int cookiesWanted = scanner.nextInt();

14        double sugarNeeded = sugarPerCookie * cookiesWanted;
15        double butterNeeded = butterPerCookie * cookiesWanted;
16        double flourNeeded = flourPerCookie * cookiesWanted;

18        System.out.printf("For %d cookies, you need:\n", cookiesWanted);
19        System.out.printf("%.2f cups of sugar\n", sugarNeeded);
20        System.out.printf("%.2f cups of butter\n", butterNeeded);
21        System.out.printf("%.2f cups of flour\n", flourNeeded);

23        scanner.close();
24    }
25 }

```

2: IngredientAdjuster.java

## (Ex18 - Book) WordGame.java

**18. Word Game**

Write a program that plays a word game with the user. The program should ask the user to enter the following:

- His or her name
- His or her age
- The name of a city
- The name of a college
- A profession
- A type of animal
- A pet's name

After the user has entered these items, the program should display the following story, inserting the user's input into the appropriate locations:

There once was a person named *NAME* who lived in *CITY*. At the age of *AGE*, *NAME* went to college at *COLLEGE*. *NAME* graduated and went to work as a *PROFESSION*. Then, *NAME* adopted a(n) *ANIMAL* named *PETNAME*. They both lived happily ever after!

Figure 7: WordGame Task Requirement

```

1 import java.util.Scanner;

3 public class WordGame {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);

```

```

7      System.out.print("Enter your name: ");
8      String name = scanner.nextLine();

10     System.out.print("Enter your age: ");
11     int age = scanner.nextInt();
12     scanner.nextLine(); // Consume the newline

14     System.out.print("Enter the name of a city: ");
15     String city = scanner.nextLine();

17     System.out.print("Enter the name of a college: ");
18     String college = scanner.nextLine();

20     System.out.print("Enter a profession: ");
21     String profession = scanner.nextLine();

23     System.out.print("Enter a type of animal: ");
24     String animal = scanner.nextLine();

26     System.out.print("Enter a pet's name: ");
27     String petName = scanner.nextLine();

29     System.out.println("\nHere is your story:");
30     System.out.println("There once was a person named " + name + " who lived
    in " + city + ".");
31     System.out.println("At the age of " + age + ", " + name + " went to
    college at " + college + ".");
32     System.out.println(name + " graduated and went to work as a(n) " +
    profession + ".");
33     System.out.println("Then, " + name + " adopted a(n) " + animal + " named "
    + petName + ".");
34     System.out.println("They both lived happily ever after!");

36     scanner.close();
37 }
38 }

```

3: WordGame.java

## (Ex19 - Book) StockTransaction.java

### 19. Stock Transaction Program

Last month Joe purchased some stock in Acme Software, Inc. Here are the details of the purchase:

- The number of shares that Joe purchased was 1,000.
- When Joe purchased the stock, he paid \$32.87 per share.
- Joe paid his stockbroker a commission that amounted to 2% of the amount he paid for the stock.

Two weeks later Joe sold the stock. Here are the details of the sale:

- The number of shares that Joe sold was 1,000.
- He sold the stock for \$33.92 per share.

- He paid his stockbroker another commission that amounted to 2% of the amount he received for the stock.

Write a program that displays the following information:

- The amount of money Joe paid for the stock.
- The amount of commission Joe paid his broker when he bought the stock.
- The amount that Joe sold the stock for.
- The amount of commission Joe paid his broker when he sold the stock.
- Display the amount of profit that Joe made after selling his stock and paying the two commissions to his broker. (If the amount of profit that your program displays is a negative number, then Joe lost money on the transaction.)

Figure 8: StockTransaction Task Requirement

```
1 public class StockTransaction {
2     public static void main(String[] args) {
3         int sharesBought = 1000;
4         double purchasePricePerShare = 32.87;
5         double commissionRate = 0.02;
6
7         int sharesSold = 1000;
8         double salePricePerShare = 33.92;
9
10        double purchaseAmount = sharesBought * purchasePricePerShare;
11        double purchaseCommission = purchaseAmount * commissionRate;
12
13        double saleAmount = sharesSold * salePricePerShare;
14        double saleCommission = saleAmount * commissionRate;
15
16        double profit = (saleAmount - saleCommission) - (purchaseAmount +
17            purchaseCommission);
18
19        System.out.printf("Amount paid for the stock: $%.2f\n", purchaseAmount);
20        System.out.printf("Commission paid on the purchase: $%.2f\n",
21            purchaseCommission);
22        System.out.printf("Amount the stock sold for: $%.2f\n", saleAmount);
23        System.out.printf("Commission paid on the sale: $%.2f\n", saleCommission);
24        System.out.printf("Profit after commissions: $%.2f\n", profit);
25    }
26 }
```

## 2.3 DisplayIntegers.java

3. Write a program that ask for input 2 integer number and display them on dialogs

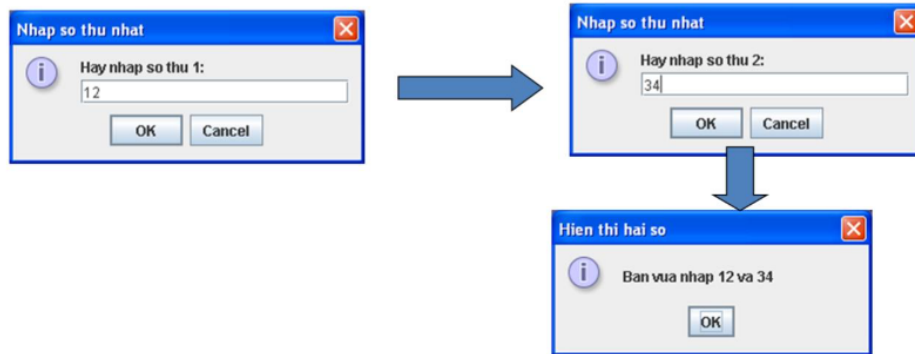


Figure 9: 2.3 Task Requirement

```

1 import javax.swing.JOptionPane;

3 public class DisplayIntegers {
4     public static void main(String[] args) {
5         String firstInput = JOptionPane.showInputDialog("Hay nhap so thu 1:");
6         String secondInput = JOptionPane.showInputDialog("Hay nhap so thu 2:");

8         int firstNumber = Integer.parseInt(firstInput);
9         int secondNumber = Integer.parseInt(secondInput);

11        JOptionPane.showMessageDialog(null, "So thu 1: " + firstNumber);
12        JOptionPane.showMessageDialog(null, "So thu 2: " + secondNumber);
13    }
14 }

```

# Assignment 3

Contributor: Nguyen Duy Khoi - 1677395

## 1. BMI Measure

### 1. BMI measure

Body Mass Index (BMI) is a measure of health based on height and weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. The interpretation of BMI for people 20 years or older is as follow:

- ▶ BMI < 18.5 Underweight
- ▶  $18.5 \leq \text{BMI} < 25.0$  Normal
- ▶  $25.0 \leq \text{BMI} < 30.0$  Overweight
- ▶  $30.0 \leq \text{BMI}$  Obese

Write a program that prompts the user to enter a weight in pounds and height in inches and displays the BMI. Note that one pound is 0.45359237 kilograms and one inch is 0.0254 meters

Figure 10: BMI Measure Task Requirement

```
1 import java.util.Scanner;
2
3 public class BMICalculator {
4     public static void main(String[] args) {
5
6         final double POUNDS_TO_KG = 0.45359237;
7         final double INCHES_TO_METERS = 0.0254;
8         Scanner input = new Scanner(System.in);
9         System.out.print("Enter your weight in pounds: ");
10        double weightPounds = input.nextDouble();
11        System.out.print("Enter your height in inches: ");
12        double heightInches = input.nextDouble();
13        double weightKg = weightPounds * POUNDS_TO_KG;
14        double heightMeters = heightInches * INCHES_TO_METERS;
15        double bmi = weightKg / (heightMeters * heightMeters);
16        System.out.printf("Your BMI is: %.2f\n", bmi);
17
18        if (bmi < 18.5) {
19            System.out.println("Category: Underweight");
20        } else if (bmi < 25.0) {
21            System.out.println("Category: Normal");
22        } else if (bmi < 30.0) {
23            System.out.println("Category: Overweight");
24        } else {
25            System.out.println("Category: Obese");
26        }
27
28        input.close();
29    }
30 }
```

6: BMICalculator.java



## 2. Programming Challenges

### 3.7 SortedNames.java

#### 7. Sorted Names

Write a program that asks the user to enter three names, and then displays the names sorted in ascending order. For example, if the user entered “Charlie”, “Leslie”, and “Andy”, the program would display:

Andy  
Charlie  
Leslie

Figure 11: 3.7 Task Requirement

```
1 import java.util.Arrays;
2 import java.util.Scanner;
3
4 public class SortedNames {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7         System.out.print("Enter the first name: ");
8         String name1 = input.nextLine();
9         System.out.print("Enter the second name: ");
10        String name2 = input.nextLine();
11        System.out.print("Enter the third name: ");
12        String name3 = input.nextLine();
13        String[] names = { name1, name2, name3 };
14        Arrays.sort(names);
15        System.out.println("\nNames in ascending order:");
16        for (String name : names) {
17            System.out.println(name);
18        }
19        input.close();
20    }
21 }
```

7: SortedNames.java

### 3.8 PackageDiscountCalculator.java

**8. Software Sales**

A software company sells a package that retails for \$99. Quantity discounts are given according to the following table:

Quantity	Discount
10–19	20%
20–49	30%
50–99	40%
100 or more	50%

Write a program that asks the user to enter the number of packages purchased. The program should then display the amount of the discount (if any) and the total amount of the purchase after the discount.

Figure 12: 3.8 Task Requirement

```
1 import java.util.Scanner;
2
3 public class PackageDiscountCalculator {
4     public static void main(String[] args) {
5         final double PACKAGE_PRICE = 99.0;
6         double discountRate = 0;
7         Scanner input = new Scanner(System.in);
8         System.out.print("Enter the number of packages purchased: ");
9         int quantity = input.nextInt();
10        if (quantity >= 10 && quantity <= 19) {
11            discountRate = 0.20;
12        } else if (quantity >= 20 && quantity <= 49) {
13            discountRate = 0.30;
14        } else if (quantity >= 50 && quantity <= 99) {
15            discountRate = 0.40;
16        } else if (quantity >= 100) {
17            discountRate = 0.50;
18        }
19        double discountAmount = quantity * PACKAGE_PRICE * discountRate;
20        double totalAmount = (quantity * PACKAGE_PRICE) - discountAmount;
21        System.out.println("Discount amount: $" + discountAmount);
22        System.out.println("Total amount after discount: $" + totalAmount);
23        input.close();
24    }
25 }
```

8: PackageDiscountCalculator.java

### 3.9 ShippingCharges.java

**9. Shipping Charges**

The Fast Freight Shipping Company charges the following rates:

Weight of Package	Rate per 500 Miles Shipped
2 pounds or less	\$1.10
Over 2 pounds but not more than 6 pounds	\$2.20
Over 6 pounds but not more than 10 pounds	\$3.70
Over 10 pounds	\$3.80

The shipping charges per 500 miles are not prorated. For example, if a 2-pound package is shipped 550 miles, the charges would be \$2.20. Write a program that asks the user to enter the weight of a package and then displays the shipping charges.

Figure 13: 3.9 Task Requirement

```
1 import java.util.Scanner;
2
3 public class ShippingCharges {
4     public static void main(String[] args) {
5         final double RATE_2_POUNDS = 1.10;
6         final double RATE_2_TO_6_POUNDS = 2.20;
7         final double RATE_6_TO_10_POUNDS = 3.70;
8         final double RATE_10_POUNDS = 3.80;
9         Scanner input = new Scanner(System.in);
10        System.out.print("Enter the weight of the package (in pounds): ");
11        double weight = input.nextDouble();
12        System.out.print("Enter the distance to be shipped (in miles): ");
13        int distance = input.nextInt();
14        int increments = (int) Math.ceil(distance / 500.0);
15        double rate;
16        if (weight <= 2) {
17            rate = RATE_2_POUNDS;
18        } else if (weight <= 6) {
19            rate = RATE_2_TO_6_POUNDS;
20        } else if (weight <= 10) {
21            rate = RATE_6_TO_10_POUNDS;
22        } else {
23            rate = RATE_10_POUNDS;
24        }
25        double totalCharges = rate * increments;
26        System.out.printf("The shipping charges are: $%.2f\n", totalCharges);
27        input.close();
28    }
29 }
```

9: ShippingCharges.java

# Assignment 4

Contributor: Nguyen Duy Khoi - 1677395

## 1. Mirror Drawing

### Assignment for Lecture 4

#### 1. Mirror Drawing

a) Write a program running in Console, that asks user to input the width and the height of the Mirror. They are even numbers.

Draw the mirror.

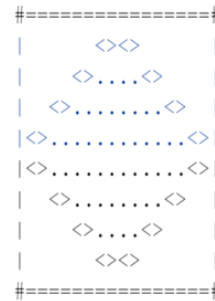


Figure 14: Mirror Drawing Task Requirement

```
1 import java.util.Scanner;
2
3 public class MirrorDrawing {
4
5     public static void drawMirror(int width, int height) {
6         if (width % 2 != 0 || height % 2 != 0) {
7             System.out.println("Please enter even numbers for width and height.");
8             return;
9         }
10        System.out.print("#");
11        for (int i = 0; i < width - 2; i++) {
12            System.out.print("=");
13        }
14        System.out.println("#");
15        int halfHeight = (height - 2) / 2;
16        for (int i = 0; i < halfHeight; i++) {
17            System.out.print("|");
18            for (int j = 0; j < halfHeight - i - 1; j++) {
19                System.out.print(" ");
20            }
21            System.out.print("<>");
22            for (int j = 0; j < i * 2 + 1; j++) {
23                System.out.print(".");
24            }
25            System.out.print(">");
26            for (int j = 0; j < halfHeight - i - 1; j++) {
27                System.out.print(" ");
28            }
29            System.out.println("|");
30        }
31        for (int i = halfHeight - 1; i >= 0; i--) {
32            System.out.print("|");
33            for (int j = 0; j < halfHeight - i - 1; j++) {
34                System.out.print(" ");
35            }
36            System.out.print("<>");
```

```

37         for (int j = 0; j < i * 2 + 1; j++) {
38             System.out.print(".");
39         }
40         System.out.print("◇");
41         for (int j = 0; j < halfHeight - i - 1; j++) {
42             System.out.print(" ");
43         }
44         System.out.println("|");
45     }
46     System.out.print("#");
47     for (int i = 0; i < width - 2; i++) {
48         System.out.print("=");
49     }
50     System.out.println("#");
51 }

53 public static void main(String[] args) {
54     Scanner scanner = new Scanner(System.in);
55     System.out.print("Enter the width of the mirror (even number): ");
56     int width = scanner.nextInt();
57     System.out.print("Enter the height of the mirror (even number): ");
58     int height = scanner.nextInt();
59     drawMirror(width, height);
60     scanner.close();
61 }
62 }

```

10: MirrorDrawing.java

## Runtime Result

## 2. Programming Challenges

### 4.18 Random Number Guessing Game Enhancement

#### 17. Random Number Guessing Game

Write a program that generates a random number and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." The program should use a loop that repeats until the user correctly guesses the random number.

#### 18. Random Number Guessing Game Enhancement

Enhance the program that you wrote for Programming Challenge 17 so it keeps a count of the number of guesses that the user makes. When the user correctly guesses the random number, the program should display the number of guesses.

Figure 15: 4.18 Task Requirement

```

1 import java.util.Random;
2 import java.util.Scanner;

4 public class RandomNumberGuessingGameEnhanced {

6     public static void main(String[] args) {
7         Random random = new Random();

```

```

8      int randomNumber = random.nextInt(100) + 1;
9      Scanner scanner = new Scanner(System.in);
10     int guess = 0;
11     int guessCount = 0;
12     System.out.println("Guess the random number between 1 and 100.");
13     while (guess != randomNumber) {
14         System.out.print("Enter your guess: ");
15         guess = scanner.nextInt();
16         guessCount++;
17         if (guess > randomNumber) {
18             System.out.println("Too high, try again.");
19         } else if (guess < randomNumber) {
20             System.out.println("Too low, try again.");
21         } else {
22             System.out.println("Congratulations! You guessed the correct
23                                 number.");
24             System.out.println("Number of guesses: " + guessCount);
25         }
26     }
27     scanner.close();
28 }
29 }

```

11: RandomNumberGuessingGameEnhanced.java

#### 4.19 ESP Game

##### 19. ESP Game

Write a program that tests your ESP (extrasensory perception). The program should randomly select the name of a color from the following list of words:

*Red, Green, Blue, Orange, Yellow*

To select a word, the program can generate a random number. For example, if the number is 0, the selected word is *Red*, if the number is 1, the selected word is *Green*, and so forth.

Next, the program should ask the user to enter the color that the computer has selected. After the user has entered his or her guess, the program should display the name of the randomly selected color. The program should repeat this 10 times and then display the number of times the user correctly guessed the selected color.

Figure 16: 4.19 Task Requirement

```

1  import java.util.Random;
2  import java.util.Scanner;
3
4  public class ESPGame {
5
6      public static void main(String[] args) {
7          String[] colors = {"Red", "Green", "Blue", "Orange", "Yellow"};
8          Random random = new Random();
9          Scanner scanner = new Scanner(System.in);
10         int correctGuesses = 0; // Counter for correct guesses
11         System.out.println("Welcome to the ESP Game!");
12         System.out.println("Try to guess the color that the computer has selected.
13                             ");
14         System.out.println("Available colors: Red, Green, Blue, Orange, Yellow");
15     }
16 }

```

```

14     for (int i = 0; i < 10; i++) {
15         int colorIndex = random.nextInt(colors.length);
16         String selectedColor = colors[colorIndex];
17         System.out.print("Enter your guess: ");
18         String userGuess = scanner.nextLine().trim();
19         if (userGuess.equalsIgnoreCase(selectedColor)) {
20             System.out.println("Correct!");
21             correctGuesses++;
22         } else {
23             System.out.println("Incorrect. The correct color was: " +
24                 selectedColor);
25         }
26     }
27     System.out.println("You guessed the correct color " + correctGuesses + "
28         times out of 10.");
29     scanner.close();
30 }

```

12: ESPGame.java

#### 4.21 Dice Game

##### 21. Dice Game

Write a program that plays a simple dice game between the computer and the user. When the program runs, a loop should repeat 10 times. Each iteration of the loop should do the following:

- Generate a random integer in the range of 1 through 6. This is the value of the computer's die.
- Generate another random integer in the range of 1 through 6. This is the value of the user's die.
- The die with the highest value wins. (In case of a tie, there is no winner for that particular roll of the dice.)

As the loop iterates, the program should keep count of the number of times the computer wins, and the number of times that the user wins. After the loop performs all of its iterations, the program should display who was the grand winner, the computer or the user.

Figure 17: 4.21 Task Requirement

```

1  import java.util.Random;
2
3  public class DiceGame {
4
5      public static void main(String[] args) {
6          Random random = new Random();
7          int computerWins = 0;
8          int userWins = 0;
9          System.out.println("Welcome to the Dice Game!");
10         System.out.println("The game will be played for 10 rounds.");
11         for (int i = 0; i < 10; i++) {
12             int computerRoll = random.nextInt(6) + 1;
13             int userRoll = random.nextInt(6) + 1;
14             System.out.println("Round " + (i + 1) + ":");
15             System.out.println("Computer rolled: " + computerRoll);

```

```

16         System.out.println("User rolled: " + userRoll);
17         if (computerRoll > userRoll) {
18             System.out.println("Computer wins this round!");
19             computerWins++;
20         } else if (userRoll > computerRoll) {
21             System.out.println("User wins this round!");
22             userWins++;
23         } else {
24             System.out.println("It's a tie for this round!");
25         }
26         System.out.println();
27     }
28     System.out.println("Game Over!");
29     System.out.println("Computer won " + computerWins + " rounds.");
30     System.out.println("User won " + userWins + " rounds.");
31     if (computerWins > userWins) {
32         System.out.println("The grand winner is the Computer!");
33     } else if (userWins > computerWins) {
34         System.out.println("The grand winner is the User!");
35     } else {
36         System.out.println("The game is a tie!");
37     }
38 }
39 }

```

13: DiceGame.java

### 3. Simple Calculator

3. Write a program that asks the user to enter two numerical values (integers) and then select an operation (addition, subtraction, multiplication and division) then prints the result based on operation selected. The code below shows examples of the output (text shown in boldface is supposed to be user input).

```

Enter first number: 4
Enter second number: 2

1. Addition (+).
2. Subtraction (-).
3. Multiplication (*).
4. Division (/).

Enter operation number: 3

The result is 8

```

Figure 18: Simple Calculator Task Requirement

```

1 import java.util.Scanner;
2
3 public class SimpleCalculator {
4
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         System.out.print("Enter first number: ");
8     }
9 }

```



```

8      int num1 = scanner.nextInt();
9      System.out.print("Enter second number: ");
10     int num2 = scanner.nextInt();
11     System.out.println("1. Addition (+)");
12     System.out.println("2. Subtraction (-)");
13     System.out.println("3. Multiplication (*)");
14     System.out.println("4. Division (/)");
15     System.out.print("Enter operation number: ");
16     int choice = scanner.nextInt();
17     int result = 0;
18     switch (choice) {
19         case 1:
20             result = num1 + num2;
21             break;
22         case 2:
23             result = num1 - num2;
24             break;
25         case 3:
26             result = num1 * num2;
27             break;
28         case 4:
29             if (num2 != 0) {
30                 result = num1 / num2;
31             } else {
32                 System.out.println("Error: Division by zero is not allowed.");
33                 scanner.close();
34                 return;
35             }
36             break;
37         default:
38             System.out.println("Invalid operation number.");
39             scanner.close();
40             return;
41     }
42     System.out.println("The result is " + result);
44     scanner.close();
45 }
46 }

```

14: SimpleCalculator.java

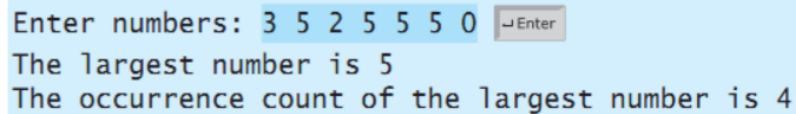
# Assignment 5

Contributor: Nguyen Duy Duc - 1624838

## 1. Occurrence of max numbers

### 1. (Occurrence of max numbers)

Write a program that reads integers, finds the largest of them, and counts its occurrences. Assume that the input ends with number 0. Suppose that you entered 3 5 2 5 5 5 0; the program finds that the largest is 5 and the occurrence count for 5 is 4.



Enter numbers: 3 5 2 5 5 5 0

The largest number is 5

The occurrence count of the largest number is 4

Figure 19: Problem 1 Task Requirement

```
1 package OccurenceOfMaxNum;
2
3 import java.util.Scanner;
4
5 public class MaxNumCounter {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8         System.out.println("(The program will read integers input til the first 0)");
9         System.out.println("Enter numbers:");
10        findLargestAndCount(scanner);
11        scanner.close();
12    }
13
14    public static void findLargestAndCount(Scanner scanner) {
15        int max = 0;
16        int count = 1;
17
18        while (true) {
19            if (scanner.hasNextInt()) {
20                int number = scanner.nextInt();
21                if (number == 0) {
22                    break;
23                }
24                if (number > max) {
25                    max = number;
26                    count = 1;
27                } else if (number == max) {
28                    count++;
29                }
30            } else {
31                System.out.println("Invalid input (not integers) ignored.");
32                scanner.next(); // Clear the invalid input
33            }
34        }
35
36        if (max != 0) {
```

```

37         System.out.println("The largest number is " + max);
38         System.out.println("The occurrence count of the largest number is " +
           count);
39     } else {
40         System.out.println("The largest number is 0");
41         System.out.println("The occurrence count of the largest number is 1");
42     }
43 }
44 }

```

15: MaxNumCounter.java

## Runtime Result

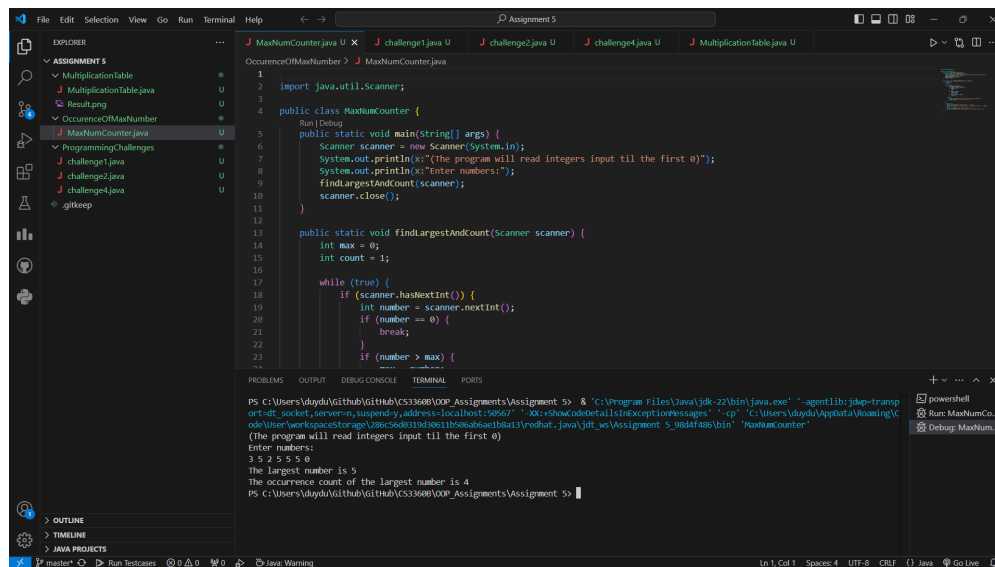


Figure 20: Runtime Result

## 2. Programming Challenges

### Challenge 1

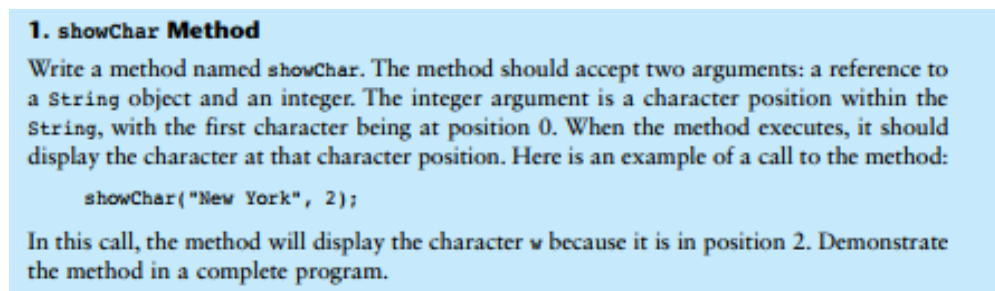


Figure 21: 5.1 Task Requirement

```

1 package ProgrammingChallenges;
2 import java.util.Scanner;

```

```

4 public class challenge1 {
5     public static void main(String[] args) {
6
7         Scanner scanner = new Scanner(System.in);
8
9         //Input string
10        System.out.print("Enter a string: ");
11        String inputString = scanner.nextLine();
12
13        //Input position
14        int position;
15        while (true) {
16            System.out.print("Enter a position: ");
17            position = scanner.nextInt();
18
19            // Check if the position is valid
20            if (position >= 0 && position < inputString.length()) {
21                break;
22            } else {
23                System.out.println("Invalid position. Please enter a position
24                                between 0 and " + (inputString.length() - 1));
25            }
26        }
27
28        // Call the showChar method with the user inputs
29        showChar(inputString, position);
30
31        scanner.close();
32    }
33
34    public static void showChar(String str, int position) {
35        if (position >= 0 && position < str.length()) {
36            char ch = str.charAt(position);
37            System.out.println("The character at position " + position + " is " +
38                               ch);
39        } else {
40            System.out.println("Invalid position. Please enter a position between
41                               0 and " + (str.length() - 1));
42        }
43    }
44 }

```

16: Challenge1.java

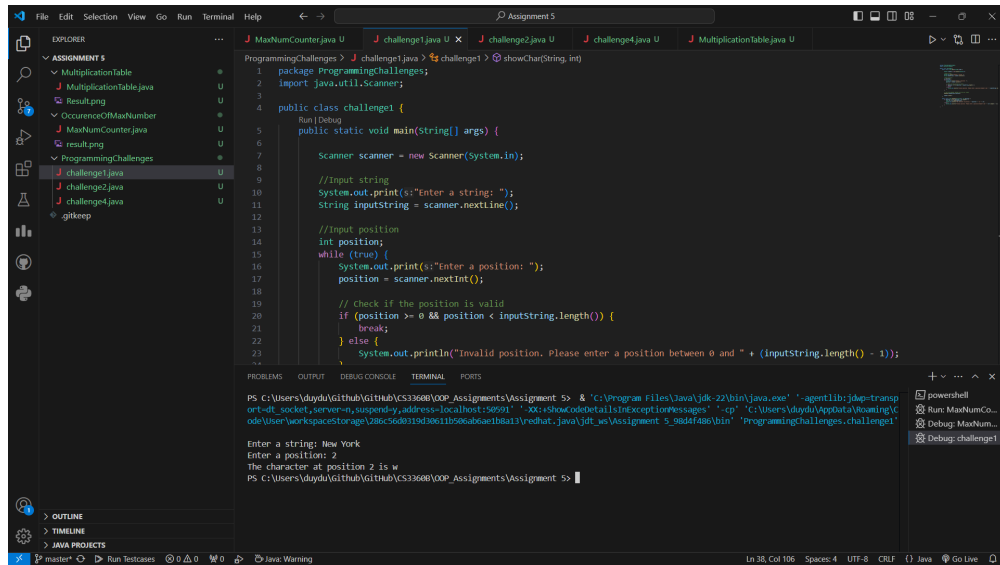


Figure 22: Runtime Result

## Challenge 2

**2. Retail Price Calculator**

Write a program that asks the user to enter an item's wholesale cost and its markup percentage. It should then display the item's retail price. For example:

- If an item's wholesale cost is 5.00 and its markup percentage is 100 percent, then the item's retail price is 10.00.
- If an item's wholesale cost is 5.00 and its markup percentage is 50 percent, then the item's retail price is 7.50.

The program should have a method named `calculateRetail` that receives the wholesale cost and the markup percentage as arguments, and returns the retail price of the item.

Figure 23: 5.2 Task Requirement

```

1 package ProgrammingChallenges;
2
3 import java.util.Scanner;
4
5 public class challenge2 {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         // Input the wholesale cost
10        System.out.print("Enter the item's wholesale cost: ");
11        double wholesaleCost = scanner.nextDouble();
12
13        //Input the markup percentage
14        System.out.print("Enter the item's markup percentage: ");
15        double markupPercentage = scanner.nextDouble();
16
17        // Calculate the retail price
18        double retailPrice = calculateRetail(wholesaleCost, markupPercentage);

```

```

20 // Display the retail price
21 System.out.printf("The item's retail price is: %.2f%n", retailPrice);

23 scanner.close();
24 }

26 public static double calculateRetail(double wholesaleCost, double
    markupPercentage) {
27     return wholesaleCost + (wholesaleCost * markupPercentage / 100);
28 }
29 }

```

17: Challenge2.java

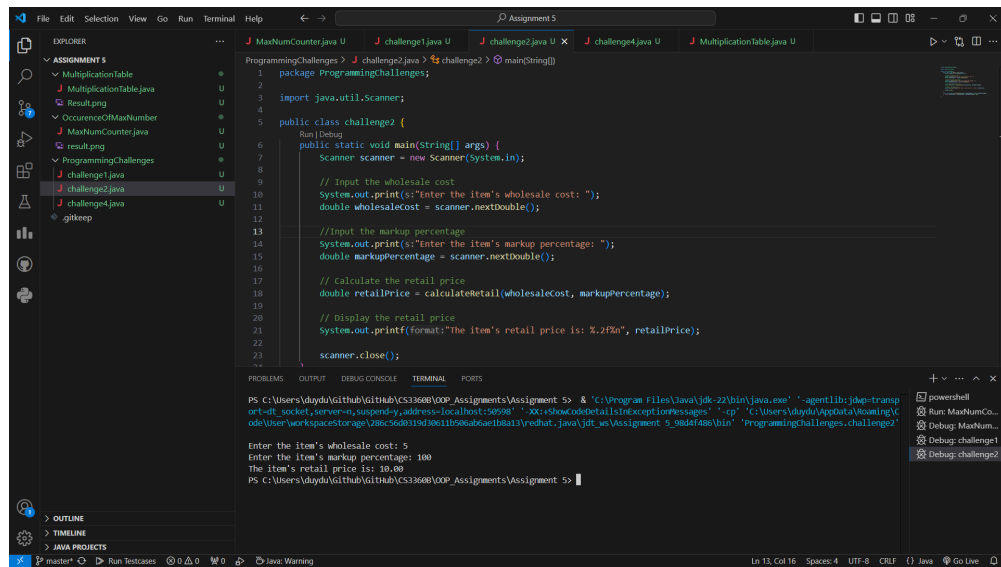


Figure 24: Runtime Result

## Challenge 4

### 4. Paint Job Estimator

A painting company has determined that for every 115 square feet of wall space, one gallon of paint and eight hours of labor will be required. The company charges \$18.00 per hour for labor. Write a program that allows the user to enter the number of rooms to be painted and the price of the paint per gallon. It should also ask for the square feet of wall space in each room. The program should have methods that return the following data:

- The number of gallons of paint required
- The hours of labor required

- The cost of the paint
- The labor charges
- The total cost of the paint job

Then it should display the data on the screen.

Figure 25: 5.4 Task Requirement

```
1 package ProgrammingChallenges;
2
3 import java.util.Scanner;
4
5 public class challenge4 {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         // Input the number of rooms
10        int numberOfRooms = getValidIntInput(scanner, "Enter the number of rooms
11        to be painted: ");
12
13        // Input the price of the paint per gallon
14        double pricePerGallon = getValidDoubleInput(scanner, "Enter the price of
15        the paint per gallon: ");
16
17        // Initialize total square feet
18        double totalSquareFeet = 0;
19
20        // Input the square feet of wall space for each room
21        for (int i = 1; i <= numberOfRooms; i++) {
22            totalSquareFeet += getValidDoubleInput(scanner, "Enter the square feet
23            of wall space for room " + i + ": ");
24        }
25
26        // Calculate the required data
27        double gallonsOfPaintRequired = calculateGallonsOfPaintRequired(
28            totalSquareFeet);
29        double hoursOfLaborRequired = calculateHoursOfLaborRequired(
30            totalSquareFeet);
31        double costOfPaint = calculateCostOfPaint(gallonsOfPaintRequired,
32            pricePerGallon);
```

```

27     double laborCharges = calculateLaborCharges(hoursOfLaborRequired);
28     double totalCost = calculateTotalCost(costOfPaint, laborCharges);

30     // Display the results
31     System.out.printf("The numbers of gallons of paint required: %.2f\n",
32                       gallonsOfPaintRequired);
33     System.out.printf("The hours of labor required: %.2f\n",
34                       hoursOfLaborRequired);
35     System.out.printf("The cost of the paint: $%.2f\n", costOfPaint);
36     System.out.printf("The labor charges: $%.2f\n", laborCharges);
37     System.out.printf("The total cost of the paint job: $%.2f\n", totalCost);

38     scanner.close();
39 }

40 public static int getValidIntInput(Scanner scanner, String prompt) {
41     int input;
42     while (true) {
43         System.out.print(prompt);
44         if (scanner.hasNextInt()) {
45             input = scanner.nextInt();
46             scanner.nextLine(); // Consume the newline character
47             break;
48         } else {
49             System.out.println("Invalid input. Please enter a valid integer.");
50             scanner.next(); // Clear the invalid input
51         }
52     }
53     return input;
54 }

56 public static double getValidDoubleInput(Scanner scanner, String prompt) {
57     double input;
58     while (true) {
59         System.out.print(prompt);
60         if (scanner.hasNextDouble()) {
61             input = scanner.nextDouble();
62             scanner.nextLine(); // Consume the newline character
63             break;
64         } else {
65             System.out.println("Invalid input. Please enter a valid number.");
66             scanner.next(); // Clear the invalid input
67         }
68     }
69     return input;
70 }

72 public static double calculateGallonsOfPaintRequired(double totalSquareFeet) {
73     return totalSquareFeet / 115;
74 }

76 public static double calculateHoursOfLaborRequired(double totalSquareFeet) {
77     return (totalSquareFeet / 115) * 8;
78 }

80 public static double calculateCostOfPaint(double gallonsOfPaintRequired,
81     double pricePerGallon) {
    return gallonsOfPaintRequired * pricePerGallon;

```



```

82     }

84     public static double calculateLaborCharges(double hoursOfLaborRequired) {
85         return hoursOfLaborRequired * 18.00;
86     }

88     public static double calculateTotalCost(double costOfPaint, double
        laborCharges) {
89         return costOfPaint + laborCharges;
90     }
91 }

```

18: Challenge4.java

```

import java.util.Scanner;

public class challenge4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input the number of rooms
        int numberOfRooms = getValidIntInput(scanner, prompt:"Enter the number of rooms to be painted: ");

        // Input the price of the paint per gallon
        double pricePerGallon = getValidDoubleInput(scanner, prompt:"Enter the price of the paint per gallon: ");

        // Initialize total square feet
        double totalSquarefeet = 0;

        // Input the square feet of wall space for each room
        for (int i = 1; i <= numberOfRooms; i++) {
            totalSquarefeet += getValidDoubleInput(scanner, "Enter the square feet of wall space for room " + i + ": ");
        }

        // calculate the required data
        double gallonsOfPaintRequired = calculateGallonsOfPaintRequired(totalSquarefeet);
        double hoursOfLaborRequired = calculateHoursOfLaborRequired(totalSquarefeet);

        Enter the number of rooms to be painted: 2
        Enter the price of the paint per gallon: 40
        Enter the square feet of wall space for room 1: 100
        Enter the square feet of wall space for room 2: 100
        The numbers of gallons of paint required: 1.74
        The hours of labor required: 13.91
        The cost of the paint: $17.39
        The labor charges: $250.43
        The total cost of the paint job: $267.83
    }
}

```

Figure 26: Runtime Result

### 3. Multiplication Table

3. Write a program that uses nested for loops to print a multiplication table inside a dialog as follows:

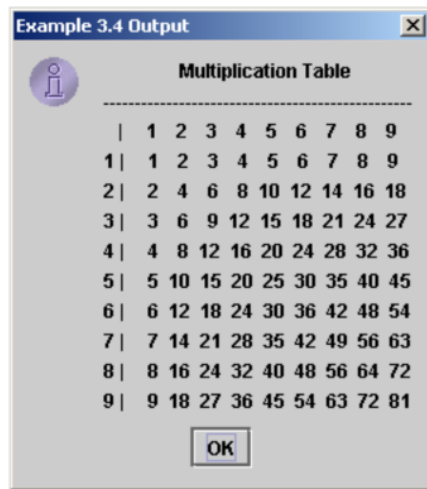


Figure 27: Problem 3 Task Requirement

```
1 package MultiplicationTable;
2
3 import javax.swing.JOptionPane;
4
5 public class MultiplicationTable {
6     public static void main(String[] args) {
7         StringBuilder table = new StringBuilder();
8
9         // Add the header and underline
10        table.append("Multiplication Table\n");
11        table.append("-----\n");
12
13        // Add the top row of numbers
14        table.append("    |");
15        for (int i = 1; i <= 9; i++) {
16            table.append(String.format("%6d", i));
17        }
18        table.append("\n");
19
20        // Create the multiplication table using nested for loops
21        for (int i = 1; i <= 9; i++) {
22            table.append(String.format("%2d |", i));
23            for (int j = 1; j <= 9; j++) {
24                if (i*j >= 10){
25                    table.append(String.format("%5d", i * j));
26                } else table.append(String.format("%6d", i * j));
27            }
28            table.append("\n");
29        }
30
31        // Display the multiplication table in a dialog box with the specified
        // header
```

```

32 | JOptionPane.showMessageDialog(null, table.toString(), "Example 3.4 Output"
    | , JOptionPane.INFORMATION_MESSAGE);
33 | }
34 | }

```

19: MultiplicationTable.java

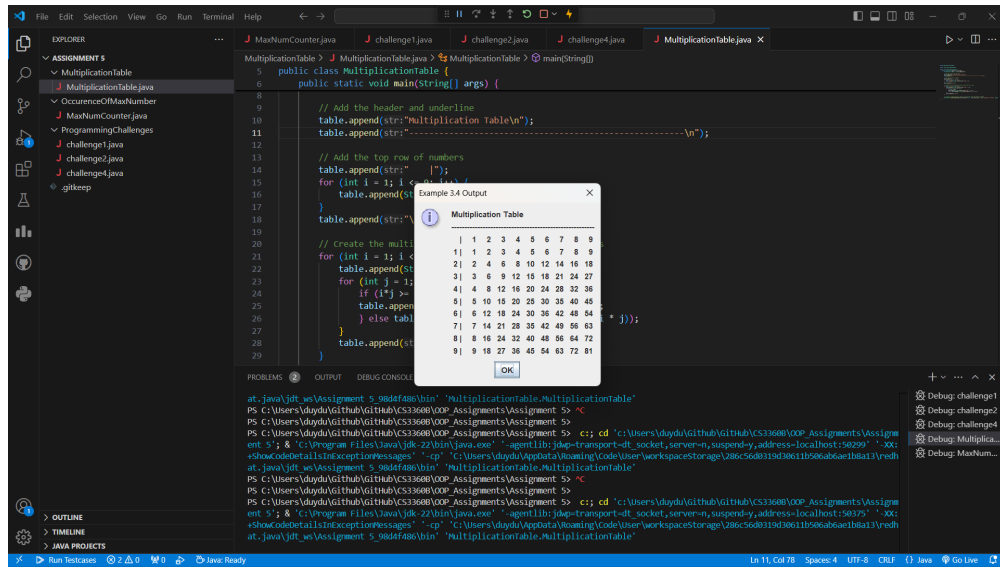


Figure 28: Runtime Result

# Assignment 6

Contributor: Ngo Thanh Trung - 1677469

## 1. Implementation of Student Class

### 1. Implementation of Student Class

a) Write a class Student which models students with their:

ID: string, name, accumulated CPA, prefer courses (an array of String).

b) Write a program that create and display about the following students:

(TROY1231, Bob Walker, 3.21, {"Math", "Philosophy", "Art"})

(TROY1217, Bao Trung, 2.67, {"Computer Science I", "Math"})

(TROY1362, Nina Gigi, 2.99, {"Sing", "Dance", "Physics"})

the find the course that is most prefer by Student.

Don't use constructor, private methods in this exercises.

Figure 29: Student Class Task Requirement

```
1 import java.util.Arrays;
2
3 public class Student {
4     private String id;
5     private String name;
6     private double accumulatedCPA;
7     private String[] preferCourse;
8
9     public String getId() {
10         return id;
11     }
12
13     public void setId(String id) {
14         this.id = id;
15     }
16
17     public String getName() {
18         return name;
19     }
20
21     public void setName(String name) {
22         this.name = name;
23     }
24
25     public double getAccumulatedCPA() {
26         return accumulatedCPA;
27     }
28 }
```

```

29     public void setAccumulatedCPA(double accumulatedCPA) {
30         this.accumulatedCPA = accumulatedCPA;
31     }

32
33     public String[] getPreferCourse() {
34         return preferCourse;
35     }

36
37     public void setPreferCourse(String[] preferCourse) {
38         this.preferCourse = preferCourse;
39     }

40
41     public String studentGetInfo(){
42         return this.getId() + " | Student name: " + this.getName() + " |
           Accumulated CPA: " + this.getAccumulatedCPA() + " | Student prefer
           course: " + Arrays.toString(this.getPreferCourse());
43     }
44 }

```

20: Student.java

```

1 public class Main {
2     public static void main(String[] args) {
3         Student Bob = new Student();
4         Bob.setId("TROY1231");
5         Bob.setName("Bob Walker");
6         Bob.setAccumulatedCPA(3.21);
7         Bob.setPreferCourse(new String[] {"Math", "Philosophy", "Art"});

8
9         Student Trung = new Student();
10        Trung.setId("TROY1217");
11        Trung.setName("Bao Trung");
12        Trung.setAccumulatedCPA(2.67);
13        Trung.setPreferCourse(new String[] {"Computer Science I", "Math"});

14
15        Student Nina = new Student();
16        Nina.setId("TROY1362");
17        Nina.setName("Nina Gigi");
18        Nina.setAccumulatedCPA(2.99);
19        Nina.setPreferCourse(new String[] {"Sing", "Dance", "Physics"});

20
21        System.out.println(Bob.studentGetInfo());
22        System.out.println(Trung.studentGetInfo());
23        System.out.println(Nina.studentGetInfo());
24    }
25 }

```

21: Main.java

```

Run Main x
/Library/Exit javaVirtualMachines/jdk-22.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt
.jar=59443:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath /Users/kiyo9w/IdeaProjects/Assignment 6/Assignment 6/1. Implementation of Student Class/out/production/1. Implementation of
Student Class Main
TRO1231 | Student name: Bob Walker | Accumulated CPA: 3.21 | Student prefer course: [Math, Philosophy, Art]
TRO1217 | Student name: Bao Trung | Accumulated CPA: 2.67 | Student prefer course: [Computer Science I, Math]
TRO1362 | Student name: Nina Gigi | Accumulated CPA: 2.99 | Student prefer course: [Sing, Dance, Physics]

Process finished with exit code 0

```

Figure 30: Runtime Result

## 2. Implementation of TV Class

### 2. Implementation of TV Class

Consider television sets. Each TV is an object with states (current channel, current volume level, power on or off) and behaviors (change channels, adjust volume, turn on/off). Write a class to model TV sets and demo their usages in a program.

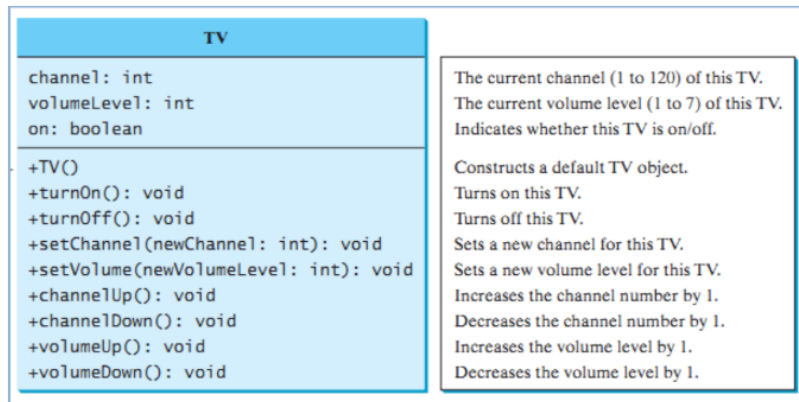


Figure 31: TV Class Task Requirement

```

1 public class TV {
2     private int channel;
3     private int volumeLevel;
4     private boolean on;

5
6     private static final int MAX_CHANNEL = 1000;
7     private static final int MIN_CHANNEL = 0;
8     private static final int MAX_VOLUME = 100;
9     private static final int MIN_VOLUME = 0;

10
11     TV() {
12         this.channel = MIN_CHANNEL;
13         this.volumeLevel = 50;
14         this.on = false;
15     }

16
17     public void turnOn() {
18         this.on = true;
19         System.out.println("TV turned on");
20     }

```

```

22     public void turnOff() {
23         System.out.println("TV turning off");
24         this.on = false;
25     }

27     private void currentVolume() {
28         System.out.println("Current volume: " + this.volumeLevel);
29     }

31     private void currentChannel() {
32         System.out.println("Channel: " + this.channel);
33     }

35     public void setChannel(int newChannel) {
36         if(this.on && newChannel >= MIN_CHANNEL && newChannel <= MAX_CHANNEL) {
37             this.channel = newChannel;
38             currentChannel();
39         }
40     }

42     public void setVolume(int newVolumeLevel) {
43         if(this.on && newVolumeLevel >= MIN_VOLUME && newVolumeLevel <= MAX_VOLUME
44             && newVolumeLevel % 5 == 0) {
45             this.volumeLevel = newVolumeLevel;
46             currentVolume();
47         }
48     }

49     public void channelUp() {
50         if(this.on && this.channel + 1 <= MAX_CHANNEL) {
51             this.channel++;
52             currentChannel();
53         }
54     }

56     public void channelDown() {
57         if(this.on && this.channel - 1 >= MIN_CHANNEL) {
58             this.channel--;
59             currentChannel();
60         }
61     }

63     public void volumeUp() {
64         if(this.on && volumeLevel + 5 <= MAX_VOLUME) {
65             this.volumeLevel += 5;
66             currentVolume();
67         }
68     }

70     public void volumeDown() {
71         if (this.on && volumeLevel - 5 >= MIN_VOLUME) {
72             this.volumeLevel -= 5;
73             currentVolume();
74         }
75     }
76 }

```

22: TV.java

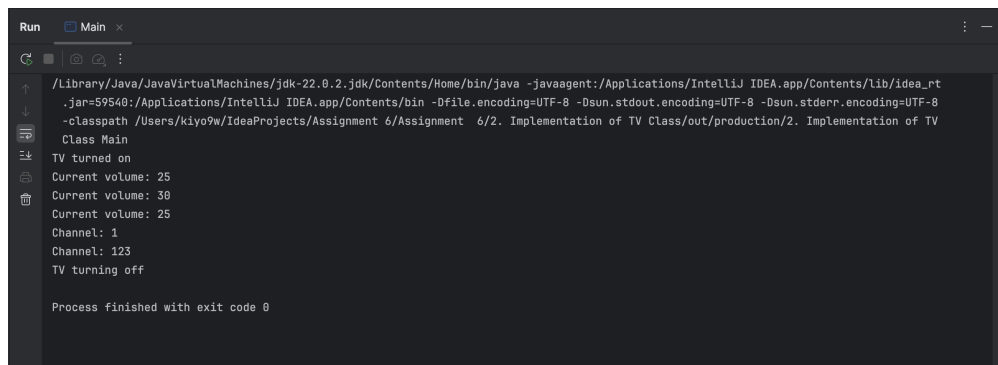
```

2 public class Main {
3     public static void main(String[] args) {
4         TV SamsungTV = new TV();
5         SamsungTV.setChannel(45);
6         SamsungTV.setVolume(25);
7         SamsungTV.volumeUp();

9         SamsungTV.turnOn();
10        SamsungTV.setVolume(25);
11        SamsungTV.volumeUp();
12        SamsungTV.volumeDown();
13        SamsungTV.channelDown();
14        SamsungTV.channelUp();
15        SamsungTV.setChannel(123);
16        SamsungTV.setChannel(1230);
17        SamsungTV.turnOff();
18    }
19 }

```

23: Main.java



```

Run Main x
/Library/Java/JavaVirtualMachines/jdk-22.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt
.jar=59540:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath /Users/kiyo9w/IdeaProjects/Assignment 6/Assignment 6/2. Implementation of TV Class/out/production/2. Implementation of TV
Class Main
TV turned on
Current volume: 25
Current volume: 30
Current volume: 25
Channel: 1
Channel: 123
TV turning off

Process finished with exit code 0

```

Figure 32: Runtime Result



## 6.1 Employee Class

### 1. Employee Class

Write a class named `Employee` that has the following fields:

- `name`. The `name` field references a `String` object that holds the employee's name.
- `idNumber`. The `idNumber` is an `int` variable that holds the employee's ID number.
- `department`. The `department` field references a `String` object that holds the name of the department where the employee works.
- `position`. The `position` field references a `String` object that holds the employee's job title.

The class should have the following constructors:

- A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name, employee's ID number, department, and position.
- A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name and ID number. The `department` and `position` fields should be assigned an empty string (`""`).
- A no-arg constructor that assigns empty strings (`""`) to the `name`, `department`, and `position` fields, and 0 to the `idNumber` field.

Write appropriate mutator methods that store values in these fields and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three `Employee` objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.

Figure 33: Employee Class Task Requirement

```
1 package employeeClass;
2
3 public class Employee {
4     private String name;
5     private int idNumber;
6     private String department;
7     private String position;
8
9     public Employee(String name, int idNumber, String department, String position)
10    {
11        this.name = name;
12        this.idNumber = idNumber;
13        this.department = department;
14        this.position = position;
15    }
16
17    public Employee(String name, int idNumber) {
18        this(name, idNumber, "", "");
19    }
20
21    public Employee() {
```

```

21         this("", 0, "", "");
22     }

24     public void setName(String name) {
25         this.name = name;
26     }

28     public void setIdNumber(int idNumber) {
29         this.idNumber = idNumber;
30     }

32     public void setDepartment(String department) {
33         this.department = department;
34     }

36     public void setPosition(String position) {
37         this.position = position;
38     }

40     public String getName() {
41         return name;
42     }

44     public int getIdNumber() {
45         return idNumber;
46     }

48     public String getDepartment() {
49         return department;
50     }

52     public String getPosition() {
53         return position;
54     }
55 }

```

24: Employee.java

```

1 package employeeClass;

4 public class Main {
5     public static void main(String[] args) {
6         Employee employee1 = new Employee("Susan Meyers", 47899, "
           Accounting", "Vice President");
7         Employee employee2 = new Employee("Mark Jones", 39119, "IT", "
           Programmer");
8         Employee employee3 = new Employee("Joy Rogers", 81774, "
           Manufacturing", "Engineer");

10         displayEmployeeInfo(employee1);
11         displayEmployeeInfo(employee2);
12         displayEmployeeInfo(employee3);
13     }

15     public static void displayEmployeeInfo(Employee employee) {
16         System.out.printf(" Name: " + employee.getName());
17         System.out.printf(" | ID Number: " + employee.getIdNumber());
18         System.out.printf(" | Department: " + employee.getDepartment());

```

```

19         System.out.printf(" | Position: " + employee.getPosition());
20         System.out.println();
21     }
22 }

```

25: Main.java

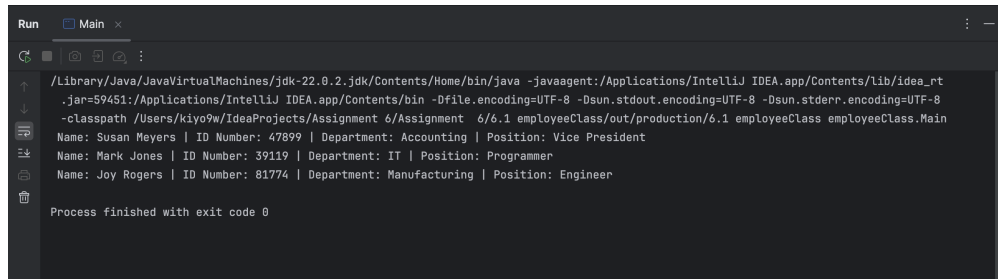


Figure 34: Runtime Result

## 6.2 Car Class

**2. Car Class**

Write a class named `Car` that has the following fields:

- **yearModel**. The `yearModel` field is an `int` that holds the car's year model.
- **make**. The `make` field references a `String` object that holds the make of the car.
- **speed**. The `speed` field is an `int` that holds the car's current speed.

In addition, the class should have the following constructor and other methods.

- **Constructor**. The constructor should accept the car's year model and make as arguments. These values should be assigned to the object's `yearModel` and `make` fields. The constructor should also assign 0 to the `speed` field.
- **Accessors**. Appropriate accessor methods should get the values stored in an object's `yearModel`, `make`, and `speed` fields.
- **accelerate**. The `accelerate` method should add 5 to the `speed` field each time it is called.
- **brake**. The `brake` method should subtract 5 from the `speed` field each time it is called.

Demonstrate the class in a program that creates a `Car` object, and then calls the `accelerate` method five times. After each call to the `accelerate` method, get the current speed of the car and display it. Then call the `brake` method five times. After each call to the `brake` method, get the current speed of the car and display it.

Figure 35: Car Class Task Requirement

```

1 package carClass;

3 public class Car {
4     private int yearModel;
5     private String make;
6     private int speed;

8     public Car(int yearModel, String make) {
9         this.yearModel = yearModel;

```

```

10         this.make = make;
11         this.speed = 0;
12     }

14     public int getYearModel() {
15         return yearModel;
16     }

18     public String getMake() {
19         return make;
20     }

22     public int getSpeed() {
23         return speed;
24     }

26     public void accelerate() {
27         speed += 10;
28     }

30     public void brake() {
31         if (speed >= 10) {
32             speed -= 10;
33         } else {
34             speed = 0;
35         }
36     }
37 }

```

26: Car.java

```

1 package carClass;

3 public class Main {
4     public static void main(String[] args) {
5         // Create a Car object
6         Car car = new Car(2023, "Toyota");

8         // Accelerate the car five times and display the speed after each
           acceleration
9         System.out.println("Accelerating...");
10        for (int i = 1; i <= 5; i++) {
11            car.accelerate();
12            System.out.println("Current speed after acceleration " + i + ": " +
               car.getSpeed() + " mph");
13        }

15        // Brake the car five times and display the speed after each brake
16        System.out.println("\nBraking...");
17        for (int i = 1; i <= 5; i++) {
18            car.brake();
19            System.out.println("Current speed after brake " + i + ": " + car.
               getSpeed() + " mph");
20        }
21    }
22 }

```

27: Main.java

```
Run Main x
/Library/Java/JavaVirtualMachines/jdk-22.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt
.jar=59344:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath /Users/Kiyo9w/IdeaProjects/Assignment 6/6.2 carClass/out/production/6.2 carClass carClass.Main
Accelerating...
Current speed after acceleration 1: 10 mph
Current speed after acceleration 2: 20 mph
Current speed after acceleration 3: 30 mph
Current speed after acceleration 4: 40 mph
Current speed after acceleration 5: 50 mph

Braking...
Current speed after brake 1: 40 mph
Current speed after brake 2: 30 mph
Current speed after brake 3: 20 mph
Current speed after brake 4: 10 mph
Current speed after brake 5: 0 mph

Process finished with exit code 0
```

Figure 36: Runtime Result

## 6.3 Person Information Class

### 3. Personal Information Class

Design a class that holds the following personal data: name, address, age, and phone number. Write appropriate accessor and mutator methods. Demonstrate the class by writing a program that creates three instances of it. One instance should hold your information, and the other two should hold your friends' or family members' information.

Figure 37: Person Information Task Requirement

```
1 package personInformationClass;
2
3 public class Person {
4     private String name;
5     private String address;
6     private int age;
7     private String phoneNumber;
8
9     public Person(String name, String address, int age, String phoneNumber) {
10         this.name = name;
11         this.address = address;
12         this.age = age;
13         this.phoneNumber = phoneNumber;
14     }
15
16     public String getName() {
17         return name;
18     }
19
20     public String getAddress() {
21         return address;
22     }
23
24     public int getAge() {
25         return age;
26     }
27 }
```

```

28     public String getPhoneNumber() {
29         return phoneNumber;
30     }

32     public void setName(String name) {
33         this.name = name;
34     }

36     public void setAddress(String address) {
37         this.address = address;
38     }

40     public void setAge(int age) {
41         this.age = age;
42     }

44     public void setPhoneNumber(String phoneNumber) {
45         this.phoneNumber = phoneNumber;
46     }
47 }

```

28: Person.java

```

1 package personInformationClass;

3 public class Main {
4     public static void main(String[] args) {
5         Person myInfo = new Person("Your Name", "123 Main St, Hometown", 30, "
6             555-1234");
7         Person friendInfo = new Person("Friend Name", "456 Maple Ave, Nearby Town"
8             , 25, "555-5678");
9         Person familyMemberInfo = new Person("Family Member Name", "789 Oak Dr,
10             Family Town", 55, "555-8765");

11         displayPersonInfo(myInfo);
12         displayPersonInfo(friendInfo);
13         displayPersonInfo(familyMemberInfo);
14     }

16     public static void displayPersonInfo(Person person) {
17         System.out.println("Name: " + person.getName());
18         System.out.println("Address: " + person.getAddress());
19         System.out.println("Age: " + person.getAge());
20         System.out.println("Phone Number: " + person.getPhoneNumber());
21         System.out.println();
22     }
23 }

```

29: Main.java

```

Run Main x
/Library/Java/JavaVirtualMachines/jdk-22.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt
.jar=59238:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath /Users/kiyo9w/IdeaProjects/Assignment 6/Assignment 6/6.3 personInformationClass/out/production/6.3 personInformationClass
personInformationClass.Main
Name: Your Name
Address: 123 Main St, Hometown
Age: 30
Phone Number: 555-1234

Name: Friend Name
Address: 456 Maple Ave, Nearby Town
Age: 25
Phone Number: 555-5678

Name: Family Member Name
Address: 789 Oak Dr, Family Town
Age: 55
Phone Number: 555-8765

Process finished with exit code 0

```

Figure 38: Runtime Result

## 6.4 RetailItem Class

**4. RetailItem Class**

Write a class named `RetailItem` that holds data about an item in a retail store. The class should have the following fields:

- **description.** The `description` field references a `String` object that holds a brief description of the item.
- **unitsOnHand.** The `unitsOnHand` field is an `int` variable that holds the number of units currently in inventory.
- **price.** The `price` field is a `double` that holds the item's retail price.

Write a constructor that accepts arguments for each field, appropriate mutator methods that store values in these fields, and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three `RetailItem` objects and stores the following data in them:

	Description	Units on Hand	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

Figure 39: RetailItem Class Task Requirement

```

1 package retailItem;

3 public class RetailItem {
4     private String description;
5     private int unitsOnHand;
6     private double price;

8     public RetailItem(String description, int unitsOnHand, double price) {
9         this.description = description;
10        this.unitsOnHand = unitsOnHand;
11        this.price = price;

```

```

12     }

14     public String getDescription() {
15         return description;
16     }

18     public int getUnitsOnHand() {
19         return unitsOnHand;
20     }

22     public double getPrice() {
23         return price;
24     }

26     public void setDescription(String description) {
27         this.description = description;
28     }

30     public void setUnitsOnHand(int unitsOnHand) {
31         this.unitsOnHand = unitsOnHand;
32     }

34     public void setPrice(double price) {
35         this.price = price;
36     }
37 }

```

30: RetailItem.java

```

1 package retailItem;

3 public class Main {
4     public static void main(String[] args) {
5         RetailItem item1 = new RetailItem("Jacket", 12, 59.95);
6         RetailItem item2 = new RetailItem("Designer Jeans", 40, 34.95);
7         RetailItem item3 = new RetailItem("Shirt", 20, 24.95);

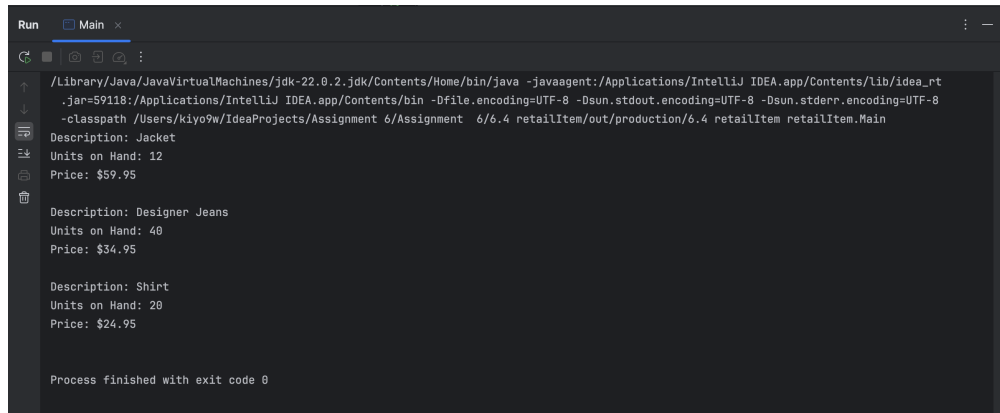
9         displayItemInfo(item1);
10        displayItemInfo(item2);
11        displayItemInfo(item3);
12    }

14    public static void displayItemInfo(RetailItem item) {
15        System.out.println("Description: " + item.getDescription());
16        System.out.println("Units on Hand: " + item.getUnitsOnHand());
17        System.out.println("Price: $" + item.getPrice());
18        System.out.println();
19    }
21 }

```

31: Main.java





```
Run Main x
/Library/Java/JavaVirtualMachines/jdk-22.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt
.jar=59118:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath /Users/kiyo9w/IdeaProjects/Assignment 6/Assignment 6/6.4 retailItem/out/production/6.4 retailItem retailItem.Main
Description: Jacket
Units on Hand: 12
Price: $59.95

Description: Designer Jeans
Units on Hand: 40
Price: $34.95

Description: Shirt
Units on Hand: 20
Price: $24.95

Process finished with exit code 0
```

Figure 40: Runtime Result