Name: Rajan Mishra

Uid: 22BCS13621

Class/Sec: 22BCS_IOT_615

Day 1 OOPs (Assignment 1)

Question 1: Find Whether a Number is Even or Odd

Answer:

Algorithm:

Start:

- Initialize the program.

Input:

- Prompt the user to enter an integer.

Process:

- Read the integer input.
- Check if the number is divisible by 2: ○ Use the modulus operator (%) to compute number % 2.
    - If the result is 0, the number is even.
    - Otherwise, the number is odd.

Output:

- Display whether the number is even or odd.

End:

- Exit the program.

Code:

```
import java.util.Scanner; public class
EvenOrOdd { public static void
main(String[] args) { // Create a
```

```
Scanner object for input Scanner

scanner = new Scanner(System.in);

// Prompt the user to enter a number

System.out.print("Enter an integer: ");

int number = scanner.nextInt(); //

Check if the number is even or odd if

(number % 2 == 0) {

System.out.println(number + " is even.");

} else {

System.out.println(number + " is odd.");

}

// Close the scanner scanner.close();

}

}
```

Output

```
Enter a number: 7

Result: 7 is odd.
```

Question 2: **Sum of Digits of a Number**

Answer:

Algorithm:

Input the number: Read the integer from the user.

Handle negatives: Convert the number to its absolute value.

Initialize sum: Set a variable sum to 0.

Extract digits: Use a loop to repeatedly extract the last digit using modulo (%) and add it to sum.

Remove digits: Divide the number by 10 (integer division) to discard the last digit.

Repeat until zero: Continue the loop until the number becomes 0.

Output the result: Print the sum of the digits.

Code:

```
import java.util.Scanner; public class
SumOfDigits { public static void
main(String[] args) {
Scanner scanner = new Scanner(System.in);
// Input the number
System.out.print("Enter a number: "); int
number = scanner.nextInt();

int sum = 0; int temp = Math.abs(number); // Handle
negative numbers

// Calculate the sum of digits while (temp > 0) {
sum += temp % 10; // Add the last digit to the
sum temp /= 10;      // Remove the last digit
}

// Output the result
System.out.println("The sum of the digits is: " + sum);
}
}
```

Output:

```
Enter a number: 123
The sum of the digits is: 6
```

Question 3: Check if a Number is Palindrome

Answer:

Algorithm:

Input the number: Read an integer from the user.

Handle negatives: Treat negative numbers as non-palindromes (optional).

Store the original number: Save the input number for comparison later.

Reverse the number:

- Initialize reversedNumber to 0.
- Extract the last digit of the number using modulo (%).
- Multiply reversedNumber by 10 and add the extracted digit.
- Remove the last digit of the number using integer division (/).
- Repeat until the number becomes 0.

Compare: Check if the original number is equal to the reversed number.

Output the result: Print whether the number is a palindrome or not.

Code:

```
import java.util.Scanner; public
class PalindromeNumber {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
// Input the number
System.out.print("Enter a number: ");
int number = scanner.nextInt(); //
Handle negatives (optional) if
(number < 0) {
System.out.println("Negative numbers are not palindromes."); return;
}
int originalNumber = number; // Store the original number int
reversedNumber = 0;
```

```
// Reverse the number while
(number > 0) {
int digit = number % 10;            // Extract the last digit
reversedNumber = reversedNumber * 10 + digit; // Build the reversed number
number /= 10;                       // Remove the last digit
}
// Check if the number is a palindrome if
(originalNumber == reversedNumber) {
System.out.println(originalNumber + " is a palindrome.");
} else {
System.out.println(originalNumber + " is not a palindrome.");
}
}
}
```

Output

```
Enter a number: 123
123 is not a palindrome.
```

Question 4: Find the Largest Element in an Array

Answer:

Algorithm:

Input the array: Read the array of integers.

Initialize the largest element: Set the first element of the array as the initial largest element.

Traverse the array:

- Loop through the array starting from the second element.
- Compare each element with the current largest element.

- If an element is greater than the current largest, update the largest element. Output the result: After completing the loop, print the largest element

Code:

```
import java.util.Scanner; public class
LargestElementInArray {     public static
void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Input the size of the array
      System.out.print("Enter the size of the array: ");
int size = scanner.nextInt();           // Input the array
elements        int[] arr = new int[size];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < size; i++) {
arr[i] = scanner.nextInt();
      }
      // Initialize the largest element as the first element
      int largest = arr[0];
      // Traverse the array to find the largest element
      for (int i = 1; i < size; i++) {
if (arr[i] > largest) {
            largest = arr[i]; // Update largest if a greater element is found
         }
      }
      // Output the largest element
      System.out.println("The largest element in the array is: " + largest);
   }
}
```

Output:

```
Enter the size of the array: 5
Enter the elements of the array:
2 4 1 7 3
The largest element in the array is: 7
```

Question 5: Calculate the Factorial of a Number

Answer:

Algorithm:

Input the number: Read an integer n for which you want to calculate the factorial.

Handle special case: If n is 0 or 1, the factorial is 1 (since 0! = 1! = 1).

Initialize the result: Set a variable result to 1.

Calculate factorial: Loop from 2 to n and multiply the current result by each number.

Output the result: After the loop, print the value of result as the factorial of n.

Code:

```
import java.util.Scanner; public class
Factorial { public static void
main(String[] args) {
Scanner scanner = new Scanner(System.in);
// Input the number
System.out.print("Enter a number:
"); int n = scanner.nextInt(); // Handle
special case for 0 and 1 if (n == 0 || n
== 1) {
System.out.println("Factorial of " + n + " is 1");
} else {
// Calculate factorial long
factorial = 1;
```

```
for (int i = 2; i <= n; i++) { factorial
*= i;
}
// Output the result
System.out.println("Factorial of " + n + " is " + factorial);
}
}
```

Output

```
Enter a number: 5
Factorial of 5 is 120
```

Question 6:

A Simple Bank Account Class Create a class BankAccount that represents a bank account. The class should have: • Private data members: accountNumber, balance. • Public methods: • deposit(double amount) to deposit money into the account. • withdraw(double amount) to withdraw money from the account. • display() to display the account details (accountNumber and balance).

Code:

```
// BankAccount Class public
class BankAccount { // Private
data members private String
accountNumber; private
double balance;
// Constructor to initialize account details public
BankAccount(String accountNumber, double initialBalance) {
this.accountNumber = accountNumber;
this.balance = initialBalance;
}
```

```
// Public method to deposit money
public void deposit(double amount) { if
(amount > 0) {
balance += amount;
System.out.println(amount + " deposited successfully.");
} else {
System.out.println("Deposit amount must be positive.");
}
}

// Public method to withdraw money
public void withdraw(double amount) { if
(amount > 0 && amount <= balance) {
balance -= amount;
System.out.println(amount + " withdrawn successfully.");
} else if (amount > balance) {
System.out.println("Insufficient balance to withdraw " + amount);
} else {
System.out.println("Withdrawal amount must be positive.");
}
}

// Public method to display account details public
void display() {
System.out.println("Account Number: " + accountNumber);
System.out.println("Balance: " + balance);
}

// Main method for testing the BankAccount class
```

```
public static void main(String[] args) {
// Create an object of BankAccount
BankAccount account = new BankAccount("123456789", 1000.0);

// Display account details account.display();

// Deposit money account.deposit(500.0);

// Withdraw money account.withdraw(300.0);

// Display updated account details account.display();

// Attempt to withdraw more than balance account.withdraw(2000.0);
}
}
```

Output

```
Account Number: 123456789
Balance: 1000.0
500.0 deposited successfully.
300.0 withdrawn successfully.
Account Number: 123456789
Balance: 1200.0
Insufficient balance to withdraw 2000.0
```

Question 7:

Extending the BankAccount Class for Different Account Types Now, extend the previous BankAccount class to handle different types of bank accounts: 1. Savings Account (inherits from BankAccount) with an interest rate. 2. Checking Account (inherits from BankAccount) with a fee for withdrawals. The base class BankAccount should still provide basic functionality, but now we want to add additional features to the derived classes: • SavingsAccount: Automatically apply interest every month. • CheckingAccount: Apply a withdrawal fee each time a withdrawal is made.

Code:

```
// Base class BankAccount
class BankAccount { // Private
data members private String
accountNumber; private
double balance;

// Constructor to initialize account details public
BankAccount(String accountNumber, double initialBalance) {
this.accountNumber = accountNumber;
this.balance = initialBalance;
}

// Public method to deposit money
public void deposit(double amount) { if
(amount > 0) {
balance += amount;
System.out.println(amount + " deposited successfully.");
} else {
System.out.println("Deposit amount must be positive.");
}
}

// Public method to withdraw money
public void withdraw(double amount) { if
(amount > 0 && amount <= balance) {
balance -= amount;
System.out.println(amount + " withdrawn successfully.");
} else if (amount > balance) {
```

```
System.out.println("Insufficient balance to withdraw " + amount);
} else {
System.out.println("Withdrawal amount must be positive.");
}
}

// Public method to display account details public
void display() {
System.out.println("Account Number: " + accountNumber);
System.out.println("Balance: " + balance);
}
}

// SavingsAccount class (inherits from BankAccount) class
SavingsAccount extends BankAccount { private double
interestRate; // Interest rate for savings account

// Constructor to initialize savings account with interest rate
public SavingsAccount(String accountNumber, double initialBalance, double
interestRate) { super(accountNumber, initialBalance);  // Call the constructor
of BankAccount this.interestRate = interestRate;
}

// Apply interest to the savings account every month
public void applyInterest() { double interest =
(super.balance * interestRate) / 100;
deposit(interest);  // Deposit interest to the account
System.out.println("Interest of " + interest + " applied.");
}
}
```

```
// CheckingAccount class (inherits from BankAccount)
class CheckingAccount extends BankAccount { private
double withdrawalFee; // Fee for each withdrawal

// Constructor to initialize checking account with withdrawal fee
public CheckingAccount(String accountNumber, double initialBalance, double
withdrawalFee) { super(accountNumber, initialBalance);  // Call the
constructor of BankAccount this.withdrawalFee = withdrawalFee;
}

// Withdraw money with withdrawal fee
@Override public void withdraw(double amount) { if (amount > 0 && amount +
withdrawalFee <= super.balance) { super.withdraw(amount + withdrawalFee);
// Subtract withdrawal amount + fee
System.out.println("Withdrawal fee of " + withdrawalFee + " applied.");
} else {
System.out.println("Insufficient balance to withdraw " + (amount + withdrawalFee));
}
}
}

// Main class to test BankAccount and its subclasses
public class Main { public static void main(String[]
args) {
// Create Savings Account
SavingsAccount savingsAccount = new SavingsAccount("SA123", 1000.0, 5.0);
savingsAccount.display(); savingsAccount.applyInterest();  // Apply interest
savingsAccount.display();
```

```
// Create Checking Account
CheckingAccount checkingAccount = new CheckingAccount("CA456", 1000.0, 2.0);
checkingAccount.display(); checkingAccount.withdraw(200.0);  // Withdraw with
fee checkingAccount.display();
}
}
```

Output

```
Account Number: SA123
Balance: 1000.0
Interest of 50.0 applied.
Account Number: SA123
Balance: 1050.0
Account Number: CA456
Balance: 1000.0
200.0 withdrawn successfully.
Withdrawal fee of 2.0 applied.
Account Number: CA456
Balance: 798.0
```