



Bangladesh University of Business & Technology

Assignment On

Course Title: Advanced Programming Lab

Course Code: CSE 332

Topic: Java Related Problems

Submitted By

Name: Md. Shahariar Nabi

Id: 21225103164

Intake: 49

Section: 4

Submitted To

Md. Mahbubur Rahman

Assistant Professor,

Department of Computer Science and Engineering,

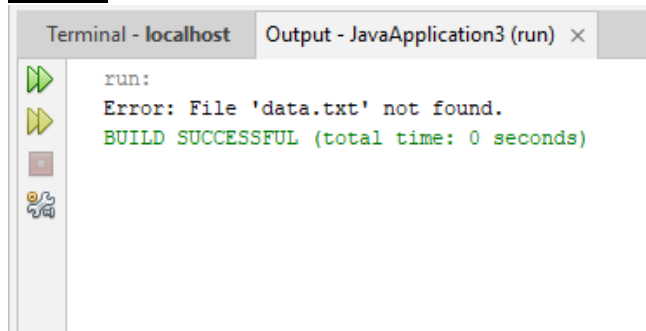
Bangladesh University of Business & Technology

Questions 1: Write a Java program that reads data from a file named "data.txt". Implement error handling using try-catch blocks to handle File Not Found Exception. If the file is not found, print an error message indicating the issue.

Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class FileReaderExample {
    public static void main(String[] args) {
        String fileName = "data.txt";
        try {
            File file = new File(fileName);
            Scanner scanner = new Scanner(file);
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                System.out.println(line);
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File '" + fileName + "' not found.");
        }
    }
}
```

Output



Questions 2: Write a Java program that initializes an array of integers and attempts to access an element at an index beyond the array's length. Implement try-catch blocks to handle the Array Index Out Of Bounds Exception that may occur. If the exception occurs, print a message indicating the invalid index.

Code:

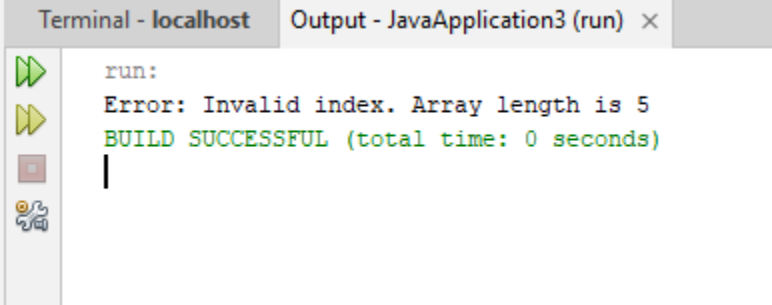
```
public class ArrayAccessExample {
    public static void main(String[] args) {
        int[] numbers = { 10, 20, 30, 40, 50 };
        try {
            int index = 10;
            int value = numbers[index];
        }
    }
}
```

```

        System.out.println("Value at index " + index + ": " + value);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Error: Invalid index. Array length is " + numbers.length);
    }
}
}
}

```

Output



```

Terminal - localhost  Output - JavaApplication3 (run) x
run:
Error: Invalid index. Array length is 5
BUILD SUCCESSFUL (total time: 0 seconds)
|

```

Questions 3: Write a Java program to simulate bank account transactions. Implement try-catch blocks to handle exceptions that may occur during withdrawal or deposit operations, such as `InsufficientFundsException` for insufficient balance and `NegativeAmountException` for negative amounts. Use a finally block to ensure that resources are properly released after each transaction.

Code:

```

import java.util.Scanner;
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}
class NegativeAmountException extends Exception {
    public NegativeAmountException(String message) {
        super(message);
    }
}
class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    public void deposit(double amount) throws NegativeAmountException {
        if (amount < 0) {
            throw new NegativeAmountException("Deposit amount cannot be negative.");
        }
        balance += amount;
    }
    public void withdraw(double amount) throws InsufficientFundsException,
NegativeAmountException {

```

```

    if (amount < 0) {
        throw new NegativeAmountException("Withdrawal amount cannot be negative.");
    }
    if (amount > balance) {
        throw new InsufficientFundsException("Insufficient balance for withdrawal.");
    }
    balance -= amount;
}
public double getBalance() {
    return balance;
}
}
public class BankTransactionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter initial balance: ");
            double initialBalance = scanner.nextDouble();
            BankAccount account = new BankAccount(initialBalance);
            System.out.print("Enter deposit amount: ");
            double depositAmount = scanner.nextDouble();
            account.deposit(depositAmount);
            System.out.print("Enter withdrawal amount: ");
            double withdrawalAmount = scanner.nextDouble();
            account.withdraw(withdrawalAmount);
            System.out.println("Updated balance: $" + account.getBalance());
        } catch (NegativeAmountException e) {
            System.err.println("Error: " + e.getMessage());
        } catch (InsufficientFundsException e) {
            System.err.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
}

```

Output

```

Terminal - localhost  Output - JavaApplication3 (run) ×
run:
Enter initial balance: 100
Enter deposit amount: 10
Enter withdrawal amount: 50
Updated balance: $60.0
BUILD SUCCESSFUL (total time: 54 seconds)
Ant Settings

```

Questions 4: Imagine you have a bank account. You can deposit and withdraw money from your account. You should keep in mind that the total amount of money withdrawn from your account must not exceed the total balance present in your account. If such a scenario happens, you need to safely execute from the banking system by using scanner. Implement the above case in Java with the proper utilization of user-defined exception mechanism.

Code:

```
import java.util.Scanner;
class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}
class NegativeAmountException extends Exception {

    public NegativeAmountException(String message) {
        super(message);
    }
}
class BankAccount {
    private double balance;
    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    public void deposit(double amount) throws NegativeAmountException {
        if (amount < 0) {
            throw new NegativeAmountException("Deposit amount cannot be negative.");
        }
        balance += amount;
    }
    public void withdraw(double amount) throws InsufficientBalanceException,
    NegativeAmountException {
        if (amount < 0) {
            throw new NegativeAmountException("Withdrawal amount cannot be negative.");
        }
        if (amount > balance) {
            throw new InsufficientBalanceException("Insufficient balance for withdrawal.");
        }
        balance -= amount;
    }
    public double getBalance() {
        return balance;
    }
}
public class BankApplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter initial balance: ");
            double initialBalance = scanner.nextDouble();
```

```

BankAccount account = new BankAccount(initialBalance);
    System.out.print("Enter deposit amount: ");
    double depositAmount = scanner.nextDouble();
    account.deposit(depositAmount);
    System.out.print("Enter withdrawal amount: ");
    double withdrawalAmount = scanner.nextDouble();
    account.withdraw(withdrawalAmount);
    System.out.println("Updated balance: $" + account.getBalance());
} catch (NegativeAmountException e) {
    System.out.println("Error: " + e.getMessage());
} catch (InsufficientBalanceException e) {
    System.out.println("Error: " + e.getMessage());
} finally {
    scanner.close();
}
}
}

```

Output

```

Terminal - localhost  Output - JavaApplication3 (run) x
run:
Enter initial balance: 20
Enter deposit amount: 40
Enter withdrawal amount: 65
Error: Insufficient balance for withdrawal.
BUILD SUCCESSFUL (total time: 19 seconds)

```

Questions 5: Write a Java program to validate an email address entered by the user. Implement multiple catch blocks to handle different types of exceptions that may occur during validation, such as `IllegalArgumentException` for invalid format and `NullPointerException` for null input. Use a finally block to close any resources opened during validation.

Code:

```

import java.util.Scanner;
public class EmailValidator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter an email address: ");
            String email = scanner.nextLine();

            validateEmail(email);
            System.out.println("Email address is valid!");
        } catch (IllegalArgumentException e) {
            System.err.println("Error: Invalid email format.");
        } catch (NullPointerException e) {
            System.err.println("Error: Email cannot be null.");
        } finally {
            scanner.close();
        }
    }
}

```

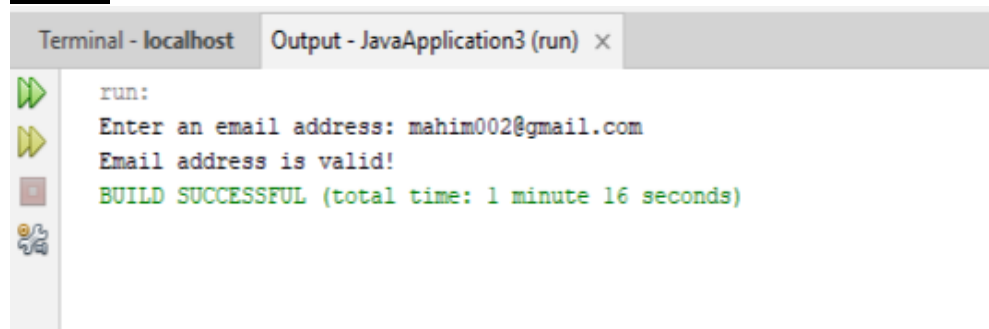
```

}
}
private static void validateEmail(String email) {
    if (email == null) {
        throw new NullPointerException();
    }

    if (!email.matches("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}")) {
        throw new IllegalArgumentException();
    }
}
}
}
}

```

Output



```

Terminal - localhost  Output - JavaApplication3 (run) x
run:
Enter an email address: mahim002@gmail.com
Email address is valid!
BUILD SUCCESSFUL (total time: 1 minute 16 seconds)

```

Questions 6: Write a program to create four threads. Inside the first thread print your Dept. 10 times but wait for 2 second before printing each time. Inside the second thread print your Name 20 times. Inside the third thread print your ID 30 times. Make sure second thread gets more OS access than the first thread and the third thread starts after finishing the second thread.

Code:

```

import java.util.Scanner;
public class MultithreadingExample {
    public static void main(String[] args) throws InterruptedException {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your department: ");
        String department = scanner.nextLine();
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();
        System.out.print("Enter your ID: ");
        String id = scanner.nextLine();
        Thread thread1 = new Thread(() -> {
            for (int i = 0; i < 10; i++) {
                System.out.println("Dept: " + department);
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
    }
}

```


Questions 7: Write a Java program to perform matrix multiplication using multithreading for parallel computation. Implement a method that takes two matrices as input and computes their product using multiple threads, each responsible for computing a portion of the result matrix. Ensure efficient utilization of resources and minimize thread synchronization overhead.

Code:

```
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
public class MatrixMultiplication {
    private static final int THREAD_POOL_SIZE = 4;
    public static int[][] multiply(int[][] A, int[][] B) {
        int rowsA = A.length;
        int colsA = A[0].length;
        int colsB = B[0].length;
        int[][] result = new int[rowsA][colsB];
        ExecutorService executor = Executors.newFixedThreadPool(THREAD_POOL_SIZE);
        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                executor.execute(new Worker(A, B, result, i, j));
            }
        }
        executor.shutdown();
        try {
            executor.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        return result;
    }
    private static class Worker implements Runnable {
        private final int[][] A;
        private final int[][] B;
        private final int[][] result;
        private final int row;
        private final int col;
        public Worker(int[][] A, int[][] B, int[][] result, int row, int col) {
            this.A = A;
            this.B = B;
            this.result = result;
            this.row = row;
            this.col = col;
        }
        @Override
        public void run() {
            int sum = 0;
            for (int k = 0; k < A[0].length; k++) {
                sum += A[row][k] * B[k][col];
            }
        }
    }
}
```

```

result[row][col] = sum;}
}
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of the first matrix:");
    int rowsA = scanner.nextInt();
    int colsA = scanner.nextInt();
    int[][] A = new int[rowsA][colsA];
    System.out.println("Enter the elements of the first matrix:");
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            A[i][j] = scanner.nextInt();}
        }
    System.out.println("Enter the number of rows and columns of the second matrix:");
    int rowsB = scanner.nextInt();
    int colsB = scanner.nextInt();
    if (colsA != rowsB) {
        System.out.println("Matrix multiplication not possible.");
        return;}
    int[][] B = new int[rowsB][colsB];
    System.out.println("Enter the elements of the second matrix:");
    for (int i = 0; i < rowsB; i++) {
        for (int j = 0; j < colsB; j++) {
            B[i][j] = scanner.nextInt();}
        }
    int[][] result = multiply(A, B);
    System.out.println("Resultant matrix:");
    for (int i = 0; i < result.length; i++) {
        for (int j = 0; j < result[0].length; j++) {
            System.out.print(result[i][j] + " ");
        }
        System.out.println();
    }
    scanner.close();
}
}

```

Output:

```

Terminal - localhost  Output - JavaApplication3 (run) x
run:
Enter the number of rows and columns of the first matrix:
2
2
Enter the elements of the first matrix:
2
2
2
2
Enter the number of rows and columns of the second matrix:
2
2
Enter the elements of the second matrix:
2
2
1
2
Resultant matrix:|
6 8
6 8
BUILD SUCCESSFUL (total time: 1 minute 18 seconds)

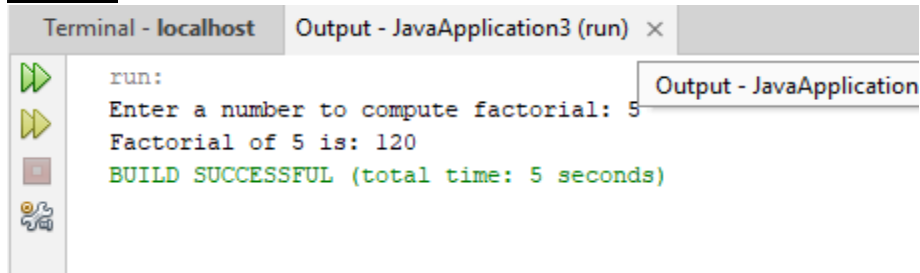
```

Questions 8: Write a Java program to compute the factorial of a given number using multithreading. Create two threads, one for computing the factorial of even numbers and the other for computing the factorial of odd numbers. Combine the results to get the final factorial.

Code:

```
import java.math.BigInteger;
import java.util.Scanner;
class FactorialThread implements Runnable {
    private int start;
    private int step;
    private int n;
    private BigInteger result = BigInteger.ONE;
    public FactorialThread(int start, int step, int n) {
        this.start = start;
        this.step = step;
        this.n = n;
    }
    public BigInteger getResult() {
        return result;
    }
    @Override
    public void run() {
        for (int i = start; i <= n; i += step) {
            result = result.multiply(BigInteger.valueOf(i));
        }
    }
}
public class Factorial {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to compute factorial: ");
        int n = scanner.nextInt();
        scanner.close();
        FactorialThread evenTask = new FactorialThread(2, 2, n);
        FactorialThread oddTask = new FactorialThread(1, 2, n);
        Thread evenThread = new Thread(evenTask);
        Thread oddThread = new Thread(oddTask);
        evenThread.start();
        oddThread.start();
        try {
            evenThread.join();
            oddThread.join();
        } catch (InterruptedException e) {
            System.err.println("Thread interrupted: " + e.getMessage());
        }
        BigInteger totalFactorial = evenTask.getResult().multiply(oddTask.getResult());
        System.out.println("Factorial of " + n + " is: " + totalFactorial);
    }
}
```

Output:



```
Terminal - localhost Output - JavaApplication3 (run) x
run:
Enter a number to compute factorial: 5
Factorial of 5 is: 120
BUILD SUCCESSFUL (total time: 5 seconds)
```

Questions 9: Write a Java program that creates two threads, one for printing uppercase letters from A to Z and the other for printing lowercase letters from a to z. Ensure that the letters are printed in sequence, with uppercase letters followed by lowercase letters.

Code:

```
public class PrintLetters {
    private static final Object lock = new Object();
    private static boolean uppercaseTurn = true;
    public static void main(String[] args) {
        Thread uppercaseThread = new Thread(new PrintUppercase());
        Thread lowercaseThread = new Thread(new PrintLowercase());
        uppercaseThread.start();
        lowercaseThread.start();
    }
    static class PrintUppercase implements Runnable {
        @Override
        public void run() {
            synchronized (lock) {
                for (char letter = 'a'; letter <= 'z'; letter++) {
                    while (!uppercaseTurn) {
                        try {
                            lock.wait();
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                        }
                    }
                    System.out.print(letter + " ");
                    uppercaseTurn = false;
                    lock.notifyAll();
                }
            }
        }
    }
    static class PrintLowercase implements Runnable {
        @Override
        public void run() {
            synchronized (lock) {
                for (char letter = 'A'; letter <= 'Z'; letter++) {
                    while (uppercaseTurn) {
                        try {
```

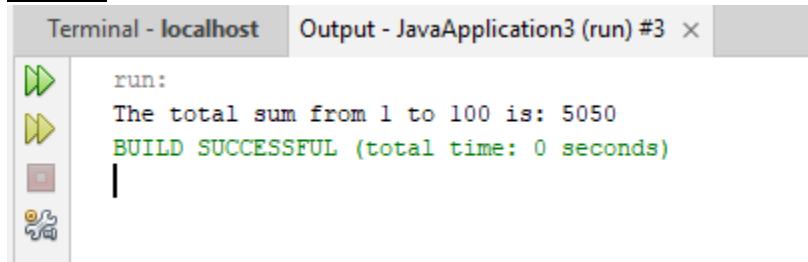


```

        System.out.println("The total sum from 1 to 100 is: " + totalSum);
    }
    static class SumTask implements Runnable {
        private int threadIndex;
        private int start;
        private int end;
        public SumTask(int threadIndex, int start, int end) {
            this.threadIndex = threadIndex;
            this.start = start;
            this.end = end;
        }
        @Override
        public void run() {
            int sum = 0;
            for (int i = start; i <= end; i++) {
                sum += i;
            }
            results[threadIndex] = sum;
        }
    }
}
}
}

```

Output



```

Terminal - localhost  Output - JavaApplication3 (run) #3 x
run:
The total sum from 1 to 100 is: 5050
BUILD SUCCESSFUL (total time: 0 seconds)

```

Questions 11: Write a program that takes a paragraph of text as input and counts the occurrences of each word. Additionally, identify the five most common words and display them along with their frequencies.

Code:

```

import java.util.*;
public class WordCounter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a paragraph of text:");
        String paragraph = scanner.nextLine();
        paragraph = paragraph.replaceAll("[^a-zA-Z ]", "").toLowerCase();
        String[] words = paragraph.split("\\s+");
        Map<String, Integer> wordFrequency = new HashMap<>();
        for (String word : words) {
            wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);
        }
        List<Map.Entry<String, Integer>>sortedList=new
        ArrayList<>(wordFrequency.entrySet());
        sortedList.sort((a, b) -> b.getValue().compareTo(a.getValue()));
    }
}

```

```

System.out.println("Five most common words and their frequencies:");
for (int i = 0; i < 5 && i < sortedList.size(); i++) {
    Map.Entry<String, Integer> entry = sortedList.get(i);
    System.out.println(entry.getKey() + ": " + entry.getValue());
}
}
}
}

```

Output:

```

Terminal - localhost  Output - JavaApplication3 (run) #3 x
run:
Enter a paragraph of text:
he is a student good student engineering student
Five most common words and their frequencies:
student: 3
a: 1
is: 1
engineering: 1
he: 1
BUILD SUCCESSFUL (total time: 10 seconds)

```

Questions 12: Write a program that takes a sentence and a word as input and finds whether the word is present as a substring in the sentence.

Code

```

import java.util.Scanner;
public class Checker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a sentence: ");
        String sentence = scanner.nextLine();
        System.out.print("Enter a word: ");
        String word = scanner.nextLine();
        if (isSubstring(sentence, word)) {
            System.out.println("The word '" + word + "' is a substring of the sentence.");
        } else {
            System.out.println("The word '" + word + "' is not a substring of the sentence.");
        }
    }
    public static boolean isSubstring(String sentence, String word) {
        return sentence.toLowerCase().contains(word.toLowerCase());
    }
}

```

Output:

```

Output - JavaApplication3 (run)
run:
Enter a sentence: he is a student
Enter a word: is
The word 'is' is a substring of the sentence.
BUILD SUCCESSFUL (total time: 22 seconds)

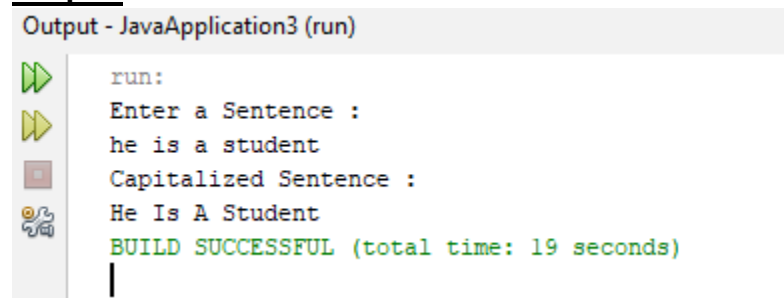
```

Questions 13: Write a program that takes a sentence as input and capitalizes the first letter of each word. For example, "hello world" should become "Hello World".

Code

```
import java.util.Scanner;
public class Word {
    public static void main(String [] args){
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a Sentence : ");
        String sentence=scanner.nextLine();
        String[] words=sentence.split("\\s+");
        StringBuilder capital= new StringBuilder();
        for (String word:words){
            if(word.length(>0)){
                capital.append(Character.toUpperCase(word.charAt(0))).append(word.substring
                (1).toLowerCase()).append(" ");
            }
        }
        System.out.println("Capitalized Sentence : ");
        System.out.println(capital.toString().trim());
    }
}
```

Output:



```
Output - JavaApplication3 (run)
run:
Enter a Sentence :
he is a student
Capitalized Sentence :
He Is A Student
BUILD SUCCESSFUL (total time: 19 seconds)
|
```

Questions 14: Create a function that takes a sentence as input and reverses the order of words in it. For example, "Hello world" should become "world Hello".

Code

```
import java.util.Scanner;
public class Reverse {
    public static void main(String [] args){
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a Sentence : ");
        String sentence=scanner.nextLine();
        String[] words=sentence.split("\\s+");
        StringBuilder reverse= new StringBuilder();
        for(int i=words.length-1;i>=0;i--){
            reverse.append(words[i]).append(" ");
        }
        System.out.println("Reversed Sentence: "+reverse.toString().trim());
    }
}
```


Output:

```
Output - JavaApplication3 (run)
run:
Enter a Sentence :
he is a student
Reversed Sentence: student a is he
BUILD SUCCESSFUL (total time: 21 seconds)
|
```

Questions 15: Write a program that counts the occurrences of each character in a given string and displays the count for each character.

Code

```
import java.util.Scanner;
public class Count {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        System.out.println("Enter a sentence : ");
        String inputString=scanner.nextLine();
        int[] charCounts=new int[256];
        inputString = inputString.toLowerCase();
        char[] charArray = inputString.toCharArray();
        for(int i=0;i<charArray.length;i++){
            char c= charArray[i];
            if(c!=' ') charCounts[c]++;
        }
        for(int i=0;i<charCounts.length;i++){
            if(charCounts[i]>0) System.out.println("Character "+(char)i+ " is total:"+charCounts[i]);
        }
    }
}
```

Output:

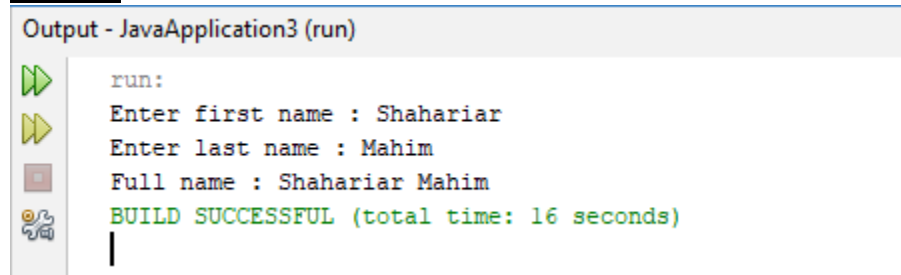
```
Output - JavaApplication3 (run)
run:
Enter a sentence :
a student
Character a is total :1
Character d is total :1
Character e is total :1
Character n is total :1
Character s is total :1
Character t is total :2
Character u is total :1
BUILD SUCCESSFUL (total time: 14 seconds)
```

Questions 16: Write a program that takes the first name and last name of a person as input and concatenates them to form a full name.

Code

```
import java.util.Scanner;
public class Add {
    public static void main(String [] args){
        Scanner scanner=new Scanner(System.in);
        System.out.print("Enter first name : ");
        String first=scanner.nextLine();
        System.out.print("Enter last name : ");
        String last=scanner.nextLine();
        String full=first+" "+last;
        System.out.println("Full name : "+full);
    }
}
```

Output:



```
Output - JavaApplication3 (run)
run:
Enter first name : Shahariar
Enter last name : Mahim
Full name : Shahariar Mahim
BUILD SUCCESSFUL (total time: 16 seconds)
```

Questions 17: Given the following strings: A = "The early bird catches the worm" B = "Patience is a virtue" Your task is to extract the word "early" from A and "virtue" from B. Then, concatenate these two words to form a sentence. After that, capitalize the sentence and find the last occurrence of the letter 'V' from the capitalized sentence. Perform all of these tasks using proper String class functions.

Code

```
public class Sen {
    public static void main(String [] args){
        String A= "The early bird catches the worm";
        String B= "Patience is a virtue";
        String word1= A.substring(4,9);
        String word2= B.substring(14,20);
        String concat= word1+" "+word2;
        String capital= concat.toUpperCase();
        int lastV= capital.lastIndexOf('V');
        System.out.println("Extracted word from A is "+word1+" and B is "+word2);
        System.out.println("Concatenated sentence : "+concat);
        System.out.println("Capitalized sentence : "+capital);
        System.out.println("Last occurrence of letter 'V' : "+lastV);
    }
}
```

Output:

```
Output - JavaApplication3 (run)
run:
Extracted word from A is early and B is virtue
Concatenated sentence : early virtue
Capitalized sentence : EARLY VIRTUE
Last occurrence of letter 'V' : 6
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

Questions 18: You are developing a ticket booking system for a movie theater. Design a Java program that uses a Queue to manage ticket requests, where each request represents a customer wanting to book a ticket. Implement methods to add new booking requests, process bookings in the order they were received, and display the status of ticket bookings.

Code

```
import java.util.*;
class TicketRequest {
    private String customerName;
    private int requestedSeats;
    public TicketRequest(String customerName,int requestedSeats){
        this.customerName = customerName;
        this.requestedSeats = requestedSeats;
    }
    public String getCustomerName(){
        return customerName;
    }
    public int getRequestedSeats(){
        return requestedSeats;
    }
    public String toString(){
        return "Customer Name : "+customerName+" and No. of Seats Want : "+requestedSeats;
    }
}
class TicketBookingSystem {
    private Queue<TicketRequest> bookingQueue;
    public TicketBookingSystem() {
        bookingQueue = new LinkedList<>();
    }
    public void addBookingRequest(TicketRequest request) {
        bookingQueue.add(request);
        System.out.println("Added booking request : " + request);
    }
    public void processNextBooking() {
        if(bookingQueue.isEmpty()) {
            System.out.println("No booking requests to process.");
        } else {
            TicketRequest request = bookingQueue.poll();
```

```

        System.out.println("Processed booking request : " + request);
    }
}
public void displayBookingQueueStatus() {
    if (bookingQueue.isEmpty()) {
        System.out.println("No pending booking requests.");
    } else {
        System.out.println("Pending booking requests :");
        for (TicketRequest request : bookingQueue) {
            System.out.println(request);
        }
    }
}
}
}
public class Tbank {
    public static void main(String [] args){
        TicketBookingSystem bookingSystem= new TicketBookingSystem();
        Scanner scanner= new Scanner(System.in);
        boolean exit=false;
        while(!exit) {
            System.out.println("1. Add booking request");
            System.out.println("2. Process next booking");
            System.out.println("3. Display booking queue status");
            System.out.println("4. Exit");
            System.out.print("Enter your choice : ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch(choice){
                case 1: System.out.print("Enter customer name : ");
                    String customerName =scanner.nextLine();
                    System.out.print("Enter number of requested seats : ");
                    int requestedSeats= scanner.nextInt();
                    scanner.nextLine();
                    TicketRequest request= new TicketRequest(customerName, requestedSeats);
                    bookingSystem.addBookingRequest(request);
                    break;
                case 2:
                    bookingSystem.processNextBooking();
                    break;
                case 3:
                    bookingSystem.displayBookingQueueStatus();
                    break;
                case 4:
                    exit = true;
                    System.out.println("Exit...");
                    break;
                default:
                    System.out.println("Invalid!!! Please try again.");
            }
        }
    }
}

```

```

    }
}
}

```

Output:

```

Output - JavaApplication3 (run)
run:
1. Add booking request
2. Process next booking
3. Display booking queue status
4. Exit
Enter your choice : 1
Enter customer name : mahim
Enter number of requested seats : 1
Added booking request : Customer Name : mahim and No. of Seats Want : 1
1. Add booking request
2. Process next booking
3. Display booking queue status
4. Exit
Enter your choice : 2
Processed booking request : Customer Name : mahim and No. of Seats Want : 1
1. Add booking request
2. Process next booking
3. Display booking queue status
4. Exit
Enter your choice : 3
No pending booking requests.
1. Add booking request
2. Process next booking
3. Display booking queue status
4. Exit
Enter your choice : 4
Exit...
BUILD SUCCESSFUL (total time: 49 seconds)
|

```

Questions 19: Create a class named Car with properties such as price (double), brand (String), and speed (double). These properties will be initialized when an object of the class is created. Create five objects of the Car class and add them to an ArrayList. Display the cars whose price is over 2000000 takas. Complete the program.

Code

```

import java.util.*;
class Car {
    private double price; private String brand; private double speed;
    public Car(double price, String brand, double speed) {
        this.price = price; this.brand = brand; this.speed = speed;
    }
    public double getPrice() {
        return price;
    }
    public String getBrand() {
        return brand;
    }
    public double getSpeed() {
        return speed;
    }
    public String toString() {
        return "Price : " +price+"", Brand : "+brand+" and Speed : "+speed;
    }
}

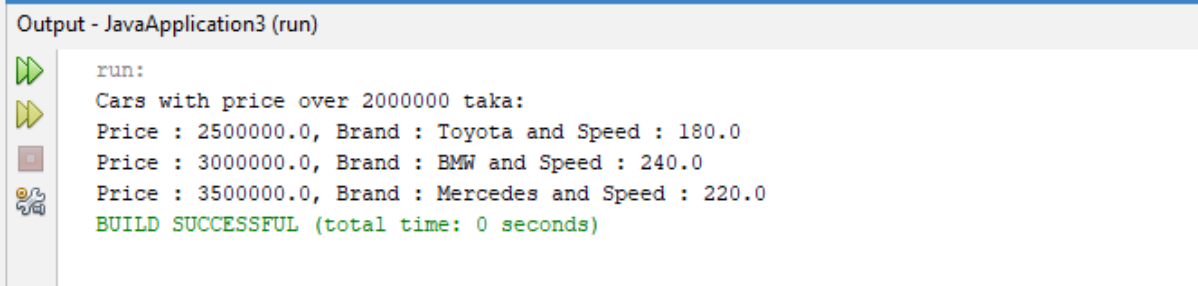
```

```

    }
    public static void main(String [] args) {
        ArrayList<Car> cars= new ArrayList<>();
        cars.add(new Car(2500000, "Toyota", 180));
        cars.add(new Car(1500000, "Honda", 160));
        cars.add(new Car(3000000, "BMW", 240));
        cars.add(new Car(3500000, "Mercedes", 220));
        cars.add(new Car(1800000, "Nissan", 170));
        System.out.println("Cars with price over 2000000 taka:");
        for (Car car : cars) {
            if (car.getPrice(>2000000) System.out.println(car);
        }
    }
}

```

Output:



```

Output - JavaApplication3 (run)
run:
Cars with price over 2000000 taka:
Price : 2500000.0, Brand : Toyota and Speed : 180.0
Price : 3000000.0, Brand : BMW and Speed : 240.0
Price : 3500000.0, Brand : Mercedes and Speed : 220.0
BUILD SUCCESSFUL (total time: 0 seconds)

```

Questions 20: Write a basic Java program for managing student IDs and their grades in a gradebook system. Implement methods to add new student IDs, remove existing student IDs, display the list of student IDs, and store/display grades for each student. Utilize simple data structures like arrays for storing student IDs and grades.

Code

```

import java.util.Scanner;
public class Gradebook {
    private static final int MAX_STUDENTS = 100;
    private String[] studentIds = new String[MAX_STUDENTS];
    private double[] mathGrades = new double[MAX_STUDENTS];
    private double[] englishGrades = new double[MAX_STUDENTS];
    private int numStudents = 0;
    public void addStudent(String studentId, double mathGrade, double englishGrade) {
        if (numStudents < MAX_STUDENTS) {
            studentIds[numStudents] = studentId;
            mathGrades[numStudents] = mathGrade;
            englishGrades[numStudents] = englishGrade;
            numStudents++;
            System.out.println("Student added successfully!");
        } else {
            System.out.println("Gradebook is full. Cannot add more students.");
        }
    }
    public void removeStudent(String studentId) {

```

```

for (int i = 0; i < numStudents; i++) {
    if (studentIds[i].equals(studentId)) {

        for (int j = i; j < numStudents - 1; j++) {
            studentIds[j] = studentIds[j + 1];
            mathGrades[j] = mathGrades[j + 1];
            englishGrades[j] = englishGrades[j + 1];
        }
        numStudents--;
        System.out.println("Student removed successfully!");
        return;
    }
}
System.out.println("Student with ID " + studentId + " not found.");
}

public void displayStudents() {
    System.out.println("Student IDs:");
    for (int i = 0; i < numStudents; i++) {
        System.out.println(studentIds[i]);
    }
}

public void displayGrades(String studentId) {
    for (int i = 0; i < numStudents; i++) {
        if (studentIds[i].equals(studentId)) {
            System.out.println("Math Grade: " + mathGrades[i]);
            System.out.println("English Grade: " + englishGrades[i]);
            return;
        }
    }
    System.out.println("Student with ID " + studentId + " not found.");
}

public static void main(String[] args) {
    Gradebook gradebook = new GradebookSystem();
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\nGradebook Menu:");
        System.out.println("1. Add Student");
        System.out.println("2. Remove Student");
        System.out.println("3. Display Student IDs");
        System.out.println("4. Display Student Grades");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        switch (choice) {
            case 1:
                System.out.print("Enter student ID: ");
                String id = scanner.nextLine();
                System.out.print("Enter math grade: ");
                double mathGrade = scanner.nextDouble();

```

```

        System.out.print("Enter English grade: ");
        double englishGrade = scanner.nextDouble();
        gradebook.addStudent(id, mathGrade, englishGrade);
        break;
    case 2:
        System.out.print("Enter student ID to remove: ");
        String removeId = scanner.nextLine();
        gradebook.removeStudent(removeId);
        break;
    case 3:
        gradebook.displayStudents();
        break;
    case 4:
        System.out.print("Enter student ID to display grades: ");
        String displayId = scanner.nextLine();
        gradebook.displayGrades(displayId);
        break;
    case 5:
        System.out.println("Exit");
        System.exit(0);
    default:
        System.out.println("Invalid choice. Please try again.");
    }
}
}
}
private static class GradebookSystem extends Gradebook {
    public GradebookSystem() {
    }
}
}
}

```

Output

```

run:
Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 1
Enter student ID: 12
Enter math grade: 3.9
Enter English grade: 3.8
Student added successfully!

Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 1
Enter student ID: 13
Enter math grade: 0.0
Enter English grade: 2.0
Student added successfully!

Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 2
Enter student ID to remove: 13
Student removed successfully!

Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 3
Student IDs:
12

Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 4
Enter student ID to display grades: 12
Math Grade: 3.9
English Grade: 3.8

Gradebook Menu:
1. Add Student
2. Remove Student
3. Display Student IDs
4. Display Student Grades
5. Exit
Enter your choice: 5
Exit
BUILD SUCCESSFUL (total time: 53 seconds)
|

```


Questions 21: Create a class named Student. Write a program to insert 10 Student objects in a Stack list. Now take user input for a variable named “menu”. If menu is 1 then insert another Student object. If menu is 2, delete the top Student object from the stack list. If menu is 3, just output the top Student object. Use proper stack list methods.

Code

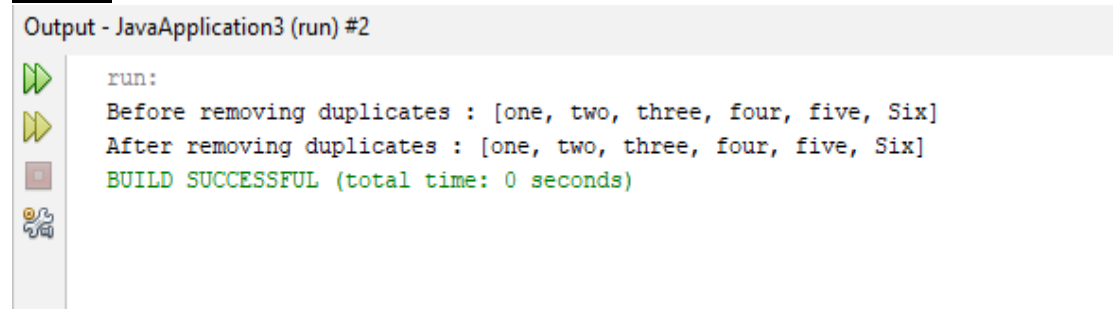
```
import java.util.*;
class Student {
    String name;
    String rollNumber;
    public Student(String name, String rollNumber) {
        this.name= name;
        this.rollNumber= rollNumber;
    }
    public String toString() {
        return "Student Name: "+name+ ", Roll Number: "+rollNumber;
    }
}
public class Std {
    public static void main(String [] args) {
        Stack<Student> stack=new Stack<>();
        for(int i=1;i<=10;i++){
            Student student = new Student("Student "+ i,"Roll-"+i);
            stack.push(student);
        }
        Scanner scanner =new Scanner(System.in);
        while(true){
            System.out.println("\nMenu:");
            System.out.println("1. Insert");
            System.out.println("2. Delete");
            System.out.println("3. Output");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int menu = scanner.nextInt();
            scanner.nextLine();
            switch(menu){
                case 1:
                    System.out.print("Enter student name : ");
                    String name = scanner.nextLine();
                    System.out.print("Enter student roll number : ");
                    String rollNumber= scanner.nextLine();
                    Student newStudent= new Student(name, rollNumber);
                    stack.push(newStudent);
                    System.out.println("Student object inserted successfully!");
                    break;
                case 2:
                    if(!stack.isEmpty()){
                        Student deletedStudent= stack.pop();
                        System.out.println("Top student object deleted : "+deletedStudent);
```


Questions 22: Write a Java program to remove duplicates from a list of strings. Implement a method remove duplicates that takes a List of strings as input and removes any duplicate elements, keeping only the first occurrence of each element.

Code

```
import java.util.*;
public class Dupli {
    public static void removeDuplicates(List<String> list){
        Set<String> set= new LinkedHashSet<>(list);
        list.clear();
        list.addAll(set);
    }
    public static void main(String [] args){
        List<String>strings =new
        ArrayList<>(Arrays.asList("one","two","three","four","five","Six"));
        System.out.println("Before removing duplicates : "+strings);
        removeDuplicates(strings);
        System.out.println("After removing duplicates : "+strings);
    }
}
```

Output



```
Output - JavaApplication3 (run) #2
run:
Before removing duplicates : [one, two, three, four, five, Six]
After removing duplicates : [one, two, three, four, five, Six]
BUILD SUCCESSFUL (total time: 0 seconds)
```