Week - 4

Create and initialize a String and determine the following. (Built in String methods may be used):

a) the length of the string.
b) the number of a’s in the string.
c) convert all lowercase to uppercase and vice versa.
d) find if a sub string pattern “abc” is present; if so, display the first position of occurrence.

(Be aware of multiple ways of string creation and accessing single characters)

```
public class StringOperations {
  public static void main(String[] args) {
    // Create and initialize a string
    String myString = "abCde Abcdefg abcDEF";

    // a) Find the length of the string
    int length = myString.length();
    System.out.println("The length of the string is: " + length);

    // b) Count the number of 'a's in the string
    long count_a = myString.chars().filter(ch -> ch == 'a').count();
    System.out.println("The number of 'a's in the string is: " + count_a);

    // c) Convert all lowercase to uppercase and vice versa
    String uppercaseString = myString.toUpperCase();
    String lowercaseString = myString.toLowerCase();
    System.out.println("Uppercase string: " + uppercaseString);
    System.out.println("Lowercase string: " + lowercaseString);

    // d) Find if the substring pattern "abc" is present and display the first position of
occurrence
    String substring = "abc";
    int position = myString.indexOf(substring);
    if (position != -1) {
      System.out.println("The substring '" + substring + "' is present at position: " +
position);
    } else {
      System.out.println("The substring '" + substring + "' is not present in the string.");
    }
  }
}
```

Create and initialize an array of strings. Display them. Every string should contain a sentence. Capitalize every starting letter of the words in the string. Assume that one blank separate every 2
words in a string. Display the resultant array.

```
public class CapitalizeWords {
   public static void main(String[] args) {
      // Create and initialize an array of strings containing sentences
      String[] sentences = {
         "this is a sample sentence",
         "capitalize every word in this sentence",
         "java programming is fun"
      };

      // Capitalize the first letter of each word in the strings
      String[] capitalizedSentences = new String[sentences.length];
      for (int i = 0; i < sentences.length; i++) {
         String[] words = sentences[i].split("\\s+");
         StringBuilder sb = new StringBuilder();
         for (String word : words) {
            if (!word.isEmpty()) {

sb.append(Character.toUpperCase(word.charAt(0))).append(word.substring(1)).append(" ");
            }
         }
         capitalizedSentences[i] = sb.toString().trim();
      }

      // Display the original and capitalized sentences
      System.out.println("Original Sentences:");
      for (String sentence : sentences) {
         System.out.println(sentence);
      }

      System.out.println("\nCapitalized Sentences:");
      for (String capitalizedSentence : capitalizedSentences) {
         System.out.println(capitalizedSentence);
      }
   }
}
```

Week 5: Inheritance and packages

1. Design a class, which represents a patient. Fields of this class are name, age and hospital number.
Provide methods to input and display all fields. Next design a class called Inpatient, which inherits
from patient class. Provide fields to represent department name, admission date and room type.
Provide methods to input and display these fields. Next design a class called Billing, which inherits
from Inpatient class. Provide a field to represent the discharge date. Provide methods to input this
field value as well as to display the total amount. The total amount is calculated based on room type
and doctor charges as shown below:

```
import java.util.Scanner;

class Patient {
protected String name;
protected int age;
protected int hospitalNumber;

public void input() {
Scanner scanner = new Scanner(System.in);
System.out.println("Enter patient's name:");
name = scanner.nextLine();
System.out.println("Enter patient's age:");
age = scanner.nextInt();
System.out.println("Enter patient's hospital number:");
hospitalNumber = scanner.nextInt();
}

public void display() {
System.out.println("Patient Name: " + name);
System.out.println("Patient Age: " + age);
System.out.println("Hospital Number: " + hospitalNumber);
}
}

class Inpatient extends Patient {
private String departmentName;
private String admissionDate;
private String roomType;

public void input() {
super.input();
Scanner scanner = new Scanner(System.in);
```

```
System.out.println("Enter department name:");
departmentName = scanner.nextLine();
System.out.println("Enter admission date:");
admissionDate = scanner.nextLine();
System.out.println("Enter room type (Special/SemiSpecial/General):");
roomType = scanner.nextLine();
}

public void display() {
super.display();
System.out.println("Department Name: " + departmentName);
System.out.println("Admission Date: " + admissionDate);
System.out.println("Room Type: " + roomType);
}

public double calculateRoomCharges() {
double roomCharge = 0;
switch (roomType) {
case "Special":
roomCharge = 200;
break;
case "SemiSpecial":
roomCharge = 100;
break;
case "General":
roomCharge = 50;
break;
default:
System.out.println("Invalid room type entered.");
}
return roomCharge;
}
}

class Billing extends Inpatient {
private String dischargeDate;

public void input() {
super.input();
Scanner scanner = new Scanner(System.in);
System.out.println("Enter discharge date:");
dischargeDate = scanner.nextLine();
}

public void displayTotalAmount() {
super.display();
double roomCharge = calculateRoomCharges();
double consultancyCharge = 1000; // Assuming a fixed consultancy charge
System.out.println("Discharge Date: " + dischargeDate);
double totalAmount = roomCharge + consultancyCharge;
System.out.println("Total Amount: " + totalAmount);
}
}

public class HospitalManagement {
public static void main(String[] args) {
Billing billing = new Billing();
billing.input();
billing.displayTotalAmount();
}
}
```

2. Develop a set of methods, which work with an integer array. The methods to be implemented are:
i. min (which finds the minimum element in the array)
ii. max (which finds the maximum element in the array)
iii. sort (method to sort the array)
iv. reverse (method to reverse the array)
v. scale (method to multiply the array)
Place this in a package called p1. Let this package be present in a folder called "myPackages", which
is a folder in your present working directory (eg: c\student\4rthcse01\mypackages\p1). Write a main
method to use the methods of package p1. Try various other alternatives too.

```
package myPackages.p1;

import java.util.Arrays;

public class ArrayOperations {

   public static int min(int[] arr) {
      int min = arr[0];
      for (int i = 1; i < arr.length; i++) {
         if (arr[i] < min) {
            min = arr[i];
         }
      }
      return min;
   }

   public static int max(int[] arr) {
      int max = arr[0];
```

```
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }

    public static void sort(int[] arr) {
        Arrays.sort(arr);
    }

    public static void reverse(int[] arr) {
        int start = 0;
        int end = arr.length - 1;
        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }

    public static void scale(int[] arr, int factor) {
        for (int i = 0; i < arr.length; i++) {
            arr[i] *= factor;
        }
    }
}
```

Now, to use these methods in the `main` method:

```
import myPackages.p1.ArrayOperations;

public class Main {
    public static void main(String[] args) {
        int[] arr = {4, 2, 8, 1, 6, 10};

        // Find minimum
        int minValue = ArrayOperations.min(arr);
        System.out.println("Minimum value: " + minValue);

        // Find maximum
        int maxValue = ArrayOperations.max(arr);
        System.out.println("Maximum value: " + maxValue);
        // Sort array
        ArrayOperations.sort(arr);
        System.out.println("Sorted array: " + Arrays.toString(arr));

        // Reverse array
        ArrayOperations.reverse(arr);
        System.out.println("Reversed array: " + Arrays.toString(arr));

        // Scale array
        ArrayOperations.scale(arr, 2);
        System.out.println("Scaled array: " + Arrays.toString(arr));
    }
}
```

3. Create an abstract class Figure with abstract method area and two integer dimensions. Extend this
class to inherit three more classes Rectangle Triangle and Square which implements the area method.
Show how the area can be computed dynamically during run time for Rectangle, Square and Triangle.

```
abstract class Figure {
    abstract double area();
}

class Rectangle extends Figure {
    private int length;
    private int width;

    Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }

    @Override
    double area() {
        return length * width;
    }
}

class Triangle extends Figure {
    private int base;
    private int height;

    Triangle(int base, int height) {
        this.base = base;
        this.height = height;
    }

    @Override
    double area() {
        return 0.5 * base * height;
    }
}

class Square extends Figure {
    private int side;

    Square(int side) {
        this.side = side;
    }

    @Override
    double area() {
        return side * side;
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating objects and computing areas dynamically at runtime

        // Rectangle
        Figure rectangle = new Rectangle(4, 6);
        System.out.println("Area of Rectangle: " + rectangle.area());

        // Triangle
        Figure triangle = new Triangle(5, 8);
        System.out.println("Area of Triangle: " + triangle.area());

        // Square
        Figure square = new Square(7);
        System.out.println("Area of Square: " + square.area());
    }
}
```

4. Design a list class. (The previously designed one can be used) Make it a super class. Design a class,
which represents a double-ended list (List operations can be performed on both ends of the list). Let
this class inherit from the list class. Provide all other additional methods to this class. Show the usage

of this class in a main method

```
class List {
   protected int[] array;
   protected int size;
   protected int capacity;

   public List(int capacity) {
      this.capacity = capacity;
      this.array = new int[capacity];
      this.size = 0;
   }

   public void display() {
      System.out.print("List: ");
      for (int i = 0; i < size; i++) {
         System.out.print(array[i] + " ");
      }
      System.out.println();
   }

   public void insertAtEnd(int value) {
      if (size < capacity) {
         array[size] = value;
         size++;
      } else {
         System.out.println("List is full. Cannot insert more elements.");
      }
   }

   public void removeFromEnd() {
      if (size > 0) {
         size--;
      } else {
         System.out.println("List is empty. Cannot remove elements.");
      }
   }
}

class DoubleEndedList extends List {
   public DoubleEndedList(int capacity) {
      super(capacity);
   }

   public void insertAtBeginning(int value) {
      if (size < capacity) {
```

```java
            for (int i = size; i > 0; i--) {
                array[i] = array[i - 1];
            }
            array[0] = value;
            size++;
        } else {
            System.out.println("List is full. Cannot insert more elements.");
        }
    }

    public void removeFromBeginning() {
        if (size > 0) {
            for (int i = 0; i < size - 1; i++) {
                array[i] = array[i + 1];
            }
            size--;
        } else {
            System.out.println("List is empty. Cannot remove elements.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Using DoubleEndedList
        DoubleEndedList doubleEndedList = new DoubleEndedList(5);

        doubleEndedList.insertAtEnd(10);
        doubleEndedList.insertAtEnd(20);
        doubleEndedList.insertAtEnd(30);
        doubleEndedList.display();

        doubleEndedList.insertAtBeginning(5);
        doubleEndedList.display();

        doubleEndedList.removeFromEnd();
        doubleEndedList.display();

        doubleEndedList.removeFromBeginning();
        doubleEndedList.display();
    }
}
```

Week 6: Interfaces, Exception Handling.

1. Design an interface called Stack with 2 methods namely push() and pop() in it. Design a class called
FixedStack, which implements a fixed length version of the stack. Also design a class called DynamicStack, which implements a growable version of the stack. Write a main method, which uses
both these classes through an interface reference.

```
interface Stack {
    void push(int item);
    int pop();
}

class FixedStack implements Stack {
    private int[] stackArray;
    private int top;
    private int capacity;

    public FixedStack(int capacity) {
        this.capacity = capacity;
        this.stackArray = new int[capacity];
        this.top = -1;
    }

    @Override
    public void push(int item) {
        if (top == capacity - 1) {
            System.out.println("Stack Overflow");
        } else {
            stackArray[++top] = item;
            System.out.println(item + " pushed into FixedStack");
        }
    }

    @Override
    public int pop() {
        if (top == -1) {
            System.out.println("Stack Underflow");
            return -1;
        } else {
            int item = stackArray[top--];
            System.out.println(item + " popped from FixedStack");
            return item;
        }
    }
```

```java
}

class DynamicStack implements Stack {
    private int[] stackArray;
    private int top;
    private int capacity;

    public DynamicStack(int capacity) {
        this.capacity = capacity;
        this.stackArray = new int[capacity];
        this.top = -1;
    }

    @Override
    public void push(int item) {
        if (top == capacity - 1) {
            int[] newArray = new int[capacity * 2]; // doubling the capacity
            System.arraycopy(stackArray, 0, newArray, 0, capacity);
            stackArray = newArray;
            capacity *= 2;
        }
        stackArray[++top] = item;
        System.out.println(item + " pushed into DynamicStack");
    }

    @Override
    public int pop() {
        if (top == -1) {
            System.out.println("Stack Underflow");
            return -1;
        } else {
            int item = stackArray[top--];
            System.out.println(item + " popped from DynamicStack");
            return item;
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Stack fixedStack = new FixedStack(3);
        fixedStack.push(10);
        fixedStack.push(20);
        fixedStack.push(30);
        fixedStack.push(40); // This will cause Stack Overflow
        fixedStack.pop();
        fixedStack.pop();
        fixedStack.pop();
        fixedStack.pop(); // This will cause Stack Underflow

        Stack dynamicStack = new DynamicStack(3);
        dynamicStack.push(50);
        dynamicStack.push(60);
        dynamicStack.push(70);
        dynamicStack.push(80); // This will double the capacity
        dynamicStack.pop();
        dynamicStack.pop();
        dynamicStack.pop();
        dynamicStack.pop(); // This will cause Stack Underflow
    }
}
```

2. Design a class, which represents an employee. The data members are:
name (string), age (int), grossSalary (double), takeHomeSalary(float), grade (char). Provide methods called input() and display() which reads all details of a record from the keyboard and displays them respectively. Handle IOException while reading from the keyboard. Provide a menu with the options: Input, Display and Exit to read users choice. (Make use of Wrapper classes).

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Employee {
    private String name;
    private int age;
    private double grossSalary;
    private float takeHomeSalary;
    private char grade;

    public void input() {
        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter employee name:");
            name = reader.readLine();

            System.out.println("Enter employee age:");
            age = Integer.parseInt(reader.readLine());

            System.out.println("Enter employee gross salary:");
            grossSalary = Double.parseDouble(reader.readLine());

            System.out.println("Enter employee take-home salary:");
            takeHomeSalary = Float.parseFloat(reader.readLine());
            System.out.println("Enter employee grade:");
            grade = reader.readLine().charAt(0);
        } catch (IOException | NumberFormatException e) {
            System.out.println("Error reading input: " + e.getMessage());
        }
    }

    public void display() {
        System.out.println("Employee Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Gross Salary: " + grossSalary);
        System.out.println("Take-Home Salary: " + takeHomeSalary);
        System.out.println("Grade: " + grade);
    }

    public static void main(String[] args) {
        Employee employee = new Employee();
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Input");
            System.out.println("2. Display");
            System.out.println("3. Exit");
            System.out.println("Enter your choice:");

            try {
                int choice = Integer.parseInt(reader.readLine());
                switch (choice) {
                    case 1:
                        employee.input();
                        break;
                    case 2:
                        employee.display();
                        break;
                    case 3:
                        System.out.println("Exiting...");
                        return;
                    default:
                        System.out.println("Invalid choice. Please enter 1, 2, or 3.");
                }
            } catch (IOException | NumberFormatException e) {
                System.out.println("Error reading input: " + e.getMessage());
            }
        }
    }
}
```

3. Design a stack class. Provide your own stack exceptions namely push exception and pop exception,which throw exceptions when the stack is full and when the stack is empty respectively. Show the usage of these exceptions in handling a stack object in the main.

```
class PushException extends Exception {
    public PushException(String message) {
        super(message);
    }
}

class PopException extends Exception {
    public PopException(String message) {
        super(message);
    }
}

class Stack {
    private int[] stackArray;
    private int top;
    private int capacity;

    public Stack(int capacity) {
        this.capacity = capacity;
        this.stackArray = new int[capacity];
        this.top = -1;
    }

    public void push(int item) throws PushException {
        if (top == capacity - 1) {
            throw new PushException("Stack is full. Cannot push more elements.");
        } else {
            stackArray[++top] = item;
            System.out.println(item + " pushed into stack");
        }
    }

    public int pop() throws PopException {
        if (top == -1) {
            throw new PopException("Stack is empty. Cannot pop elements.");
        } else {
            int item = stackArray[top--];
            System.out.println(item + " popped from stack");
            return item;
        }
    }
}
```

```
public class Main {
   public static void main(String[] args) {
      Stack stack = new Stack(3);

      try {
         stack.push(10);
         stack.push(20);
         stack.push(30);
         stack.push(40); // This will throw PushException as the stack is full
      } catch (PushException e) {
         System.out.println("PushException: " + e.getMessage());
      }

      try {
         stack.pop();
         stack.pop();
         stack.pop();
         stack.pop(); // This will throw PopException as the stack is empty
      } catch (PopException e) {
         System.out.println("PopException: " + e.getMessage());
      }
   }
}
```

Week 7: Threads and Input/output

1. Write a java program to create and initialize a matrix of integers. Create n threads where n is equal to the number of rows in the matrix. Each of these threads should compute a distinct row sum. The main thread computes the complete sum by looking into the partial sums given by the threads.

```
class RowSumCalculator extends Thread {
   private int[] row;
   private int partialSum;

   public RowSumCalculator(int[] row) {
      this.row = row;
      this.partialSum = 0;
   }

   public int getPartialSum() {
      return partialSum;
   }

   @Override
   public void run() {
      for (int num : row) {
          partialSum += num;
      }
   }
}

public class MatrixRowSum {
   public static void main(String[] args) {
      int[][] matrix = {
              {1, 2, 3},
              {4, 5, 6},
              {7, 8, 9}
      };

      int numRows = matrix.length;
      RowSumCalculator[] rowSumCalculators = new RowSumCalculator[numRows];

      // Creating threads for each row to calculate partial sums
      for (int i = 0; i < numRows; i++) {
          rowSumCalculators[i] = new RowSumCalculator(matrix[i]);
          rowSumCalculators[i].start();
      }

      // Wait for all threads to complete
      try {
          for (RowSumCalculator calculator : rowSumCalculators) {
              calculator.join();
          }
      } catch (InterruptedException e) {
          System.out.println("Thread interrupted: " + e.getMessage());
      }

      // Calculating complete sum by combining partial sums
      int totalSum = 0;
      for (RowSumCalculator calculator : rowSumCalculators) {
          totalSum += calculator.getPartialSum();
      }

      System.out.println("Complete sum of matrix: " + totalSum);
   }
}
```

2. Write information of n employees to a file. This information should be read from the keyboard. The
data members of employee class are:

a. name (string), age (int), GrossSalary (double), TakeHomeSalary(float), Grade (char). (Each record can be stored on a separate line with a blank separating every 2 fields). Transfer the records with grade ‘A’ to another file. Also display those records on the screen.

```
import java.io.*;

class Employee {
   private String name;
   private int age;
   private double grossSalary;
   private float takeHomeSalary;
   private char grade;

   public Employee(String name, int age, double grossSalary, float takeHomeSalary, char
grade) {
       this.name = name;
       this.age = age;
       this.grossSalary = grossSalary;
       this.takeHomeSalary = takeHomeSalary;
       this.grade = grade;
   }

   public String getName() {
      return name;
   }

   public int getAge() {
      return age;
   }

   public double getGrossSalary() {
      return grossSalary;
   }

   public float getTakeHomeSalary() {
      return takeHomeSalary;
   }

   public char getGrade() {
      return grade;
   }
}

public class EmployeeRecords {
   public static void main(String[] args) {
      try {
         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
         FileWriter fileWriter = new FileWriter("employees.txt");
         BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

         int n;
         System.out.println("Enter the number of employees:");
         n = Integer.parseInt(reader.readLine());
         for (int i = 0; i < n; i++) {
            System.out.println("Enter details for employee " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = reader.readLine();
            System.out.print("Age: ");
            int age = Integer.parseInt(reader.readLine());
            System.out.print("Gross Salary: ");
            double grossSalary = Double.parseDouble(reader.readLine());
            System.out.print("Take-Home Salary: ");
            float takeHomeSalary = Float.parseFloat(reader.readLine());
            System.out.print("Grade: ");
            char grade = reader.readLine().charAt(0);

            // Writing employee details to file
            String employeeRecord = name + " " + age + " " + grossSalary + " " +
takeHomeSalary + " " + grade;
            bufferedWriter.write(employeeRecord);
            bufferedWriter.newLine();

            // Check if the grade is 'A' and write to another file
            if (grade == 'A') {
               FileWriter gradeAFileWriter = new FileWriter("gradeA_employees.txt", true);
               BufferedWriter gradeABufferedWriter = new
BufferedWriter(gradeAFileWriter);
               gradeABufferedWriter.write(employeeRecord);
               gradeABufferedWriter.newLine();
               gradeABufferedWriter.close();
            }
         }

         bufferedWriter.close();

         // Read and display records with grade 'A'
         System.out.println("\nRecords with Grade 'A':");
         BufferedReader gradeAReader = new BufferedReader(new
FileReader("gradeA_employees.txt"));
         String line;
         while ((line = gradeAReader.readLine()) != null) {
            System.out.println(line);
         }
         gradeAReader.close();

      } catch (IOException e) {
         System.out.println("IOException: " + e.getMessage());
      }
   }
}
```

2. Assume that there exists a file with different file names stored in it. Every file name is placed on a separate line. Create two threads, which scan the first half, and the second half of the file respectively, to search the filenames which end with .cpp. Write those file names onto the screen.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

class FileSearcher extends Thread {
   private String filePath;
   private int startLine;
   private int endLine;

   public FileSearcher(String filePath, int startLine, int endLine) {
      this.filePath = filePath;
      this.startLine = startLine;
      this.endLine = endLine;
   }

   @Override
   public void run() {
      try {
         BufferedReader reader = new BufferedReader(new FileReader(filePath));
         String line;
         int lineCount = 0;

         while ((line = reader.readLine()) != null && lineCount <= endLine) {
            if (lineCount >= startLine && line.endsWith(".cpp")) {
               System.out.println(line);
            }
            lineCount++;
         }
         reader.close();
      } catch (IOException e) {
         System.out.println("IOException: " + e.getMessage());
      }
   }
}

public class FileSearch {
   public static void main(String[] args) {
      String filePath = "file_names.txt"; // Replace with your file path

      try {
         BufferedReader reader = new BufferedReader(new FileReader(filePath));
         int totalLines = 0;

         // Count total number of lines in the file
         while (reader.readLine() != null) {
            totalLines++;
         }
         reader.close();

         int mid = totalLines / 2;

         // Create two threads to search the first and second halves of the file
         FileSearcher firstHalfSearcher = new FileSearcher(filePath, 0, mid - 1);
         FileSearcher secondHalfSearcher = new FileSearcher(filePath, mid, totalLines - 1);

         // Start the threads
         firstHalfSearcher.start();
         secondHalfSearcher.start();

      } catch (IOException e) {
         System.out.println("IOException: " + e.getMessage());
      }
   }
}
```

3. Count the number of single characters, numbers (sequence of 1 or more digits), words and lines from a file and place the result on the screen.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class FileStatistics {
    public static void main(String[] args) {
        String filePath = "your_file.txt"; // Replace with your file path

        int singleCharacters = 0;
        int numbers = 0;
        int words = 0;
        int lines = 0;

        try {
            BufferedReader reader = new BufferedReader(new FileReader(filePath));
            String line;

            while ((line = reader.readLine()) != null) {
                lines++;

                String[] wordsArray = line.split("\\s+"); // Split line into words using whitespace as
delimiter
                for (String word : wordsArray) {
                    if (word.length() == 1) {
                        singleCharacters++;
                    } else if (word.matches("\\d+")) {
```

```
                numbers++;
            }
        }

        words += wordsArray.length;
    }

    reader.close();

    System.out.println("Single Characters: " + singleCharacters);
    System.out.println("Numbers: " + numbers);
    System.out.println("Words: " + words);
    System.out.println("Lines: " + lines);

} catch (IOException e) {
    System.out.println("IOException: " + e.getMessage());
}
    }
}
```

5. Write a program to copy one file to other using
   I. char stream classes ii. Byte stream classes

Using Character Stream Classes (FileReader and FileWriter):

```
import java.io.*;



public class FileCopyCharacterStream {

    public static void main(String[] args) {

        String sourceFilePath = "source.txt"; // Replace with your source file path

        String destinationFilePath = "destination_char.txt"; // Replace with your destination file
path



        try (FileReader fileReader = new FileReader(sourceFilePath);

             FileWriter fileWriter = new FileWriter(destinationFilePath)) {



            int c;
            while ((c = fileReader.read()) != -1) {
                fileWriter.write(c);
            }
            System.out.println("File copied successfully using character streams.");

        } catch (IOException e) {
            System.out.println("IOException: " + e.getMessage());
        }
    }
}
```

Using Byte Stream Classes (FileInputStream and FileOutputStream):

```
import java.io.*;


public class FileCopyByteStream {
    public static void main(String[] args) {
        String sourceFilePath = "source.txt"; // Replace with your source file path
        String destinationFilePath = "destination_byte.txt"; // Replace with your destination file
path

        try (FileInputStream inputStream = new FileInputStream(sourceFilePath);
             FileOutputStream outputStream = new FileOutputStream(destinationFilePath)) {

            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
```

```
            outputStream.write(buffer, 0, bytesRead);

        }

        System.out.println("File copied successfully using byte streams.");



    } catch (IOException e) {

        System.out.println("IOException: " + e.getMessage());

    }

  }

}
```

Week 8: Applets and Event Handling

1. Write an applet to display your biodata in the middle. Have suitable choice of background and foreground colors.

```
import java.applet.*;
import java.awt.*;
public class BioDataApplet extends Applet {
   public void init() {
      setBackground(Color.LIGHT_GRAY);
      setForeground(Color.BLUE);
   }

   public void paint(Graphics g) {
      String name = "Your Name";
      String age = "Your Age";
      String profession = "Your Profession";
      String email = "Your Email";

      Font font = new Font("Arial", Font.BOLD, 16);
      g.setFont(font);

      FontMetrics fm = g.getFontMetrics();
      int x = (getWidth() - fm.stringWidth(name)) / 2;
      int y = (getHeight() - fm.getHeight() * 4) / 2;

      g.drawString("Name: " + name, x, y);
      g.drawString("Age: " + age, x, y + fm.getHeight());
      g.drawString("Profession: " + profession, x, y + 2 * fm.getHeight());
      g.drawString("Email: " + email, x, y + 3 * fm.getHeight());
   }
}
```

To view this applet, you'll need to embed it in an HTML file or run it using an applet viewer. Here's an example HTML code to embed the applet:

```
<html>
<head>
   <title>BioData Applet</title>
</head>
<body>
   <applet code="BioDataApplet.class" width="400" height="300">
      Your browser does not support Java applets.
   </applet>
</body>
</html>
```

2. Develop an applet which scrolls a message “Java Programming”, horizontally from left to right across the applet window. Set the background and foreground colors of the banner to cyan and red respectively. Also, in the status window display a message “this is the status window”. Modify the banner applet such that the message to be scrolled is read from the user.

```
import java.applet.Applet;
import java.awt.*;

public class ScrollingBanner extends Applet implements Runnable {
   private String bannerMessage = "Java Programming"; // Default message
   private int xCoordinate = 0;
   private Thread thread;

   public void init() {
      setBackground(Color.CYAN);
      setForeground(Color.RED);
      showStatus("This is the status window");
   }

   public void start() {
      if (thread == null) {
         thread = new Thread(this);
         thread.start();
      }
   }

   public void stop() {
      if (thread != null) {
         thread = null;
      }
   }

   public void run() {
      while (true) {
            repaint();
            try {
                Thread.sleep(100); // Adjust the speed of scrolling by changing the sleep duration
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted: " + e.getMessage());
            }
        }
    }

    public void update(Graphics g) {
        g.clearRect(0, 0, getSize().width, getSize().height);
        g.drawString(bannerMessage, xCoordinate, 50);
        xCoordinate++;
        if (xCoordinate > getSize().width) {
            xCoordinate = 0;
        }
    }

    public boolean mouseDown(Event event, int x, int y) {
        String input = showInputDialog("Enter your message:");
        if (input != null && !input.isEmpty()) {
            bannerMessage = input;
        }
        return true;
    }
}
```

3. Write a Java program to demonstrate the mouse event handlers.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MouseEventDemo extends JFrame implements MouseListener,
MouseMotionListener {
    private JLabel statusLabel;

    public MouseEventDemo() {
        setTitle("Mouse Event Demo");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        statusLabel = new JLabel("No mouse event yet.");
        add(statusLabel, BorderLayout.SOUTH);

        addMouseListener(this);
        addMouseMotionListener(this);
    }

    public void mouseClicked(MouseEvent e) {
        statusLabel.setText("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mousePressed(MouseEvent e) {
        statusLabel.setText("Mouse Pressed at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mouseReleased(MouseEvent e) {
        statusLabel.setText("Mouse Released at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mouseEntered(MouseEvent e) {
        statusLabel.setText("Mouse Entered at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mouseExited(MouseEvent e) {
        statusLabel.setText("Mouse Exited at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mouseDragged(MouseEvent e) {
        statusLabel.setText("Mouse Dragged at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mouseMoved(MouseEvent e) {
        statusLabel.setText("Mouse Moved at (" + e.getX() + ", " + e.getY() + ")");
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            MouseEventDemo mouseEventDemo = new MouseEventDemo();
            mouseEventDemo.setVisible(true);
        });
    }
}
```

4. Write a Java program to demonstrate the keyboard event handlers.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class KeyboardEventDemo extends JFrame implements KeyListener {
    private JLabel statusLabel;

    public KeyboardEventDemo() {
        setTitle("Keyboard Event Demo");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        statusLabel = new JLabel("No key event yet.");
        add(statusLabel, BorderLayout.SOUTH);

        addKeyListener(this);
        setFocusable(true);
    }

    public void keyTyped(KeyEvent e) {
        statusLabel.setText("Key Typed: " + e.getKeyChar());
    }

    public void keyPressed(KeyEvent e) {
        statusLabel.setText("Key Pressed: " + KeyEvent.getKeyText(e.getKeyCode()));
    }

    public void keyReleased(KeyEvent e) {
        statusLabel.setText("Key Released: " +
KeyEvent.getKeyText(e.getKeyCode()));
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            KeyboardEventDemo keyboardEventDemo = new KeyboardEventDemo();
            keyboardEventDemo.setVisible(true);
        });
    }
}
```

<u>Week 9: AWT</u>

1. Write Java programs to demonstrate the usage of following AWT controls:
i. Buttons, Check Boxes,
ii. Checkbox Group,
iii. Combo Boxes,
iv. Radio Buttons,
v. Lists, Text Field,
vi. Text Area,
vii. Tabbed Panes,
viii. Scroll Panes.

Buttons, Check Boxes

```
import java.awt.*;

import java.awt.event.*;

public class ButtonCheckBoxExample {

    public static void main(String[] args) {
        Frame frame = new Frame("Button and Checkbox Example");
        Button button = new Button("Click me!");
        Checkbox checkbox = new Checkbox("Check me!");
        frame.add(button, BorderLayout.NORTH);
        frame.add(checkbox, BorderLayout.SOUTH);
        frame.setSize(300, 200);
        frame.setVisible(true);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button Clicked!");
            }
        });

        checkbox.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                if (checkbox.getState()) {
                    System.out.println("Checkbox checked!");
                } else {
                    System.out.println("Checkbox unchecked!");
                }
            }
        });
    }
}
```

Checkbox Group

```
import java.awt.*;
import java.awt.event.*;

public class CheckboxGroupExample {
    public static void main(String[] args) {
        Frame frame = new Frame("Checkbox Group Example");
        CheckboxGroup checkboxGroup = new CheckboxGroup();

        Checkbox checkbox1 = new Checkbox("Option 1", checkboxGroup, false);
        Checkbox checkbox2 = new Checkbox("Option 2", checkboxGroup, false);

        frame.add(checkbox1);
        frame.add(checkbox2);

        frame.setLayout(new FlowLayout());
        frame.setSize(300, 200);
        frame.setVisible(true);

        checkbox1.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                System.out.println("Option 1 selected!");
            }
        });

        checkbox2.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                System.out.println("Option 2 selected!");
            }
        });
    }
}
```

Combo Boxes

```
import java.awt.*;
import java.awt.event.*;

public class ComboBoxExample {
    public static void main(String[] args) {
        Frame frame = new Frame("Combo Box Example");

        String[] items = {"Option 1", "Option 2", "Option 3"};
        JComboBox<String> comboBox = new JComboBox<>(items);

        frame.add(comboBox);

        frame.setLayout(new FlowLayout());
        frame.setSize(300, 200);
        frame.setVisible(true);

        comboBox.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JComboBox cb = (JComboBox) e.getSource();
                String selectedOption = (String) cb.getSelectedItem();
                System.out.println("Selected: " + selectedOption);
            }
        });
    }
}
```

Radio Buttons

```
import java.awt.*;

import java.awt.event.*;



public class RadioButtonExample {

    public static void main(String[] args) {

        Frame frame = new Frame("Radio Button Example");



        CheckboxGroup radioGroup = new CheckboxGroup();

        Checkbox radioButton1 = new Checkbox("Option 1", radioGroup, false);

        Checkbox radioButton2 = new Checkbox("Option 2", radioGroup, false);



        frame.add(radioButton1);

        frame.add(radioButton2);
        frame.setLayout(new FlowLayout());
        frame.setSize(300, 200);
        frame.setVisible(true);

        radioButton1.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                System.out.println("Option 1 selected!");
            }
        });

        radioButton2.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                System.out.println("Option 2 selected!");
            }
        });
    }
}
```

Lists, Text Field

```
import java.awt.*;
import java.awt.event.*;

public class ListTextFieldExample {
    public static void main(String[] args) {
        Frame frame = new Frame("List and Text Field Example");
        List list = new List(4); // Create a list with 4 visible rows
        list.add("Item 1");
        list.add("Item 2");
        list.add("Item 3");
        TextField textField = new TextField(20); // Create a text field
        Button button = new Button("Add");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String item = textField.getText();
                list.add(item);
                textField.setText("");
            }
        });
        frame.add(list, BorderLayout.WEST);
        frame.add(textField, BorderLayout.CENTER);
        frame.add(button, BorderLayout.EAST);
        frame.setSize(400, 200);
        frame.setVisible(true);
    }
}
```

Text Area

```
import java.awt.*;
public class TextAreaExample {
    public static void main(String[] args) {
        Frame frame = new Frame("Text Area Example");
        TextArea textArea = new TextArea("This is a text area", 10, 30); // Create a text area
        frame.add(textArea);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

Tabbed Panes

```
import java.awt.*;
public class TabbedPaneExample {
    public static void main(String[] args) {
        Frame frame = new Frame("Tabbed Pane Example");
        Panel panel1 = new Panel();
        panel1.add(new Label("Content of Tab 1"));
        Panel panel2 = new Panel();
        panel2.add(new Label("Content of Tab 2"));
        TabbedPane tabbedPane = new TabbedPane();
        tabbedPane.add("Tab 1", panel1);
        tabbedPane.add("Tab 2", panel2);
        frame.add(tabbedPane);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

Scroll Panes

```
import java.awt.*;

public class ScrollPaneExample {

   public static void main(String[] args) {

      Frame frame = new Frame("Scroll Pane Example");

      TextArea textArea = new TextArea(10, 30); // Create a large text area

      ScrollPane scrollPane = new ScrollPane();

      scrollPane.add(textArea); // Add text area to scroll pane

      frame.add(scrollPane);

      frame.setSize(300, 200);

      frame.setVisible(true);

   }

}
```

2. Write a Java program to demonstrate the usage of Menu Bar and Menus. Use Dialog Boxes to display the Menu options selected.

```
import java.awt.*;
import java.awt.event.*;

public class MenuBarExample {
   public static void main(String[] args) {
      Frame frame = new Frame("Menu Bar Example");

      // Create a menu bar
      MenuBar menuBar = new MenuBar();

      // Create a menu
      Menu menu = new Menu("Options");

      // Create menu items
      MenuItem menuItem1 = new MenuItem("Option 1");
```

```
        MenuItem menuItem2 = new MenuItem("Option 2");
        MenuItem menuItem3 = new MenuItem("Option 3");

        // Add menu items to the menu
        menu.add(menuItem1);
        menu.add(menuItem2);
        menu.add(menuItem3);

        // Add the menu to the menu bar
        menuBar.add(menu);

        // Set the menu bar to the frame
        frame.setMenuBar(menuBar);

        // Add action listeners to menu items
        menuItem1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Dialog dialog = new Dialog(frame, "Option 1 selected", true);
                dialog.setSize(200, 100);
                dialog.setVisible(true);
            }
        });

        menuItem2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Dialog dialog = new Dialog(frame, "Option 2 selected", true);
                dialog.setSize(200, 100);
                dialog.setVisible(true);
            }
        });

        menuItem3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Dialog dialog = new Dialog(frame, "Option 3 selected", true);
                dialog.setSize(200, 100);
                dialog.setVisible(true);
            }
        });

        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

2. Write Java program that allows the user to draw a rectangle by dragging the mouse on the application window. The upper-left coordinate should be the location where the user presses the mouse button, and the lower-right coordinate should be the location where the user releases the mouse button. Also display the area of the rectangle.

```
import java.awt.*;
import java.awt.event.*;

public class RectangleDrawing extends Frame {
   private int startX, startY, endX, endY;
   private boolean isDrawing = false;

   public RectangleDrawing() {
      setTitle("Rectangle Drawing");
      setSize(400, 400);

      addMouseListener(new MouseAdapter() {
         public void mousePressed(MouseEvent e) {
            startX = e.getX();
            startY = e.getY();
            isDrawing = true;
         }

         public void mouseReleased(MouseEvent e) {
            endX = e.getX();
            endY = e.getY();
            isDrawing = false;

            int width = Math.abs(endX - startX);
            int height = Math.abs(endY - startY);
            int area = width * height;

            System.out.println("Area of the rectangle: " + area);
         }
      });

      addMouseMotionListener(new MouseMotionAdapter() {
         public void mouseDragged(MouseEvent e) {
            if (isDrawing) {
               endX = e.getX();
               endY = e.getY();
               repaint();
            }
         }
      });

      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            dispose();
         }
      });
   }

   public void paint(Graphics g) {
        if (isDrawing) {
            int width = Math.abs(endX - startX);
            int height = Math.abs(endY - startY);

            int upperLeftX = Math.min(startX, endX);
            int upperLeftY = Math.min(startY, endY);

            g.drawRect(upperLeftX, upperLeftY, width, height);
        }
    }

    public static void main(String[] args) {
        RectangleDrawing app = new RectangleDrawing();
        app.setVisible(true);
    }
}
```

Week 10: AWT and Swings

1. Write a Java program to implement a simple calculator.

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double num1, num2, result;
        char operator;

        System.out.print("Enter first number: ");
        num1 = scanner.nextDouble();

        System.out.print("Enter operator (+, -, *, /): ");
        operator = scanner.next().charAt(0);

        System.out.print("Enter second number: ");
        num2 = scanner.nextDouble();

        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 == 0) {
                    System.out.println("Error! Division by zero is not allowed.");
                    return;
                }
                result = num1 / num2;
                break;
            default:
                System.out.println("Error! Invalid operator.");
                return;
        }

        System.out.println("Result: " + result);
    }
}
```

2 . Write an applet to test username and password.

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UserLoginApplet extends Applet implements ActionListener {
    TextField usernameField, passwordField;
    Button loginButton;
    Label statusLabel;

    public void init() {
        usernameField = new TextField(20);
        passwordField = new TextField(20);
        passwordField.setEchoChar('*'); // Setting echo char to mask password

        Label usernameLabel = new Label("Username:");
        Label passwordLabel = new Label("Password:");
        statusLabel = new Label("");

        loginButton = new Button("Login");
        loginButton.addActionListener(this);

        add(usernameLabel);
        add(usernameField);
        add(passwordLabel);
        add(passwordField);
        add(loginButton);
        add(statusLabel);
    }

    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        String password = passwordField.getText();

        // Check username and password
        if (username.equals("admin") && password.equals("password")) {
            statusLabel.setText("Login successful!");
        } else {
            statusLabel.setText("Invalid username or password.");
        }
    }
}
```

3. Write an applet that obtains two floating point numbers from the user and displays the sum, product, difference and quotient of these numbers.

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CalculatorApplet extends Applet implements ActionListener {
    TextField num1Field, num2Field;
    Button calculateButton;
    Label sumLabel, productLabel, differenceLabel, quotientLabel;

    public void init() {
        num1Field = new TextField(10);
        num2Field = new TextField(10);

        Label num1Label = new Label("Enter first number:");
        Label num2Label = new Label("Enter second number:");
        sumLabel = new Label("Sum: ");
        productLabel = new Label("Product: ");
        differenceLabel = new Label("Difference: ");
        quotientLabel = new Label("Quotient: ");

        calculateButton = new Button("Calculate");
        calculateButton.addActionListener(this);

        add(num1Label);
        add(num1Field);
        add(num2Label);
        add(num2Field);
        add(calculateButton);
        add(sumLabel);
        add(productLabel);
        add(differenceLabel);
        add(quotientLabel);
    }

    public void actionPerformed(ActionEvent e) {
        double num1, num2, sum, product, difference, quotient;
        num1 = Double.parseDouble(num1Field.getText());
        num2 = Double.parseDouble(num2Field.getText());

        sum = num1 + num2;
        product = num1 * num2;
        difference = num1 - num2;
        quotient = num1 / num2;

        sumLabel.setText("Sum: " + sum);
        productLabel.setText("Product: " + product);
        differenceLabel.setText("Difference: " + difference);

        if(Double.isInfinite(quotient) || Double.isNaN(quotient)) {
            quotientLabel.setText("Cannot divide by zero or invalid input.");
        } else {
            quotientLabel.setText("Quotient: " + quotient);
        }
    }
}
```

Week 11: JDBC

Create and populate a simple student Database that contains name, password and other details using MS ACCESS. (Use windowed applications of java)

1. Write a Java program which runs the user supplied query on student database.

```
import java.sql.*;

public class AccessDatabaseQuery {
    static final String JDBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String DATABASE_URL = "jdbc:odbc:StudentDB"; // Change 'StudentDB' to
your Access database name

    public static void main(String[] args) {
        Connection connection = null;
        Statement statement = null;

        try {
            Class.forName(JDBC_DRIVER);
            connection = DriverManager.getConnection(DATABASE_URL);

            statement = connection.createStatement();
            // User-supplied query
            String userQuery = "SELECT * FROM Students"; // Example query, change as
needed

            ResultSet resultSet = statement.executeQuery(userQuery);

            // Display query results
            while (resultSet.next()) {
                String name = resultSet.getString("Name");
                // Retrieve other details as needed based on column names in the database
                // Example: String password = resultSet.getString("Password");

                System.out.println("Name: " + name);
                // Display other details similarly
            }

            resultSet.close();
            statement.close();
            connection.close();
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            try {
                if (statement != null)
                    statement.close();
            } catch (SQLException se2) {
                // Ignored
            }
            try {
                if (connection != null)
                    connection.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
        }
    }
}
```

2. Write a menu driven Java program to do the following operations on the student database:
a) Insert a new record.
b) Display the specified records.
c) Delete the specified records.
d) Update the specified records.
e) Exit.

```
import java.sql.*;
import java.util.Scanner;

public class StudentDatabaseOperations {
  static final String JDBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
  static final String DATABASE_URL = "jdbc:odbc:StudentDB"; // Change 'StudentDB' to
your Access database name

  public static void main(String[] args) {
    Connection connection = null;
    Statement statement = null;
    Scanner scanner = new Scanner(System.in);

    try {
      Class.forName(JDBC_DRIVER);
      connection = DriverManager.getConnection(DATABASE_URL);
      statement = connection.createStatement();

      while (true) {
        System.out.println("\nMenu:");
        System.out.println("a) Insert a new record");
        System.out.println("b) Display specified records");
        System.out.println("c) Delete specified records");
        System.out.println("d) Update specified records");
        System.out.println("e) Exit");
        System.out.print("Enter your choice: ");
        String choice = scanner.nextLine();

        switch (choice.toLowerCase()) {
          case "a":
            // Insert a new record
            // Example: insertRecord(statement);
            break;
          case "b":
            // Display specified records
            // Example: displayRecords(statement);
            break;
          case "c":
            // Delete specified records
            // Example: deleteRecords(statement);
            break;
          case "d":
            // Update specified records
            // Example: updateRecords(statement);
            break;
          case "e":
            System.out.println("Exiting program...");
            statement.close();
            connection.close();
            scanner.close();
```

```
                    return;
                default:
                    System.out.println("Invalid choice. Please select again.");
            }
        }
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        try {
            if (statement != null)
                statement.close();
        } catch (SQLException se2) {
            // Ignored
        }
        try {
            if (connection != null)
                connection.close();
        } catch (SQLException se) {
            se.printStackTrace();
        }
        if (scanner != null)
            scanner.close();
    }
}


}
```