SET A 1. Write a java program for Solving problems by using sequential search.

Certainly! Below is an example of a Java program that demonstrates sequential search (also known as linear search). This program will search for a specific element in an array of integers.

```
java
Copy code
public class SequentialSearch {

    // Method to perform sequential search
    public static int sequentialSearch(int[] arr, int target) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == target) {
                return i;  // Return the index of the target element
            }
        }
        return -1;  // Return -1 if the target is not found
    }

    public static void main(String[] args) {
        // Sample array
        int[] numbers = {5, 3, 8, 4, 2, 9, 7};

        // Target element to search for
        int target = 4;

        // Perform sequential search
        int result = sequentialSearch(numbers, target);

        // Output the result
        if (result != -1) {
            System.out.println("Element found at index: " + result);
        } else {
            System.out.println("Element not found in the array.");
        }
    }
}
```

Explanation:

1. **`sequentialSearch` method:** This method takes an array of integers (`arr`) and a target integer (`target`) as inputs. It iterates through the array and checks if each element matches the target. If a match is found, it returns the index of the element. If no match is found after checking all elements, it returns -1.
2. **`main` method:** This is the entry point of the program. It defines a sample array `numbers` and a target element `target`. It then calls the `sequentialSearch` method and prints the result, indicating whether the target element was found and, if so, its index.

You can compile and run this program using any Java development environment or command line to see how it works.

4o

2. Write a java program to search a number using binary search algorithm.

Sure! Below is a Java program that demonstrates the binary search algorithm. Note that for binary search to work, the array must be sorted.

```
java
Copy code
import java.util.Arrays;

public class BinarySearch {

    // Method to perform binary search
    public static int binarySearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if target is present at mid
            if (arr[mid] == target) {
                return mid;
            }

            // If target greater, ignore left half
            if (arr[mid] < target) {
                left = mid + 1;
            }
            // If target is smaller, ignore right half
            else {
                right = mid - 1;
            }
        }

        // Target is not present in array
        return -1;
    }

    public static void main(String[] args) {
        // Sample sorted array
        int[] numbers = {2, 3, 4, 7, 8, 9, 10};

        // Target element to search for
        int target = 7;

        // Perform binary search
        int result = binarySearch(numbers, target);

        // Output the result
        if (result != -1) {
            System.out.println("Element found at index: " + result);
        } else {
            System.out.println("Element not found in the array.");
        }
    }
}
```

Explanation:

1. **`binarySearch` method:** This method takes a sorted array of integers (`arr`) and a target integer (`target`) as inputs. It uses two pointers, `left` and `right`, to keep track of the current search range. It calculates the middle index `mid` and compares the middle element with the target. Depending on the comparison, it adjusts the search range by moving either the `left` or `right` pointer. The loop continues until the target is found or the search range is exhausted.

2. **`main` method:** This is the entry point of the program. It defines a sample sorted array `numbers` and a target element `target`. It then calls the `binarySearch` method and prints the result, indicating whether the target element was found and, if so, its index.

To run this program, ensure that the array is sorted, as binary search requires a sorted array to function correctly. You can compile and execute the program using any Java development environment or command line.

4o
3. Write a java program to sort list of numbers using Selection Sort (Quadratic sorting) algorithm.

Sure! Below is a Java program that demonstrates the Selection Sort algorithm, which is a quadratic sorting algorithm.

```
java
Copy code
public class SelectionSort {

    // Method to perform selection sort
    public static void selectionSort(int[] arr) {
        int n = arr.length;

        // One by one move the boundary of the unsorted subarray
        for (int i = 0; i < n - 1; i++) {
            // Find the minimum element in the unsorted array
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }

            // Swap the found minimum element with the first element
            int temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;
        }
    }

    public static void main(String[] args) {
        // Sample array
        int[] numbers = {64, 25, 12, 22, 11};

        // Print array before sorting
        System.out.println("Array before sorting:");
        for (int num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println();

        // Perform selection sort
        selectionSort(numbers);

        // Print array after sorting
        System.out.println("Array after sorting:");
        for (int num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Explanation:

1. **`selectionSort` method:** This method takes an array of integers (`arr`) as input and sorts it using the Selection Sort algorithm. It iterates through the array, finds the minimum element in the unsorted portion of the array, and swaps it with the first unsorted element. This process repeats for each element until the entire array is sorted.
2. **`main` method:** This is the entry point of the program. It defines a sample array `numbers`, prints the array before sorting, calls the `selectionSort` method to sort the array, and then prints the array after sorting.

To run this program, compile and execute it using any Java development environment or command line. The program will demonstrate the Selection Sort algorithm by sorting the provided sample array.

4o
4. Write a java program to sort list of numbers using insertion Sort (Quadratic sorting) algorithm.

Certainly! Below is a Java program that demonstrates the Insertion Sort algorithm, which is another quadratic sorting algorithm.

```
java
Copy code
public class InsertionSort {

    // Method to perform insertion sort
    public static void insertionSort(int[] arr) {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;

            // Move elements of arr[0..i-1], that are greater than key,
            // to one position ahead of their current position
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    public static void main(String[] args) {
        // Sample array
        int[] numbers = {12, 11, 13, 5, 6};

        // Print array before sorting
        System.out.println("Array before sorting:");
        for (int num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println();

        // Perform insertion sort
        insertionSort(numbers);

        // Print array after sorting
        System.out.println("Array after sorting:");
        for (int num : numbers) {
            System.out.print(num + " ");
```

```
        }
        System.out.println();
    }
}
```

Explanation:

1. **`insertionSort` method:** This method takes an array of integers (`arr`) as input and sorts it using the Insertion Sort algorithm. It iterates through the array starting from the second element (index 1). For each element, it compares it with the elements before it and shifts the larger elements one position to the right to make space for the current element (`key`). This process continues until the entire array is sorted.
2. **`main` method:** This is the entry point of the program. It defines a sample array `numbers`, prints the array before sorting, calls the `insertionSort` method to sort the array, and then prints the array after sorting.

To run this program, compile and execute it using any Java development environment or command line. The program will demonstrate the Insertion Sort algorithm by sorting the provided sample array.

4o
5. Write a java program to demonstrate Stack data structure using classes and objects.

Certainly! Below is a Java program that demonstrates a Stack data structure using classes and objects.

```
java
Copy code
// Define the Stack class
class Stack {
    private int maxSize;
    private int[] stackArray;
    private int top;

    // Constructor to initialize the stack
    public Stack(int size) {
        maxSize = size;
        stackArray = new int[maxSize];
        top = -1;  // indicates an empty stack
    }

    // Method to add an element to the stack
    public void push(int value) {
        if (isFull()) {
            System.out.println("Stack is full. Cannot push " + value);
        } else {
            stackArray[++top] = value;
            System.out.println("Pushed " + value + " onto the stack.");
        }
    }

    // Method to remove an element from the stack
    public int pop() {
        if (isEmpty()) {
            System.out.println("Stack is empty. Cannot pop.");
            return -1;
        } else {
```

```
            int value = stackArray[top--];
            System.out.println("Popped " + value + " from the stack.");
            return value;
        }
    }

    // Method to peek at the top element of the stack
    public int peek() {
        if (isEmpty()) {
            System.out.println("Stack is empty. Nothing to peek.");
            return -1;
        } else {
            return stackArray[top];
        }
    }

    // Method to check if the stack is empty
    public boolean isEmpty() {
        return (top == -1);
    }

    // Method to check if the stack is full
    public boolean isFull() {
        return (top == maxSize - 1);
    }
}

public class StackDemo {
    public static void main(String[] args) {
        // Create a stack of size 5
        Stack stack = new Stack(5);

        // Perform stack operations
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.push(50);

        // Attempt to push when stack is full
        stack.push(60);

        // Peek at the top element
        System.out.println("Top element is: " + stack.peek());

        // Pop elements from the stack
        stack.pop();
        stack.pop();

        // Peek at the top element again
        System.out.println("Top element is: " + stack.peek());

        // Pop remaining elements
        stack.pop();
        stack.pop();
        stack.pop();

        // Attempt to pop when stack is empty
        stack.pop();
    }
}
```

Explanation:

1. **`Stack` class:** This class defines a stack data structure with methods to push, pop, peek, check if the stack is empty, and check if the stack is full.
    - **Constructor:** Initializes the stack with a given size.
    - **`push` method:** Adds an element to the stack if it's not full.
    - **`pop` method:** Removes and returns the top element from the stack if it's not empty.
    - **`peek` method:** Returns the top element of the stack without removing it.
    - **`isEmpty` method:** Checks if the stack is empty.
    - **`isFull` method:** Checks if the stack is full.
2. **`StackDemo` class:** This is the main class that demonstrates the usage of the `Stack` class.
    - It creates a stack of size 5 and performs various stack operations like push, pop, and peek.
    - It also demonstrates edge cases like attempting to push when the stack is full and attempting to pop when the stack is empty.

To run this program, compile and execute it using any Java development environment or command line. The program will demonstrate the basic operations of a stack data structure using classes and objects.

4o
6. Write a java program to demonstrate queue data structure using classes and objects. 7. Write java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it’s not a duplicate of any number already read display the complete set of unique values input after the user enters each new value 8. Write a java program to create a super class called Figure that receives the dimensions of two dimensional objects. It also defines a method called area that computes the area of an object. The program derives two subclasses from Figure. The first is Rectangle and second is Triangle. Each of the sub class overridden area() so that it returns the area of a rectangle and a triangle respectively Implement a program that finds the largest element in an array of integers

6. Demonstrating Queue Data Structure Using Classes and Objects

Below is a Java program that demonstrates a Queue data structure using classes and objects.

```
java
Copy code
class Queue {
    private int maxSize;
    private int[] queueArray;
    private int front;
    private int rear;
    private int nItems;

    // Constructor to initialize the queue
    public Queue(int size) {
        maxSize = size;
        queueArray = new int[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }
```

```
    // Method to add an element to the queue
    public void enqueue(int value) {
        if (isFull()) {
            System.out.println("Queue is full. Cannot enqueue " + value);
        } else {
            if (rear == maxSize - 1) {
                rear = -1;
            }
            queueArray[++rear] = value;
            nItems++;
            System.out.println("Enqueued " + value + " to the queue.");
        }
    }

    // Method to remove an element from the queue
    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty. Cannot dequeue.");
            return -1;
        } else {
            int temp = queueArray[front++];
            if (front == maxSize) {
                front = 0;
            }
            nItems--;
            System.out.println("Dequeued " + temp + " from the queue.");
            return temp;
        }
    }

    // Method to check if the queue is empty
    public boolean isEmpty() {
        return (nItems == 0);
    }

    // Method to check if the queue is full
    public boolean isFull() {
        return (nItems == maxSize);
    }

    // Method to peek at the front element of the queue
    public int peek() {
        if (isEmpty()) {
            System.out.println("Queue is empty. Nothing to peek.");
            return -1;
        } else {
            return queueArray[front];
        }
    }
}

public class QueueDemo {
    public static void main(String[] args) {
        // Create a queue of size 5
        Queue queue = new Queue(5);

        // Perform queue operations
        queue.enqueue(10);
        queue.enqueue(20);
        queue.enqueue(30);
        queue.enqueue(40);
        queue.enqueue(50);
```

```
        // Attempt to enqueue when queue is full
        queue.enqueue(60);

        // Peek at the front element
        System.out.println("Front element is: " + queue.peek());

        // Dequeue elements from the queue
        queue.dequeue();
        queue.dequeue();

        // Peek at the front element again
        System.out.println("Front element is: " + queue.peek());

        // Dequeue remaining elements
        queue.dequeue();
        queue.dequeue();
        queue.dequeue();

        // Attempt to dequeue when queue is empty
        queue.dequeue();
    }
}
```

7. Java Program to Input 5 Numbers and Display Unique Values

Below is a Java program that inputs 5 numbers (each between 10 and 100 inclusive) and displays the complete set of unique values input after each new value is entered.

```
java
Copy code
import java.util.HashSet;
import java.util.Scanner;

public class UniqueNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashSet<Integer> uniqueNumbers = new HashSet<>();

        System.out.println("Enter 5 numbers between 10 and 100:");

        for (int i = 0; i < 5; i++) {
            int number = scanner.nextInt();

            if (number < 10 || number > 100) {
                System.out.println("Number out of range. Please enter a number
between 10 and 100.");
                i--; // retry this iteration
                continue;
            }

            if (!uniqueNumbers.contains(number)) {
                uniqueNumbers.add(number);
                System.out.println("Number " + number + " added to the set of
unique numbers.");
            } else {
                System.out.println("Number " + number + " is a duplicate and was
not added.");
            }

            // Display the complete set of unique numbers
            System.out.println("Unique numbers so far: " + uniqueNumbers);
        }
```

```
        scanner.close();
    }
}
```

8. Java Program to Create a Superclass Called Figure and Subclasses Rectangle and Triangle

Below is a Java program that creates a superclass `Figure` and subclasses `Rectangle` and `Triangle`, each with an overridden `area` method. Additionally, it includes a method to find the largest element in an array of integers.

```
java
Copy code
abstract class Figure {
    double dimension1;
    double dimension2;

    // Constructor to initialize dimensions
    public Figure(double dim1, double dim2) {
        dimension1 = dim1;
        dimension2 = dim2;
    }

    // Abstract method to compute area
    abstract double area();
}

class Rectangle extends Figure {
    // Constructor to initialize dimensions for rectangle
    public Rectangle(double length, double width) {
        super(length, width);
    }

    // Override area method to compute area of rectangle
    @Override
    double area() {
        return dimension1 * dimension2;
    }
}

class Triangle extends Figure {
    // Constructor to initialize dimensions for triangle
    public Triangle(double base, double height) {
        super(base, height);
    }

    // Override area method to compute area of triangle
    @Override
    double area() {
        return 0.5 * dimension1 * dimension2;
    }
}

public class FigureDemo {
    public static void main(String[] args) {
        // Create instances of Rectangle and Triangle
        Figure rectangle = new Rectangle(10, 5);
        Figure triangle = new Triangle(10, 5);

        // Display area of rectangle and triangle
```

```
        System.out.println("Area of rectangle: " + rectangle.area());
        System.out.println("Area of triangle: " + triangle.area());

        // Find the largest element in an array of integers
        int[] numbers = {3, 9, 2, 15, 6};
        int largest = findLargest(numbers);

        // Display the largest element
        System.out.println("The largest element in the array is: " + largest);
    }

    // Method to find the largest element in an array of integers
    public static int findLargest(int[] arr) {
        int max = arr[0];
        for (int num : arr) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
}
```

Explanation:

1. **Queue Data Structure:**
    - The `Queue` class defines a queue with methods to enqueue, dequeue, check if the queue is empty or full, and peek at the front element.
    - The `QueueDemo` class demonstrates how to use the `Queue` class.
2. **Unique Numbers:**
    - The program reads 5 numbers between 10 and 100 inclusive and stores unique values in a `HashSet`.
    - It displays the complete set of unique values after each new value is entered.
3. **Figure Superclass with Subclasses Rectangle and Triangle:**
    - The `Figure` class is an abstract superclass with an abstract method `area`.
    - The `Rectangle` and `Triangle` classes extend `Figure` and override the `area` method to compute the area of a rectangle and a triangle, respectively.
    - The `FigureDemo` class demonstrates creating instances of `Rectangle` and `Triangle` and finding the largest element in an array of integers.

4o
9. Write a java program to develop a java application with an Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club funds. Generate pay slips for the employees with their gross and net salary.

Sure! Below is a Java program that develops an application with an `Employee` class and its subclasses `Programmer`, `AssistantProfessor`, `AssociateProfessor`, and

`Professor`. Each subclass includes a method to calculate the gross and net salary and generate a pay slip.

```
java
Copy code
class Employee {
    String empName;
    int empId;
    String address;
    String mailId;
    String mobileNo;

    // Constructor to initialize Employee details
    public Employee(String empName, int empId, String address, String mailId,
String mobileNo) {
        this.empName = empName;
        this.empId = empId;
        this.address = address;
        this.mailId = mailId;
        this.mobileNo = mobileNo;
    }

    // Method to display Employee details
    public void displayDetails() {
        System.out.println("Employee Name: " + empName);
        System.out.println("Employee ID: " + empId);
        System.out.println("Address: " + address);
        System.out.println("Mail ID: " + mailId);
        System.out.println("Mobile No: " + mobileNo);
    }
}

class Programmer extends Employee {
    double basicPay;

    // Constructor to initialize Programmer details
    public Programmer(String empName, int empId, String address, String mailId,
String mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    // Method to generate pay slip
    public void generatePaySlip() {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - (pf + staffClubFund);

        displayDetails();
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("Staff Club Fund: " + staffClubFund);
        System.out.println("Gross Salary: " + grossSalary);
        System.out.println("Net Salary: " + netSalary);
    }
}

class AssistantProfessor extends Employee {
```

```
    double basicPay;

    // Constructor to initialize Assistant Professor details
    public AssistantProfessor(String empName, int empId, String address, String
mailId, String mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    // Method to generate pay slip
    public void generatePaySlip() {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - (pf + staffClubFund);

        displayDetails();
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("Staff Club Fund: " + staffClubFund);
        System.out.println("Gross Salary: " + grossSalary);
        System.out.println("Net Salary: " + netSalary);
    }
}

class AssociateProfessor extends Employee {
    double basicPay;

    // Constructor to initialize Associate Professor details
    public AssociateProfessor(String empName, int empId, String address, String
mailId, String mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    // Method to generate pay slip
    public void generatePaySlip() {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - (pf + staffClubFund);

        displayDetails();
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("Staff Club Fund: " + staffClubFund);
        System.out.println("Gross Salary: " + grossSalary);
        System.out.println("Net Salary: " + netSalary);
    }
}

class Professor extends Employee {
    double basicPay;

    // Constructor to initialize Professor details
```

```
    public Professor(String empName, int empId, String address, String mailId,
String mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    // Method to generate pay slip
    public void generatePaySlip() {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - (pf + staffClubFund);

        displayDetails();
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("Staff Club Fund: " + staffClubFund);
        System.out.println("Gross Salary: " + grossSalary);
        System.out.println("Net Salary: " + netSalary);
    }
}

public class EmployeeDemo {
    public static void main(String[] args) {
        // Create instances of different employee types
        Programmer programmer = new Programmer("Alice", 101, "123 Main St",
"alice@example.com", "555-1234", 50000);
        AssistantProfessor assistantProfessor = new AssistantProfessor("Bob",
102, "456 Elm St", "bob@example.com", "555-5678", 60000);
        AssociateProfessor associateProfessor = new
AssociateProfessor("Charlie", 103, "789 Oak St", "charlie@example.com", "555-
9012", 70000);
        Professor professor = new Professor("David", 104, "321 Pine St",
"david@example.com", "555-3456", 80000);

        // Generate pay slips for each employee
        System.out.println("Programmer Pay Slip:");
        programmer.generatePaySlip();
        System.out.println();

        System.out.println("Assistant Professor Pay Slip:");
        assistantProfessor.generatePaySlip();
        System.out.println();

        System.out.println("Associate Professor Pay Slip:");
        associateProfessor.generatePaySlip();
        System.out.println();

        System.out.println("Professor Pay Slip:");
        professor.generatePaySlip();
        System.out.println();
    }
}
```

Explanation:

1. **Employee Class:**

- The `Employee` class defines the common attributes for all employees: `empName`, `empId`, `address`, `mailId`, and `mobileNo`.
- The constructor initializes these attributes, and the `displayDetails` method prints the employee details.

2. **Programmer, AssistantProfessor, AssociateProfessor, and Professor Classes:**
   - Each of these classes inherits from the `Employee` class and adds a `basicPay` attribute.
   - The constructor initializes the `basicPay` along with the common attributes.
   - The `generatePaySlip` method calculates the gross and net salary using the given percentages for DA, HRA, PF, and staff club fund, and prints the pay slip.
3. **EmployeeDemo Class:**
   - The `EmployeeDemo` class creates instances of `Programmer`, `AssistantProfessor`, `AssociateProfessor`, and `Professor`.
   - It calls the `generatePaySlip` method for each instance to display their pay slips.

To run this program, compile and execute it using any Java development environment or command line. The program will generate and display pay slips for different types of employees, including their gross and net salaries.

4o

10. Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the methodprintArea( ) that prints the area of the given shape.

Certainly! Below is a Java program that demonstrates an abstract class `Shape` and its subclasses `Rectangle`, `Triangle`, and `Circle`. Each subclass implements the `printArea` method to calculate and print the area of the respective shape.

```
java
Copy code
abstract class Shape {
    int dimension1;
    int dimension2;

    // Abstract method to print the area
    abstract void printArea();
}

class Rectangle extends Shape {
    // Constructor to initialize dimensions for rectangle
    public Rectangle(int length, int width) {
        this.dimension1 = length;
        this.dimension2 = width;
    }

    // Override printArea method to compute and print area of rectangle
    @Override
    void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}
```

```
class Triangle extends Shape {
    // Constructor to initialize dimensions for triangle
    public Triangle(int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    // Override printArea method to compute and print area of triangle
    @Override
    void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    // Constructor to initialize radius for circle
    public Circle(int radius) {
        this.dimension1 = radius;
        this.dimension2 = 0; // not used for circle
    }

    // Override printArea method to compute and print area of circle
    @Override
    void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        // Create instances of Rectangle, Triangle, and Circle
        Shape rectangle = new Rectangle(10, 5);
        Shape triangle = new Triangle(10, 5);
        Shape circle = new Circle(7);

        // Print area of each shape
        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
    }
}
```

Explanation:

1. **Shape Abstract Class:**
    - The `Shape` class is an abstract class that contains two integer fields, `dimension1` and `dimension2`, and an abstract method `printArea`.
    - Subclasses of `Shape` must implement the `printArea` method to print the area of the shape.
2. **Rectangle Class:**
    - The `Rectangle` class extends `Shape` and includes a constructor to initialize the length and width.
    - The `printArea` method calculates the area of the rectangle (`length * width`) and prints it.

3. **Triangle Class:**
    - The `Triangle` class extends `Shape` and includes a constructor to initialize the base and height.
    - The `printArea` method calculates the area of the triangle (`0.5 * base * height`) and prints it.
4. **Circle Class:**
    - The `Circle` class extends `Shape` and includes a constructor to initialize the radius.
    - The `printArea` method calculates the area of the circle (`π * radius^2`) and prints it.
5. **ShapeDemo Class:**
    - The `ShapeDemo` class contains the `main` method, which creates instances of `Rectangle`, `Triangle`, and `Circle`.
    - It calls the `printArea` method for each instance to print their respective areas.

To run this program, compile and execute it using any Java development environment or command line. The program will print the areas of the rectangle, triangle, and circle.

4o

11. Write a JAVA program divide by zero to cause an ArithmeticException. Use try, catch, and finally blocks to throw an exception, catches the exception and provide a way to handle it. Use finally block regardless of whether an exception occurred or not. 12. Write a java program to create a user defined a custom exception class named CustomException that inherits from the standard Exception class. This custom exception class has a constructor that takes a message as an argument and passes it to the superclass constructor using the super() call. Here age is the input. The custom exception is intentionally set in a way that the age has a negative value to trigger the custom exception. An appropriate error message is thrown when age is negative. The catch block catches this custom exception and prints the error message. 13. Write a java program to create a File object representing the file has to be read. Create an input stream (e.g., FileInputStream or BufferedReader) to read data from the file. Use the methods provided by the stream to read data from the file like String line. 14. Write a Java program that creates three threads. First thread displays “Good Morning” every one second, the second thread displays “Hello” every two seconds and the third thread displays “Welcome” every three seconds 15. Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number. 16. Write a java program for getting a list of fruits. Print the list using for loop. Add one fruit to the list. Print the new list. 17. Write a java program for getting a list of numbers. Print the list using for loop. Using map –map the words of the numbers with number like “one” to 1. Print the new list. 18. Write a java program to read employee details from a file, sort them based on salary, and write the sorted data back to the file 19. Write a java program to use JDBC to connect to a database, execute queries, and retrieve data. Create a class JDBCDemo. Use the Database connection details like URL, username and password to connect to data base. Load and register the JDBC driver. Establish the connection. Create a statement. Execute a query. Process the result set. Retrieve other columns. Close resources 20. Write a java program to creating tasks, adding them to the list, marking them as complete, and displaying the list.

Sure! Below are the Java programs for the specified tasks.

11. Divide by Zero to Cause an ArithmeticException

```
java
Copy code
public class DivideByZero {
    public static void main(String[] args) {
        try {
            int result = 10 / 0; // This will cause ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Caught an exception: " + e.getMessage());
        } finally {
            System.out.println("This is the finally block. It always
executes.");
        }
    }
}
```

12. Custom Exception Class

```
java
Copy code
class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}

public class CustomExceptionDemo {
    public static void checkAge(int age) throws CustomException {
        if (age < 0) {
            throw new CustomException("Age cannot be negative!");
        } else {
            System.out.println("Age is: " + age);
        }
    }

    public static void main(String[] args) {
        try {
            checkAge(-5);
        } catch (CustomException e) {
            System.out.println("Caught a custom exception: " + e.getMessage());
        }
    }
}
```

13. Reading Data from a File

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

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

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

```
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

14. Creating Threads to Display Messages

```
java
Copy code
class MessageThread extends Thread {
    private String message;
    private int interval;

    public MessageThread(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
    }
}

public class MessageThreads {
    public static void main(String[] args) {
        Thread t1 = new MessageThread("Good Morning", 1000);
        Thread t2 = new MessageThread("Hello", 2000);
        Thread t3 = new MessageThread("Welcome", 3000);

        t1.start();
        t2.start();
        t3.start();
    }
}
```

15. Multi-threaded Application for Random Integers

```
java
Copy code
import java.util.Random;

class RandomNumberThread extends Thread {
    public void run() {
        Random random = new Random();
        try {
            while (true) {
                int num = random.nextInt(100);
                System.out.println("Generated number: " + num);
                if (num % 2 == 0) {
                    new SquareThread(num).start();
                } else {
```

```
                    new CubeThread(num).start();
                }
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
    }
}

class SquareThread extends Thread {
    private int number;

    public SquareThread(int number) {
        this.number = number;
    }

    public void run() {
        System.out.println("Square of " + number + " is: " + (number * number));
    }
}

class CubeThread extends Thread {
    private int number;

    public CubeThread(int number) {
        this.number = number;
    }

    public void run() {
        System.out.println("Cube of " + number + " is: " + (number * number *
number));
    }
}

public class MultiThreadedApplication {
    public static void main(String[] args) {
        new RandomNumberThread().start();
    }
}
```

16. List of Fruits

```
java
Copy code
import java.util.ArrayList;

public class FruitList {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");

        System.out.println("List of fruits:");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }

        fruits.add("Dragonfruit");
        System.out.println("\nUpdated list of fruits:");
        for (String fruit : fruits) {
            System.out.println(fruit);
```

```
        }
    }
}
```

17. List of Numbers with Mapping

```
java
Copy code
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class NumberList {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);

        System.out.println("List of numbers:");
        for (int number : numbers) {
            System.out.println(number);
        }

        Map<String, Integer> numberMap = new HashMap<>();
        numberMap.put("one", 1);
        numberMap.put("two", 2);
        numberMap.put("three", 3);

        System.out.println("\nMapped numbers:");
        for (String key : numberMap.keySet()) {
            System.out.println(key + " -> " + numberMap.get(key));
        }
    }
}
```

18. Sorting Employee Details Based on Salary

```
java
Copy code
import java.io.*;
import java.util.*;

class Employee implements Comparable<Employee> {
    String name;
    int salary;

    public Employee(String name, int salary) {
        this.name = name;
        this.salary = salary;
    }

    public int compareTo(Employee other) {
        return Integer.compare(this.salary, other.salary);
    }

    public String toString() {
        return name + ": " + salary;
    }
}

public class EmployeeSort {
```

```
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        String inputFilePath = "employees.txt"; // Replace with your input file
path
        String outputFilePath = "sorted_employees.txt"; // Replace with your
output file path

        try (BufferedReader br = new BufferedReader(new
FileReader(inputFilePath))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",");
                String name = parts[0].trim();
                int salary = Integer.parseInt(parts[1].trim());
                employees.add(new Employee(name, salary));
            }
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }

        Collections.sort(employees);

        try (PrintWriter pw = new PrintWriter(new FileWriter(outputFilePath))) {
            for (Employee employee : employees) {
                pw.println(employee);
            }
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

19. JDBC Demo

```
java
Copy code
import java.sql.*;

public class JDBCDemo {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/your_database"; // Replace
with your database URL
        String username = "your_username"; // Replace with your database
username
        String password = "your_password"; // Replace with your database
password

        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
            // Load and register JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish connection
            connection = DriverManager.getConnection(url, username, password);

            // Create statement
            statement = connection.createStatement();

            // Execute query
```

```
            String query = "SELECT * FROM your_table"; // Replace with your SQL
query
            resultSet = statement.executeQuery(query);

            // Process result set
            while (resultSet.next()) {
                // Replace with your column names
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                System.out.println("ID: " + id + ", Name: " + name);
            }
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("An error occurred: " + e.getMessage());
        } finally {
            // Close resources
            try {
                if (resultSet != null) resultSet.close();
                if (statement != null) statement.close();
                if (connection != null) connection.close();
            } catch (SQLException e) {
                System.out.println("An error occurred while closing resources: "
+ e.getMessage());
            }
        }
    }
}
```

20. Task Management Application

```
java
Copy code
import java.util.ArrayList;
import java.util.Scanner;

class Task {
    String description;
    boolean isComplete;

    public Task(String description) {
        this.description = description;
        this.isComplete = false;
    }

    public void markComplete() {
        this.isComplete = true;
    }

    public String toString() {
        return (isComplete ? "[X] " : "[ ] ") + description;
    }
}

public class TaskManager {
    private static ArrayList<Task> tasks = new ArrayList<>();

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

        do {
            System.out.println("1. Add Task");
            System.out.println("2. Mark Task as Complete");
            System.out.println("3. Display Tasks");
```

```
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter task description: ");
                    String description = scanner.nextLine();
                    tasks.add(new Task(description));
                    break;
                case 2:
                    System.out.print("Enter task number to mark as complete: ");
                    int taskNumber = scanner.nextInt();
                    if (taskNumber > 0 && taskNumber <= tasks.size()) {
                        tasks.get(taskNumber - 1).markComplete();
                    } else {
                        System.out.println("Invalid task number.");
                    }
                    break;
                case 3:
                    System.out.println("Tasks:");
                    for (int i = 0; i < tasks.size(); i++) {
                        System.out.println((i + 1) + ". " + tasks.get(i));
                    }
                    break;
                case 4:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 4);

        scanner.close();
    }
}
```

Each program above demonstrates the specified functionality in Java. You can compile and run these programs to see their respective outputs. If you have any more tasks or need further explanations, feel free to ask!