**Module -6**

**W.A.J.P to Take three numbers from the user and print the greatest number.**

import java.util.Scanner;

public class GreatestNumber

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

// Taking three numbers from the user

System.out.print(&quot;Enter the first number: &quot;);

int num1 = scanner.nextInt();

System.out.print(&quot;Enter the second number: &quot;);

int num2 = scanner.nextInt();

System.out.print(&quot;Enter the third number: &quot;);

int num3 = scanner.nextInt();

// Finding the greatest number

int greatest;

if (num1 &gt;= num2 &amp;&amp; num1 &gt;= num3)

{

greatest = num1;

}

else if (num2 &gt;= num1 &amp;&amp; num2 &gt;= num3)

{

greatest = num2;

}

else

{

greatest = num3;

}

// Printing the greatest number

System.out.println(&quot;The greatest number is: &quot; + greatest);

scanner.close();

}

}

**W.A.J.P in Java to display the first 10 natural numbers using while loop**.

public class First10NaturalNumbers

{

public static void main(String[] args)

{

int number = 1; // Start from the first natural number

System.out.println(&quot;The first 10 natural numbers are:&quot;);

while (number &lt;= 10)

{

System.out.println(number);

number++; // Increment the number by 1

}

}

}

**W.A.J.P to find factorial for Given Number.**

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Taking input from the user

System.out.print(&quot;Enter a number to find its factorial: &quot;);

int number = scanner.nextInt();

// Initializing the result to 1 (as factorial of 0 and 1 is 1)

long factorial = 1;

// Calculating the factorial using a for loop

for (int i = 1; i &lt;= number; i++) {

factorial \*= i; // Multiply the current number to the result

}

// Printing the factorial of the given number

System.out.println(&quot;The factorial of &quot; + number + &quot; is: &quot; + factorial);

scanner.close();

}

}

**W.A.J.P to check given number is Prime or not?**

import java.util.Scanner;

public class PrimeCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Taking input from the user

System.out.print(&quot;Enter a number to check if it&#39;s prime: &quot;);

int number = scanner.nextInt();

// Variable to store if the number is prime

boolean isPrime = true;

// A prime number is greater than 1 and divisible only by 1 and itself

if (number &lt;= 1) {

isPrime = false; // Numbers less than or equal to 1 are not prime

} else

{

// Check divisibility from 2 to the square root of the number

for (int i = 2; i &lt;= Math.sqrt(number); i++) {

if (number % i == 0) {

isPrime = false; // If divisible by any number other than 1 and itself, it&#39;s not prime

break;

break;

}

}

}

// Output the result

if (isPrime) {

System.out.println(number + &quot; is a prime number.&quot;);

} else {

System.out.println(number + &quot; is not a prime number.&quot;);

}

scanner.close();

}

}

**W.A.J.P to check given number is Armstrong or not?**

import java.util.Scanner;

public class ArmstrongNumber

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

// Taking input from the user

System.out.print(&quot;Enter a number to check if it&#39;s an Armstrong number: &quot;);

int number = scanner.nextInt();

int originalNumber = number;

int sum = 0;

// Calculate the number of digits in the number

int digits = String.valueOf(number).length();

// Calculate the sum of the digits raised to the power of the number of digits

while (number &gt; 0)

{

int digit = number % 10;

// Extract the last digit

sum += Math.pow(digit, digits);

// Add the digit raised to the power of the number of digits to sum

number /= 10;

// Remove the last digit

}

// Check if the sum is equal to the original number

if (sum == originalNumber) {

System.out.println(originalNumber + &quot; is an Armstrong number.&quot;);

} else {

System.out.println(originalNumber + &quot; is not an Armstrong number.&quot;);

}

scanner.close();

}

}

**W.A.J.P for create Fibonacci Series.**

import java.util.Scanner;

public class FibonacciSeries {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Taking input from the user

System.out.print(&quot;Enter the number of terms in the Fibonacci series: &quot;);

int n = scanner.nextInt();

// Variables to store the first two Fibonacci numbers

int first = 0, second = 1;

System.out.println(&quot;The Fibonacci series is:&quot;);

// Displaying the first term

System.out.print(first + &quot; &quot;);

// Displaying the remaining terms

for (int i = 1; i &lt; n; i++) {

System.out.print(second + &quot; &quot;);

int next = first + second; // Calculating the next term in the series

first = second; // Updating first to second

second = next; // Updating second to next

}

scanner.close();

}

}

**W.A.J.P to Print pattern Given Below**

**1**

**12**

**123**

**1234**

**12345**

public class NumberPattern {

public static void main(String[] args) {

int rows = 5; // Number of rows

for (int i = 1; i &lt;= rows; i++) { // Loop through rows

for (int j = 1; j &lt;= i; j++) { // Loop to print numbers in each row

System.out.print(j);

}

System.out.println(); // Move to the next line after each row

}

}

}

output

1

12

123

1234

12345

-------------------------------------------------------------------

**1**

**01**

**101**

**0101**

**10101**

public class BinaryPattern {

public static void main(String[] args) {

int rows = 5; // Number of rows

for (int i = 1; i &lt;= rows; i++) { // Loop through rows

for (int j = 1; j &lt;= i; j++) { // Loop to print numbers in each row

// Print 1 if the sum of i and j is even, else print 0

if ((i + j) % 2 == 0) {

System.out.print(&quot;1&quot;);

} else {

System.out.print(&quot;0&quot;);

}

}

System.out.println(); // Move to the next line after each row

}

}

}

----------------------------------------------------------

**1**

**2   2**

**3   3  3**

**4  4  4  4**

public class PyramidPattern {

public static void main(String[] args) {

int rows = 4; // Number of rows

for (int i = 1; i &lt;= rows; i++) { // Loop through rows

// Print leading spaces

for (int j = 1; j &lt;= rows - i; j++) {

System.out.print(&quot; &quot;);

}

// Print the numbers with a space between them

for (int j = 1; j &lt;= i; j++) {

System.out.print(i + &quot; &quot;);

}

// Move to the next line after each row

System.out.println();

}

}

}

**\***

**\* \* \***

**\* \* \* \* \***

**\* \* \***

**\***

public class DiamondPattern {

public static void main(String[] args) {

int rows = 4; // Number of rows in the widest part of the diamond

// Print the upper part of the diamond

for (int i = 1; i &lt;= rows; i++) {

// Print leading spaces

for (int j = 1; j &lt;= rows - i; j++) {

System.out.print(&quot; &quot;);

}

// Print stars with spaces in between

for (int j = 1; j &lt;= (2 \* i - 1); j++) {

System.out.print(&quot;\* &quot;);

}

System.out.println(); // Move to the next line

}

// Print the lower part of the diamond

for (int i = rows - 1; i &gt;= 1; i--) {

// Print leading spaces

for (int j = 1; j &lt;= rows - i; j++) {

System.out.print(&quot; &quot;);

}

// Print stars with spaces in between

for (int j = 1; j &lt;= (2 \* i - 1); j++) {

System.out.print(&quot;\* &quot;);

}

System.out.println(); // Move to the next line

}

}

}

**WAP to compute the sum of the first 100 prime numbers**

public class SumOfPrimes {

public static void main(String[] args) {

int count = 0; // To count the number of prime numbers found

long sum = 0; // To store the sum of prime numbers

int number = 2; // The number to be checked for primality

while (count &lt; 100) { // Continue until we find 100 prime numbers

if (isPrime(number)) {

sum += number; // Add the prime number to the sum

count++; // Increment the count of prime numbers

}

number++; // Move to the next number

}

// Print the sum of the first 100 prime numbers

System.out.println(&quot;The sum of the first 100 prime numbers is: &quot; + sum);

}

// Method to check if a number is prime

public static boolean isPrime(int num) {

if (num &lt;= 1) return false; // Numbers less than or equal to 1 are not prime

if (num &lt;= 3) return true; // 2 and 3 are prime numbers

if (num % 2 == 0 || num % 3 == 0) return false; // Check divisibility by 2 and 3

// Check divisibility from 5 to sqrt(num)

for (int i = 5; i \* i &lt;= num; i += 6) {

if (num % i == 0 || num % (i + 2) == 0) {

return false;

}

}

return true;

}

}

**WAP to sum values of an array.**

public class ArraySum {

public static void main(String[] args) {

// Initialize the array with some values

int[] numbers = {1, 2, 3, 4, 5}; // You can change these values or size as needed

// Variable to store the sum of the array elements

int sum = 0;

// Loop through the array and add each element to the sum

for (int number : numbers) {

sum += number;

}

// Print the sum of the array elements

System.out.println(&quot;The sum of the array values is: &quot; + sum);

}

}

**WAP to calculate the average value of array elements.**

public class ArrayAverage {

public static void main(String[] args) {

// Initialize the array with some values

int[] numbers = {10, 20, 30, 40, 50}; // You can change these values or size as needed

// Variable to store the sum of the array elements

int sum = 0;

// Loop through the array and add each element to the sum

for (int number : numbers) {

sum += number;

}

// Calculate the average

double average = (double) sum / numbers.length;

// Print the average of the array elements

System.out.println(&quot;The average value of the array elements is: &quot; + average);

}

}

**WAP to find the index of an array element.**

public class FindElementIndex {

public static void main(String[] args) {

// Initialize the array with some values

int[] numbers = {10, 20, 30, 40, 50}; // You can change these values as needed

// Element to find in the array

int target = 30; // Change this value to the element you want to find

// Variable to store the index of the target element

int index = -1; // -1 indicates that the element was not found

// Loop through the array to find the target element

for (int i = 0; i &lt; numbers.length; i++) {

if (numbers[i] == target) {

index = i; // Store the index of the target element

break; // Exit the loop once the element is found

}

}

// Print the result

if (index != -1) {

System.out.println(&quot;The index of element &quot; + target + &quot; is: &quot; + index);

} else {

System.out.println(&quot;Element &quot; + target + &quot; is not found in the array.&quot;);

}

}

}

**WAP to find the maximum and minimum value of an array.**

public class MinMaxArray {

public static void main(String[] args) {

// Initialize the array with some values

int[] numbers = {45, 22, 89, 34, 67, 23, 90}; // You can change these values as needed

// Check if the array is not empty

if (numbers.length == 0) {

System.out.println(&quot;The array is empty.&quot;);

return;

}

// Initialize min and max with the first element of the array

int min = numbers[0];

int max = numbers[0];

// Loop through the array to find the min and max values

for (int i = 1; i &lt; numbers.length; i++) {

if (numbers[i] &lt; min) {

min = numbers[i]; // Update min if a smaller value is found

}

if (numbers[i] &gt; max) {

max = numbers[i]; // Update max if a larger value is found

}

}

// Print the result

System.out.println(&quot;The minimum value in the array is: &quot; + min);

System.out.println(&quot;The maximum value in the array is: &quot; + max);

}

}

**WAP to Compare Two String.**

import java.util.Scanner;

public class CompareStrings {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter the first string

System.out.print(&quot;Enter the first string: &quot;);

String string1 = scanner.nextLine();

// Prompt the user to enter the second string

System.out.print(&quot;Enter the second string: &quot;);

String string2 = scanner.nextLine();

// Compare strings using equals()

if (string1.equals(string2)) {

System.out.println(&quot;The strings are equal.&quot;);

} else {

System.out.println(&quot;The strings are not equal.&quot;);

}

// Compare strings using compareTo()

int comparisonResult = string1.compareTo(string2);

if (comparisonResult &lt; 0) {

System.out.println(&quot;The first string is less than the second string.&quot;);

} else if (comparisonResult &gt; 0) {

System.out.println(&quot;The first string is greater than the second string.&quot;);

} else {

System.out.println(&quot;The strings are equal.&quot;);

}

scanner.close();

}

}

**WAP to concatenate a given string to the end of another string.**

import java.util.Scanner;

public class StringConcatenation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter the first string

System.out.print(&quot;Enter the first string: &quot;);

String firstString = scanner.nextLine();

// Prompt the user to enter the second string

System.out.print(&quot;Enter the second string: &quot;);

String secondString = scanner.nextLine();

// Concatenate using concat() method

String concatenatedString1 = firstString.concat(secondString);

// Concatenate using + operator

String concatenatedString2 = firstString + secondString;

// Print the results

System.out.println(&quot;Concatenated using concat(): &quot; + concatenatedString1);

System.out.println(&quot;Concatenated using + operator: &quot; + concatenatedString2);

scanner.close();

}

}

**WAP to demonstrate try catch block.**

import java.util.Scanner;

public class TryCatchDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter two numbers

System.out.print(&quot;Enter the numerator: &quot;);

int numerator = scanner.nextInt();

System.out.print(&quot;Enter the denominator: &quot;);

int denominator = scanner.nextInt();

try {

// Attempt to perform division

int result = numerator / denominator;

System.out.println(&quot;The result of division is: &quot; + result);

} catch (ArithmeticException e) {

// Handle division by zero error

System.out.println(&quot;Error: Cannot divide by zero.&quot;);

} catch (Exception e) {

// Handle any other unexpected exceptions

System.out.println(&quot;An unexpected error occurred: &quot; + e.getMessage());

} finally {

// This block will always execute, whether an exception occurs or not

System.out.println(&quot;Execution completed.&quot;);

scanner.close();

}

}

}

**WAP to demonstrate multiple catch blocks**

import java.util.InputMismatchException;

import java.util.Scanner;

public class MultipleCatchDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

// Prompt the user to enter two numbers

System.out.print(&quot;Enter the numerator: &quot;);

int numerator = scanner.nextInt();

System.out.print(&quot;Enter the denominator: &quot;);

int denominator = scanner.nextInt();

// Attempt to perform division

int result = numerator / denominator;

System.out.println(&quot;The result of division is: &quot; + result);

} catch (ArithmeticException e) {

// Handle division by zero error

System.out.println(&quot;Error: Cannot divide by zero.&quot;);

} catch (InputMismatchException e) {

// Handle invalid input type (e.g., if user enters a non-integer value)

System.out.println(&quot;Error: Invalid input. Please enter integers only.&quot;);

} catch (Exception e) {

// Handle any other unexpected exceptions

System.out.println(&quot;An unexpected error occurred: &quot; + e.getMessage());

} finally {

// This block will always execute, whether an exception occurs or not

System.out.println(&quot;Execution completed.&quot;);

scanner.close();

}

}

}

**WAP to create one thread by implementing Runnable interface in Class.**

To create a thread in Java using the Runnable interface, you need to follow these steps:

1. Implement the Runnable interface: Define a class that implements Runnable and

override its run() method with the code that should be executed by the thread.

2. Create a Thread object: Instantiate a Thread object with an instance of the

Runnable class.

3. Start the thread: Call the start() method on the Thread object to begin execution.

Here&#39;s an example of how to do this:

// Implement the Runnable interface

class MyRunnable implements Runnable {

@Override

public void run() {

// Code to be executed by the thread

for (int i = 0; i &lt; 5; i++) {

System.out.println(&quot;Thread is running: &quot; + i);

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(&quot;Thread interrupted.&quot;);

}

}

}

}

public class RunnableExample {

public static void main(String[] args) {

// Create an instance of MyRunnable

MyRunnable myRunnable = new MyRunnable();

// Create a Thread object and pass the MyRunnable instance

Thread thread = new Thread(myRunnable);

// Start the thread

thread.start();

// Main thread continues to run in parallel

for (int i = 0; i &lt; 5; i++) {

System.out.println(&quot;Main thread: &quot; + i);

try {

Thread.sleep(500); // Sleep for 0.5 seconds

} catch (InterruptedException e) {

System.out.println(&quot;Main thread interrupted.&quot;);

}

}

}

}

**WAP to create one thread by extending Thread class in another Class.**

// Extend the Thread class

class MyThread extends Thread {

@Override

public void run() {

// Code to be executed by the thread

for (int i = 0; i &lt; 5; i++) {

System.out.println(&quot;Thread is running: &quot; + i);

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(&quot;Thread interrupted.&quot;);

}

}

}

}

public class ThreadExample {

public static void main(String[] args) {

// Create an instance of MyThread

MyThread myThread = new MyThread();

// Start the thread

myThread.start();

// Main thread continues to run in parallel

for (int i = 0; i &lt; 5; i++) {

System.out.println(&quot;Main thread: &quot; + i);

try {

Thread.sleep(500); // Sleep for 0.5 seconds

} catch (InterruptedException e) {

System.out.println(&quot;Main thread interrupted.&quot;);

}

}

}

}

**WAP to iterate through all elements in an array list.**

To iterate through all elements in an ArrayList in Java, you can use several methods:

1. Using a for-each loop.

2. Using an Iterator.

3. Using a for loop with an index.

4. Using Java 8&#39;s forEach method with a lambda expression.

Here &#39;s an example demonstrating each method:

import java.util.ArrayList;

import java.util.Iterator;

public class ArrayListIteration {

public static void main(String[] args) {

// Create an ArrayList and add some elements

ArrayList&lt;String&gt; list = new ArrayList&lt;&gt;();

list.add(&quot;Apple &quot;);

list.add(&quot;Banana &quot;);

list.add(&quot;Cherry &quot;);

list.add(&quot;Date &quot;);

// 1. Using for-each loop

System.out.println(&quot;Using for-each loop:&quot;);

for (String fruit : list) {

System.out.println(fruit);

}

// 2. Using Iterator

System.out.println(&quot;\nUsing Iterator:&quot;);

Iterator&lt;String&gt; iterator = list.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

// 3. Using for loop with index

System.out.println(&quot;\nUsing for loop with index:&quot;);

for (int i = 0; i &lt; list.size(); i++) {

System.out.println(list.get(i));

}

// 4. Using Java 8 forEach with lambda

System.out.println(&quot;\nUsing Java 8 forEach with lambda:&quot;);

list.forEach(fruit -&gt; System.out.println(fruit));

}

}

**WAP to update specific array elements by given element.**

import java.util.Scanner;

public class UpdateArrayElement {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Initialize the array with some values

int[] numbers = {10, 20, 30, 40, 50};

// Display the original array

System.out.println(&quot;Original array:&quot;);

for (int num : numbers) {

System.out.print(num + &quot; &quot;);

}

System.out.println();

// Prompt the user to enter the index of the element to update

System.out.print(&quot;Enter the index of the element to update (0 to &quot; + (numbers.length - 1)

+ &quot;): &quot;);

int index = scanner.nextInt();

// Check if the index is valid

if (index &lt; 0 || index &gt;= numbers.length) {

System.out.println(&quot;Error: Index out of bounds.&quot;);

} else {

// Prompt the user to enter the new value

System.out.print(&quot;Enter the new value: &quot;);

int newValue = scanner.nextInt();

// Update the array element

numbers[index] = newValue;

// Display the updated array

System.out.println(&quot;Updated array:&quot;);

for (int num : numbers) {

System.out.print(num + &quot; &quot;);

}

System.out.println();

}

scanner.close();

}

}

**WAP to remove the third element from a array list**.

---------------------------------------------------------------

import java.util.ArrayList;

import java.util.Scanner;

public class RemoveThirdElement {

public static void main(String[] args) {

// Create an ArrayList and add some elements

ArrayList&lt;String&gt; list = new ArrayList&lt;&gt;();

list.add(&quot;Apple &quot;);

list.add(&quot;Banana &quot;);

list.add(&quot;Cherry &quot;);

list.add(&quot;Date &quot;);

list.add(&quot;Elderberry &quot;);

// Display the original ArrayList

System.out.println(&quot;Original ArrayList:&quot;);

for (String item : list) {

System.out.println(item);

}

// Remove the third element if it exists

if (list.size() &gt;= 3) {

list.remove(2); // Index 2 corresponds to the third element

} else {

System.out.println(&quot;The ArrayList does not contain enough elements to remove the

third one.&quot;);

}

// Display the updated ArrayList

System.out.println(&quot;\nUpdated ArrayList:&quot;);

for (String item : list) {

System.out.println(item);

}

}

}

**22) WAP to Copy one array into another**

--------------------------------------

1. Using a Manual Loop

public class CopyArrayManual {

public static void main(String[] args) {

// Source array

int[] sourceArray = {1, 2, 3, 4, 5};

// Create a destination array of the same length

int[] destinationArray = new int[sourceArray.length];

// Copy elements using a loop

for (int i = 0; i &lt; sourceArray.length; i++) {

destinationArray[i] = sourceArray[i];

}

// Display the destination array

System.out.println(&quot;Destination Array:&quot;);

for (int num : destinationArray) {

System.out.print(num + &quot; &quot;);

}

}

}

2.Using System.arraycopy()

public class CopyArraySystemArraycopy {

public static void main(String[] args) {

// Source array

int[] sourceArray = {1, 2, 3, 4, 5};

// Create a destination array of the same length

int[] destinationArray = new int[sourceArray.length];

// Copy elements using System.arraycopy

System.arraycopy(sourceArray, 0, destinationArray, 0, sourceArray.length);

// Display the destination array

System.out.println(&quot;Destination Array:&quot;);

for (int num : destinationArray) {

System.out.print(num + &quot; &quot;);

}

}

}

3.Using Arrays.copyOf()

import java.util.Arrays;

public class CopyArrayArraysCopyOf {

public static void main(String[] args) {

// Source array

int[] sourceArray = {1, 2, 3, 4, 5};

// Copy elements using Arrays.copyOf

int[] destinationArray = Arrays.copyOf(sourceArray, sourceArray.length);

// Display the destination array

System.out.println(&quot;Destination Array:&quot;);

for (int num : destinationArray) {

System.out.print(num + &quot; &quot;);

}

}

}

**23) WAP to reverse an array of integer values.**

public class ReverseArray {

public static void main(String[] args) {

// Initialize the array with some values

int[] array = {1, 2, 3, 4, 5};

// Display the original array

System.out.println(&quot;Original array:&quot;);

for (int num : array) {

System.out.print(num + &quot; &quot;);

}

System.out.println();

// Reverse the array

reverseArray(array);

// Display the reversed array

System.out.println(&quot;Reversed array:&quot;);

for (int num : array) {

System.out.print(num + &quot; &quot;);

}

}

// Method to reverse the array

public static void reverseArray(int[] array) {

int start = 0;

int end = array.length - 1;

while (start &lt; end) {

// Swap elements at start and end indices

int temp = array[start];

array[start] = array[end];

array[end] = temp;

// Move indices towards the center

start++;

end--;

}

}

}

**24) WAP to find the second largest element in an array.**

To find the second largest element in an array, you can follow these steps:

1. Initialize Variables: Use two variables to keep track of the largest and the

second largest elements.

2. Iterate Through the Array: Compare each element with the largest and second

largest values and update them accordingly.

3. Handle Edge Cases: Ensure that the array has at least two distinct elements to

find the second largest element.

Here’s a Java program to find the second largest element in an array:

public class SecondLargestElement {

public static void main(String[] args) {

// Initialize the array with some values

int[] array = {10, 5, 20, 8, 15};

// Call the method to find the second largest element

int secondLargest = findSecondLargest(array);

// Display the result

if (secondLargest != Integer.MIN\_VALUE) {

System.out.println(&quot;The second largest element is: &quot; + secondLargest);

} else {

System.out.println(&quot;The array does not have enough distinct elements.&quot;);

}

}

// Method to find the second largest element in the array

public static int findSecondLargest(int[] array) {

if (array.length &lt; 2) {

// Not enough elements to determine the second largest

return Integer.MIN\_VALUE;

}

int largest = Integer.MIN\_VALUE;

int secondLargest = Integer.MIN\_VALUE;

for (int num : array) {

if (num &gt; largest) {

// Update both largest and second largest

secondLargest = largest;

largest = num;

} else if (num &gt; secondLargest &amp;&amp; num &lt; largest) {

// Update only second largest

secondLargest = num;

}

}

return secondLargest;

}

}

**25) W.A.J.P. Create an abstract class &#39;Parent&#39; with a method &#39;message&#39;. It has two subclasses**

**each having a method with the same name &#39;message&#39; that prints &quot;This is first subclass &quot; and**

**&quot; This is second subclass &quot; respectively. Call the methods &#39;message&#39; by creating an object for**

**each subclass.**

// Abstract class Parent

abstract class Parent {

// Abstract method

abstract void message();

}

// First subclass

class FirstSubclass extends Parent {

// Override the message method

@Override

void message() {

System.out.println(&quot;This is the first subclass &quot;);

}

}

// Second subclass

class SecondSubclass extends Parent {

// Override the message method

@Override

void message() {

System.out.println(&quot;This is the second subclass &quot;);

}

}

public class AbstractClassDemo {

public static void main(String[] args) {

// Create an object of FirstSubclass

Parent firstObject = new FirstSubclass();

// Call the message method

firstObject.message();

// Create an object of SecondSubclass

Parent secondObject = new SecondSubclass();

// Call the message method

secondObject.message();

}

}

**26) W.A.J.P. which will ask the user to enter his/her marks (out of 100). Define a method that**

**will display grades according to the marks entered as below: Marks Grade 91-100 AA 81-90**

**AB 71-80 BB 61-70 BC 51-60 CD 41-50 DD &lt;=40 Fail**

import java.util.Scanner;

public class GradeCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter marks

System.out.print(&quot;Enter your marks (out of 100): &quot;);

int marks = scanner.nextInt();

// Validate marks input

if (marks &lt; 0 || marks &gt; 100) {

System.out.println(&quot;Error: Marks should be between 0 and 100.&quot;);

} else {

// Call the method to display the grade

displayGrade(marks);

}

scanner.close();

}

// Method to determine and display the grade based on the marks

public static void displayGrade(int marks) {

if (marks &gt;= 91) {

System.out.println(&quot;Grade: AA &quot;);

} else if (marks &gt;= 81) {

System.out.println(&quot;Grade: AB &quot;);

} else if (marks &gt;= 71) {

System.out.println(&quot;Grade: BB &quot;);

} else if (marks &gt;= 61) {

System.out.println(&quot;Grade: BC &quot;);

} else if (marks &gt;= 51) {

System.out.println(&quot;Grade: CD &quot;);

} else if (marks &gt;= 41) {

System.out.println(&quot;Grade: DD &quot;);

} else {

System.out.println(&quot;Grade: Fail &quot;);

}

}

}

**27) W.A.J.P. to create a custom exception if Customer withdraw amount which is greater than**

**account balance then program will show custom exception otherwise amount will deduct**

**from account balance. Account balance is:2000 Enter withdraw amount:2500 Sorry,**

**insufficient balance, you need more 500 Rs.To perform this transaction.**

1. Define the Custom Exception

// Custom exception class

class InsufficientBalanceException extends Exception {

public InsufficientBalanceException(String message) {

super(message);

}

}

2.Create the Account Class

import java.util.Scanner;

public class Account {

private int balance;

// Constructor to initialize account with balance

public Account(int initialBalance) {

this.balance = initialBalance;

}

// Method to withdraw amount

public void withdraw(int amount) throws InsufficientBalanceException {

if (amount &gt; balance) {

// Calculate the deficit

int deficit = amount - balance;

// Throw custom exception with a message

throw new InsufficientBalanceException(&quot;Sorry, insufficient balance, you need

more &quot; + deficit + &quot; Rs. to perform this transaction.&quot;);

} else {

// Deduct the amount from the balance

balance -= amount;

System.out.println(&quot;Withdrawal successful. Remaining balance: &quot; + balance + &quot;

Rs.&quot;);

}

}

}

1. Main Program to Test the Account Class

import java.util.Scanner;

public class BankingApp {

public static void main(String[] args) {

// Initialize account with balance of 2000 Rs.

Account account = new Account(2000);

Scanner scanner = new Scanner(System.in);

// Prompt user to enter the withdrawal amount

System.out.print(&quot;Enter withdrawal amount: &quot;);

int withdrawAmount = scanner.nextInt();

try {

// Attempt to withdraw the amount

account.withdraw(withdrawAmount);

} catch (InsufficientBalanceException e) {

// Handle custom exception

System.out.println(e.getMessage());

} finally {

// Close the scanner

scanner.close();

}

}

}