**ASSINGMENT 01:** Write a JAVA program to read 3 subjects marks, calculate the total and average marks. Display the grade based on the following criteria:

Note: Percentage>=90% : Grade A Percentage>=80% : Grade B Percentage>=70% : Grade C Percentage>=60% : Grade D Percentage>=40% : Grade E Percentage Percentage <40%: Grade F

import java.util.Scanner;

class Student\_Grade

{

public static void main(String[] args)

{

Scanner input = new Scanner(System.in);

System.out.println("Enter The Five Subject Marks :");

int m1 = input.nextInt();

int m2 = input.nextInt();

int m3 = input.nextInt();

int m4 = input.nextInt();

int m5 = input.nextInt();

int tot = m1+m2+m3+m4+m5;

float per = (tot/500)\*100;

System.out.println("Total :"+tot);

System.out.println("Percentage :"+per);

if(per>=90 && per<=100)

System.out.println("Grade A");

else if(per>=80 && per<=89)

System.out.println("Grade B");

else if(per>=70 && per<=79)

System.out.println("Grade C");

else if(per>=60 && per<=69)

System.out.println("Grade D");

else if(per>=40)

System.out.println("Grade E");

else

System.out.println("Grade F");

}

}

**Assignment 02:** Write a JAVA program to create a class called Person with p\_name and age. Which includes constructor to initialize these fields and a method to display the person information. In the main method, an instance of the Person class will be created and Person information is displayed.

public class Person {

// Fields

private String name;

private int age;

// Constructor

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display information

public void displayInfo() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

public static void main(String[] args) {

// Creating an instance of Person using the constructor

Person person1 = new Person("John Doe", 25);

// Displaying information using the displayInfo method

person1.displayInfo();

}

}

**Explanation:**

The **Person** class encapsulates the data related to an individual, and its constructor ensures that a new person object is properly initialized with a name and age. The **displayInfo** method provides a way to output the person’s details in a structured format.

In the **main** method, an instance of the **Person** class, named **person1**, is created with the name “John Doe” and age 25. Subsequently, the **displayInfo** method is called on this instance, printing the person’s name and age to the console.

In this program, there’s a **Person** class with private fields **name** and **age**, a constructor that initializes these fields, and a **displayInfo** method to print the person’s information. The **main** method creates an instance of the **Person** class using the constructor and then displays the information using the **displayInfo** method.

Overall, this program demonstrates the principles of encapsulation, constructor usage, and method invocation in Java, providing a basic structure for working with person-related data.Top of Form

**Assignment 03:** Define a class named Animal with two methods: eat() and sleep(). Create a class named Dog that extends the Animal which includes a method called bark() and also create a class called Cat that extends Animal which includes Meow() method. Create a class called MainClass to write main() method and demonstrate the following:

Inside the **main** method:

* 1. Create objects of the **Dog** and **Cat** classes (**myDog** and **myCat**).
  2. Demonstrate calling methods from the parent class (**eat()** and **sleep()**).
  3. Illustrate calling methods from the **Dog** class (**bark()**).
  4. Showcase calling methods from the **Cat** class (**meow()**).

class Animal {

void eat() {

System.out.println("The animal is eating");

}

void sleep() {

System.out.println("The animal is sleeping");

}

}

// Child class inheriting from Animal

class Dog extends Animal {

void bark() {

System.out.println("The dog is barking");

}

}

// Another child class inheriting from Animal

class Cat extends Animal {

void meow() {

System.out.println("The cat is meowing");

}

}

// Main class to test the inheritance

public class InheritanceExample {

public static void main(String[] args) {

// Creating objects of the child classes

Dog myDog = new Dog();

Cat myCat = new Cat();

// Calling methods from the parent class

myDog.eat();

myDog.sleep();

// Calling methods from the child class

myDog.bark();

// Calling methods from another child class

myCat.eat();

myCat.sleep();

myCat.meow();

}

}

**ASSIGNMENT 04: Write a JAVA program calculate factorial of a given number n.**

**EXPLAINATION:**

1. **Import Statement:** The program starts by importing the **Scanner** class from the **util** package to facilitate user input.
2. **User Input:** The **main** method prompts the user to enter a non-negative integer, which is then read using the **Scanner**
3. **Input Validation:** The program checks if the entered integer is non-negative. If it is, the factorial is calculated; otherwise, an error message is displayed, and the program terminates.
4. **Factorial Calculation:** The **calculateFactorial** method is a recursive function that computes the factorial of the input integer **n**. The base cases check if **n** is 0 or 1, in which case the factorial is 1. Otherwise, the factorial is calculated by multiplying **n** with the factorial of **(n – 1)**.
5. **Display Result:** If the input is valid, the program prints the calculated factorial for the entered integer.
6. **Scanner Closure:** The **Scanner** is closed to release system resources.

**Algorithm:**

1. Begin the program.
2. Import the **Scanner**
3. Create a **Scanner** object to read user input.
4. Prompt the user to enter a non-negative integer.
5. Read the entered integer.
6. Check if the integer is non-negative.
   * If negative, display an error message and terminate the program.
   * If non-negative, proceed to the next step.
7. Call the **calculateFactorial** method with the entered integer.
8. The **calculateFactorial** method:
   * If **n** is 0 or 1, return 1.
   * Otherwise, return **n \* calculateFactorial(n – 1)**.
9. Display the calculated factorial for the entered integer.
10. Close the **Scanner**
11. End the program.

import java.util.Scanner;

public class FactorialCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a non-negative integer: ");

int n = scanner.nextInt();

if (n < 0) {

System.out.println("Please enter a non-negative integer.");

} else {

long factorial = fact(n);

System.out.println("Factorial of " + n + " is: " + factorial);

}

scanner.close();

}

private static long fact(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* fact(n - 1);

}

}

}

**ASSIGNMENT 05:** Write a JAVA program that prompts the user to enter a number between 1 and 7. The program then uses a switch case statement to determine the corresponding day of the week based on the user’s input and prints the result.

**Explanation:**

* The program begins by importing the **Scanner** class, which is used for user input.
* The **DayOfWeek** class contains the **main** method where the execution of the program starts.
* The user is prompted to input a number between 1 and 7, representing days of the week.
* The entered number is stored in the variable **dayNumber**.
* The program uses a switch case statement to match the **dayNumber** with the corresponding days of the week.
* If the entered number matches one of the cases (1-7), the respective day is assigned to the variable **day**.
* If the entered number is outside the range, the default case is triggered, assigning “Invalid day number” to the **day** variable.
* Finally, the determined day of the week (or the error message) is printed to the console.

This program is a basic example of using a switch case statement in Java to make decisions based on user input.

import java.util.Scanner;

public class DayOfWeek {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number (1-7): ");

int dayNumber = scanner.nextInt();

String day;

switch (dayNumber) {

case 1:

day = "Sunday";

break;

case 2:

day = "Monday";

break;

case 3:

day = "Tuesday";

break;

case 4:

day = "Wednesday";

break;

case 5:

day = "Thursday";

break;

case 6:

day = "Friday";

break;

case 7:

day = "Saturday";

break;

default:

day = "Invalid day number";

}

System.out.println("Day of the week: " + day);

}

}

**ASSIGNEMENT 06:** Create a  Shape class serves as the base class for all shapes. It contains a method draw() that prints a generic message indicating the act of drawing a shape. The **Circle** class extends the **Shape** class and overrides the **draw()** method to provide a specialized implementation for drawing a circle. Similarly, the **Square** class extends the **Shape** class and provides its own implementation of the **draw()** method to handle drawing a square. Create MainClass contains the **main** method, serving as the entry point of the program. Two objects, shape1 and shape2, are declared of type Shape but instantiated as Circle and Square objects, respectively. This demonstrates the polymorphic behavior, allowing objects of derived classes to be treated as objects of the base class.

### Explanation:

The program showcases the power of polymorphism, emphasizing the ability to treat objects of derived classes uniformly through a common base class. This flexibility promotes code reusability and simplifies maintenance, as new shapes can be added without modifying existing code. The concept of overriding methods enables each shape to define its unique behavior while adhering to a shared interface provided by the **Shape** class. The program output demonstrates how the correct **draw()** method is dynamically selected based on the actual type of the object at runtime, illustrating the essence of polymorphic behavior in Java. In this example, **Circle** and **Square** are subclasses of the **Shape** class. Each subclass overrides the **draw** method to provide its own implementation. In the **main** method, we create objects of type **Circle** and **Square**, but we declare them as type **Shape**. This allows us to use polymorphism, and when we call the **draw** method on these objects, it dynamically dispatches to the appropriate method based on the actual type of the object at runtime.

class Shape {

public void draw() {

System.out.println("Drawing a shape");

}

}

class Circle extends Shape {

@Override

public void draw() {

System.out.println("Drawing a circle");

}

}

class Square extends Shape {

@Override

public void draw() {

System.out.println("Drawing a square");

}

}

public class PolymorphismExample {

public static void main(String[] args) {

Shape shape1 = new Circle();

Shape shape2 = new Square();

// Polymorphism in action

shape1.draw(); // Calls the draw method of Circle

shape2.draw(); // Calls the draw method of Square

}

}

**ASSIGNEMENT 07: Write a JAVA program to check given string is palindrome or not.**

**Explanation:**

1. **User Input:**
   * The program starts by creating a **Scanner** object to take input from the user.
   * The user is prompted to enter a string.
2. **Palindrome Checking:**
   * The program then calls the **isPalindrome** method to check if the entered string is a palindrome.
   * The **isPalindrome** method performs the following steps:
     + It initializes two pointers, **left** and **right**, pointing to the start and end of the cleaned string.
     + It uses a while loop to compare characters at the **left** and **right**
       - If the characters are not equal, the method returns **false** indicating that the string is not a palindrome.
       - The pointers are then updated to check the next pair of characters.
     + If the loop completes without returning **false**, the method returns **true**, indicating that the string is a palindrome.
3. **Output:**
   * Depending on the result of the palindrome check, the program prints whether the entered string is a palindrome or not.

import java.util.Scanner;

public class PalindromeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

if (isPalindrome(input)) {

System.out.println(input + " is a palindrome!");

} else {

System.out.println(input + " is not a palindrome.");

}

scanner.close();

}

public static boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false; // Not a palindrome

}

left++;

right--;

}

return true; // It's a palindrome

}

}

**ASSIGNEMENT 08:** Create a servlet for a login page. If the username and password are correct then it says message “Hello” else a message “login failed”.

pacakage bec.com;

Import java.io.\*;

Import javax.servlet.http.\*;

Import javax.servlet.\*;

Import java.util.\*;

import java.sql.\*;

public class logincheck extends HttpServlet

{

protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

response.setContentType("text/html");

PrintWriter out=response.getWriter();

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb","root","admin");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from employee having salary>=10000");

while(rs.next())

{

}

}

catch(Exception e)

Bottom of Form