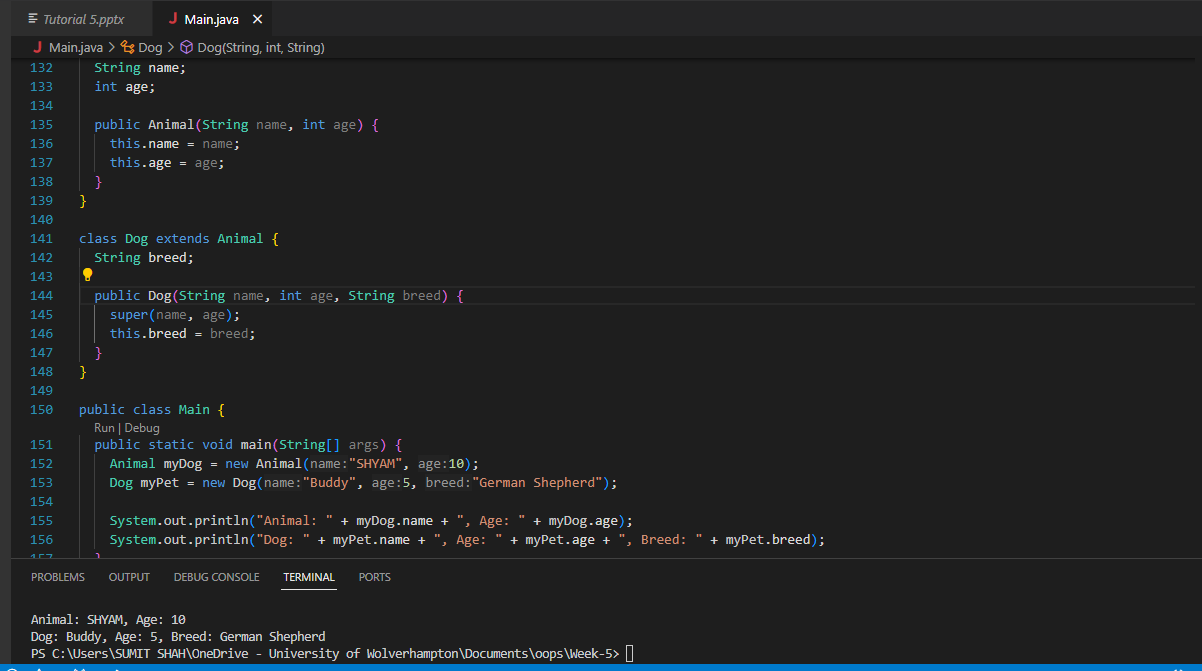
Section - 1

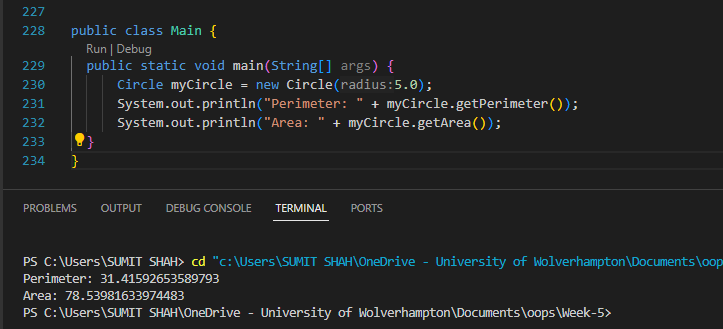
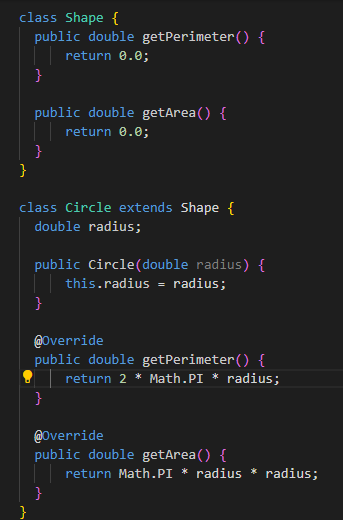
**Basic Inheritance(Single Inheritance):**

1. Create a class Animal with properties like name and age. Create a subclass Dog that inherits from Animal and adds a property breed. Demonstrate the use of constructors in both the Animal and Dog classes.



**Method Overriding:**

1. Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle



**Super Keyword:**

1. Extend the Animal and Dog example by adding a constructor to the Animal class that takes a name parameter. In the Dog class, use the super keyword to call the constructor of the Animal class. Create instances of Dog and demonstrate the use of the super keyword.

**Protected Keyword:**

1. Create a class Person with a protected attribute **address**. Extend it with a subclass Employee that adds a **department** attribute. Demonstrate how the protected keyword allows access to the address property in the Employee subclass.

**Access Modifiers and Inheritance:**

1. Create class Parent with a private variable, a protected variable, and a public variable. Create a subclass Child and demonstrate how each type of variable is accessed (or not accessed) within the subclass.

**Final Classes and Methods:**

1. Create a final class FinalClass. Attempt to extend it with another class and observe the compiler error. Also, create a final method within a class and try to override it in a subclass.

**Method overloading**

1. Create a class ,’Calculator’ that should be able to perform addition operations for both integers and doubles. Implement the following steps:
2. Create an instance of the Calculator class.
3. Use the add method to add two integers (e.g., 5 and 8) and display the result.
4. Call the add method with three integers (e.g., 10, 15, and 20) and display the result.
5. Use the add method to add two doubles (e.g., 3.5 and 2.7) and display the result.
6. Call the add method with three doubles (e.g., 1.1, 2.2, and 3.3) and display the result.

Section -2

Case Study

Online ordering has enabled many restaurants to manage their peak business hours very effectively. Thanks to online ordering, many people manage to prevent the painful experience of wasting time in a long queue.

AZA is one of the biggest restaurant chains in the United Kingdom. They decided to offer their customers a convenient and contactless mobile ordering solution in response to the growing business need and COVID restrictions.

As part of the development team in the SAS software solutions, you are required to design and develop the mobile ordering program.

Before the application release deadline, you are required to submit the following deliverables:

1. Java program for the two tasks as per the description that follows in this document.

Program Description:

In this program you have to create a signup process for the mobile ordering application for a restaurant. The restaurant has a variety of cuisines to offer their customers.

When the program starts, the user is given the following options:

1. Sign up
2. Quit Application

Output:

|  |
| --- |
| Please enter 1 for Sign up. Please enter 2 for Quit. |

Your program should keep running and enable multiple users to sign up until the quit application option is selected.

Output: If User enters 3

|  |
| --- |
| Thank you for using the Application. |

# Criteria

To start using the mobile app, users should sign up for an account. You are required to create a Java program for the signup process. The users will be asked to enter their full name, contact number, date of birth, password, and password confirmation.

1. The signup process must not be successful until:
   1. The full name is enter. The length must be greater than four.
   2. The mobile number has 10 digits starting with 0.
   3. The Password must initiate with alphabets followed by either one of @, & and ending with numeric.  
      (For Example: John@0125 or John&25) .
   4. The password confirmation matches the initial entered password.
   5. The DOB is in the format DD/MM/YYYY or MM/DD/YYYY.
   6. The user is at least 21 years old. The age should be calculated based on the year entered in the DOB (Only consider year).
2. If any of the above-mentioned conditions is not fulfilled; the sign-up process should fail, and a descriptive message should be displayed for the user explaining what has gone wrong and providing hints on the correct expected input. The program should keep asking the user to re-enter his details as long as one or more of the input fields are not correctly entered. If all fields are entered successfully, the program should stop asking the user to re-enter his details and display a message that the signup process has been completed successfully.
3. If any field is entered incorrectly, some examples of sample outputs are given below.

Output 1:

|  |
| --- |
| You have entered the Date of Birth in invalid format.  Please start again. |

Output 2:

|  |
| --- |
| Your passwords are not matching.  Please start again. |

1. If all of the above-mentioned conditions are successful, the user data is saved in appropriate data structure (Hint: Arrays can be used) to enable data checks during the login process in future builds.

Output 1:

|  |
| --- |
| Please enter your full name: Sam Jahn  Please enter your mobile number (username): 0445544455  Please enter your password: John@21  Please confirm your password: John@21  Please enter your Date of Birth #DD/MM/YYYY (No space): 21/01/1984  You have successfully signed up.  Please enter 1 for Sign up. Please enter 2 for Quit. |