05/10/2023

▼ Java Objects

- In OOP class `objects` are used to connect classes together and through the objects methods and variables will get called.
- Using class objects we are capable of passing values in between classes as well.

▼ Object Creation Syntax

```
className object = new className();

// for Bicycle class
Bicycle sportBicycle = new Bicycle();

Bicycle touringBicycle = new Bicycle():
```

Task 01

Create a Java project which takes user’s name and the degree inside a separate class and displays the output within the same class.

```
import java.util.Scanner;

public class separate {
    public static void main(String[] args) {
        System.out.println("User Details:");

        separate objSeparate = new separate();

        objSeparate.inputUserDetails();
    }

    public void inputUserDetails() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.next();

        System.out.print("Enter your degree: ");
        String degree = scanner.next();

        System.out.println(name + " " + degree);
    }
}
```

Task 02

Create a Java application to check whether the user is eligible for voting once user's age passed as parameter to the separate class. Display the output inside the separate class.

```
// main.java
package org.example;
import java.util.Scanner;

// Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,
// then press Enter. You can now see whitespace characters in your code.
public class Main {
    public static void main(String[] args) {
        inputUser objEligibility = new inputUser();
        objEligibility.eligibility();
    }
}
```

```
// inputUser.java
package org.example;

import java.util.Scanner;

public class inputUser {

    public void eligibility(){
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.next();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        if (age >= 18) {
            System.out.println("eligible");
        } else {
            System.out.println("not eligible");
        }
    }
}
```

▼ Java Encapsulation

- In encapsulation OOP concept it discusses how to access the private variables by using other classes in OOP.
- Considering the theory of access specifiers these private variables are not capable of accessing by other classes.
- In OOP public methods will use to access these private variables and using them values inside the private variables can access.

💡 In encapsulation there are 2 public methods are commonly used which known as get and set methods.

💡 `Set` method will always set a value to the private variable while `get` method will return the value out of the class and provide access to other classes.

```
class Person {
  // private field
  private int age;

  // getter method
  public int getAge() {
    return age;
  }

  // setter method
  public int setAge(int age) {
    this.age = age;
  }
}

class Main {
  public static void main(String[] args) {

    //create an object of Person
    Person p1 = new Person();

    // change age using setter
    p1. setAge(24);
```

```
    // access age using getter
    System.out.println("My age is " + p1.getAge());
  }
}
```