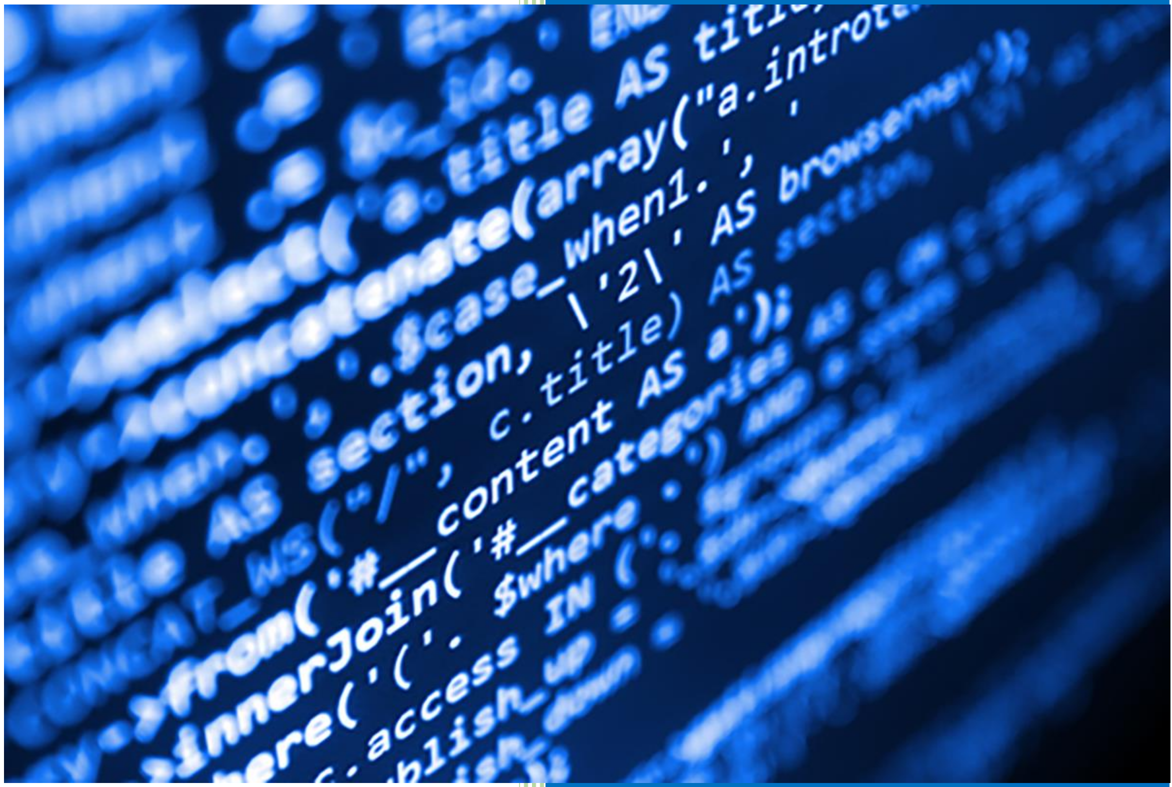
FAKULTI TEKNOLOGI KEJURUTERAAN KELAUTAN DAN INFORMATIK

2019/2020

DATA STRUCTURE & ALGORITHM



Lab 1: Object and Class

Name: OMAR ISMAIL ABDJALEEL ALOMORY

Matric Number: S63955

Lab: MP3

Date: 1/11/2022

VERSION 1

`

TABLE OF CONTENTS

`

INSTRUCTIONS

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Fakulti Teknologi Kejuruteraan Kelautan dan Informatik, Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual.

This laboratory manual is for use by the students of the Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual.

`

TASK 1: BASIC KNOWLEDGE ON OBJECT AND CLASS

OBJECTIVES

At the end of this lab, the students are able to

1. Write a class with instance variable fields of a user
2. Write constructors with and without parameters.
3. Access class members from member methods of the class.

ESTIMATED TIME

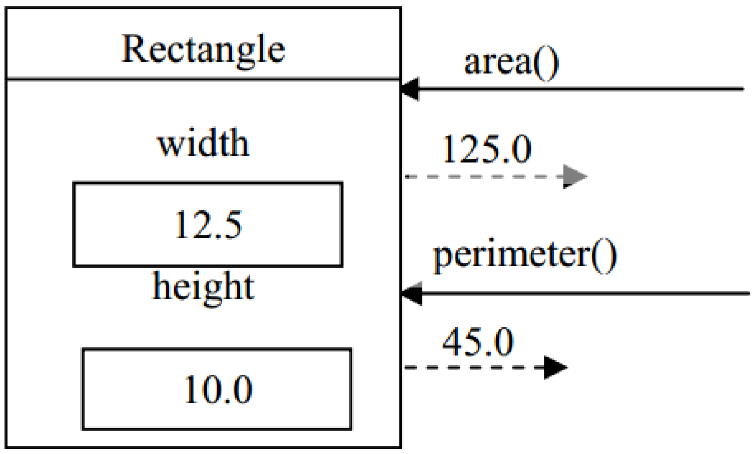
[120 Minutes]

STEPS:

In this lab, the `Rectangle` class will be defined.

A rectangle has a height and width, and given this data, the area and perimeter of a rectangle can be determined. Therefore, the class `Rectangle` should define data members: `height` and `width`. Since each `Rectangle` object stores its own values for these variables, height and width are instance variables. The methods are the method's that determine the area and perimeter of a `Rectangle` object and are, therefore, instance methods. The `Rectangle` class contains only instance variables and instance methods.

The following diagram is a diagram of a `Rectangle` object showing the instance variables width and height and the methods area and perimeter. The state of the object are the values of its data members. Since the state of this object has been set, also illustrated are the values returned when the area and perimeter methods are invoked on the object.



`

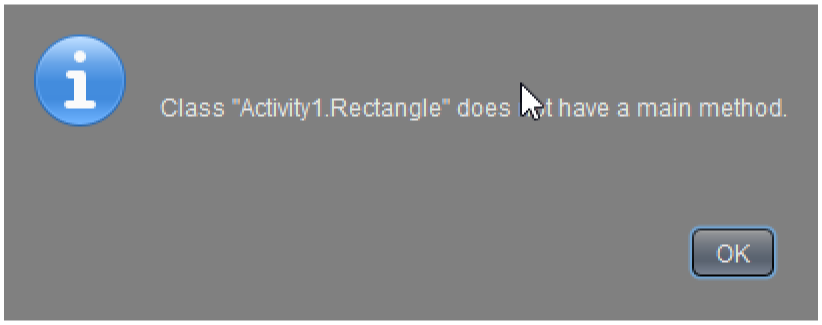
1.1 Testing the utility class

1. Run the following code and place as Rectangle.java

```
class Rectangle
{
      //instance variables
      double height, width;

      public double area()
      {
         double theArea;
         theArea = height * width;
         return theArea;
      }
}
```

2. Record the error message.



Class "Activity1.Rectangle" does not have a main method.

3. Why does the error happen?

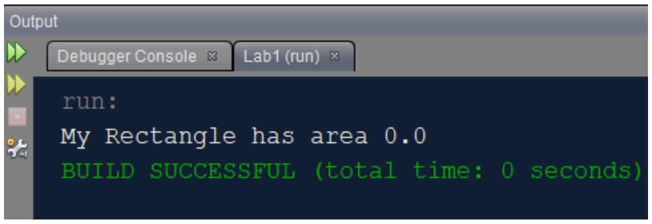
In order to run any project in java it must have main method, but this class still does not have it

4. Enter and save this in a file called `RectangleTest.java`.

```
class RectangleTest
{
   public static void main(String[] args)
   {
      Rectangle myRect = new Rectangle();
      double theAreamy = myRect.area();
      System.out.println("My rectangle has area " + myArea);
   }
}
```

`

5. Compile the program RectangleTest.java. Execute the program and record the results.



```
run:
My Rectangle has area 0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

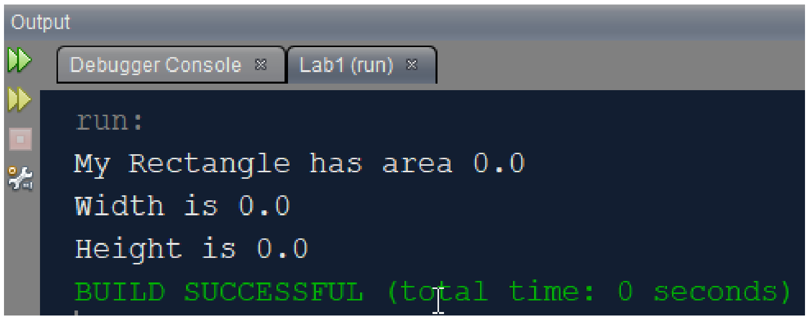
6. Currently, you can access the values of width and height directly by joining the variable to the name of the object using the dot operator. Add the following statements to the end of the main method.

```
System.out.println("Width is " + myRect.width);
System.out.println("Height is " + myRect.height);
```

Predict the output of the new program.

My Rectangle has area 0.0
Width is 0.0
Height is 0.0

7. Compile and execute the program. Record the results. Was your Step 5 prediction correct? If not, correct your answers.



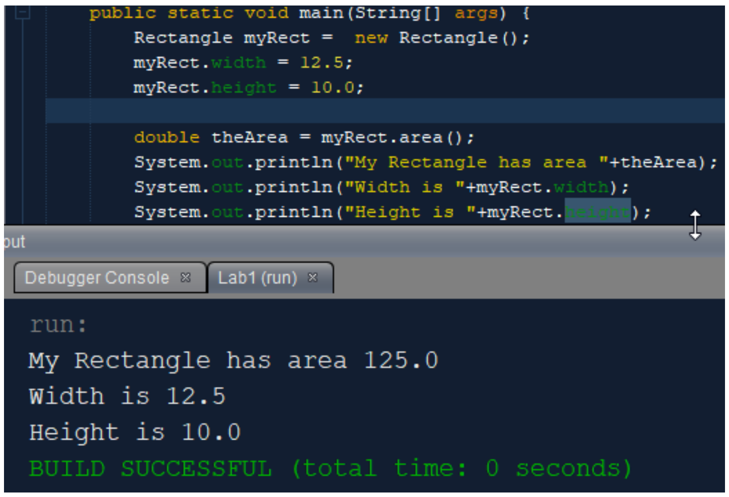
```
run:
My Rectangle has area 0.0
Width is 0.0
Height is 0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

`

8. Modify the program by adding these statements at an appropriate place in the main method so that the area of `myRect` is no longer `0`.

```
System.out.println("Width is " + myRect.width);
System.out.println("Height is " + myRect.height);
```

Compile and execute the program. Record the results.



```
public static void main(String[] args) {
    Rectangle myRect =  new Rectangle();
    myRect.width = 12.5;
    myRect.height = 10.0;

    double theArea = myRect.area();
    System.out.println("My Rectangle has area "+theArea);
    System.out.println("Width is "+myRect.width);
    System.out.println("Height is "+myRect.height);
```

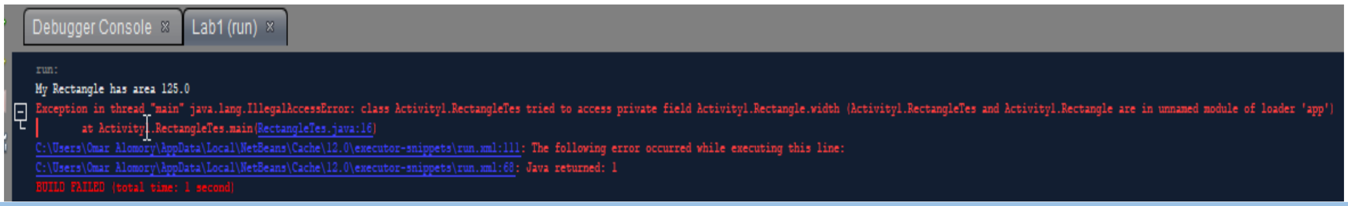
```
run:
My Rectangle has area 125.0
Width is 12.5
Height is 10.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.2 Access Modifier: `private`

9. Being able to directly access the instance variables of an object (`Rectangle`) from an outside class (`RectangleTest`) is considered to be an inappropriate practice in object-oriented languages. To prevent this, the instance variables of a class should be modified by the access modifier private. Modify the `Rectangle` class by inserting the private modifier in the data member declaration statement:

```
private double width, height;
```

Compile the modified program. Record the compiler error messages.



```
run:
My Rectangle has area 125.0
Exception in thread "main" java.lang.IllegalAccessError: class Activity1.RectangleTes tried to access private field Activity1.Rectangle.width (Activity1.RectangleTes and Activity1.Rectangle are in unnamed module of loader 'app')
        at Activity1.RectangleTes.main(RectangleTes.java:16)
C:\Users\Omar Alomory\AppData\Local\NetBeans\Cache\12.0\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Omar Alomory\AppData\Local\NetBeans\Cache\12.0\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 1 second)
```

`

1.3 Accessor Methods

10. In the Rectangle class, insert the code for the method `getWidth` which has no parameters and returns a `double`

```
public double getWidth()
{
    return width;
}
```

Compile the code. Then, insert a similar method to give the user, or client, access to the `height` of a `Rectangle`.

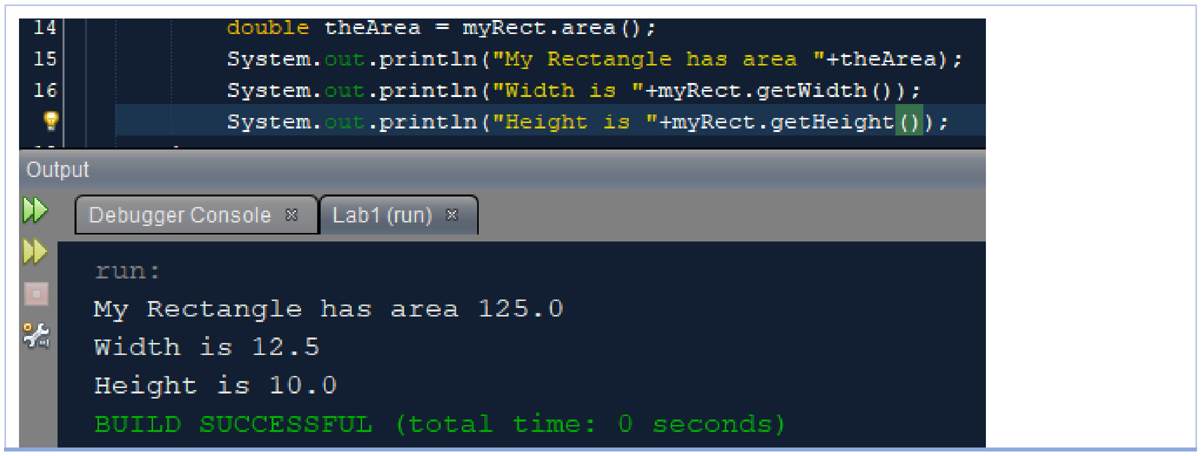
11. Modify the client class, `RectangleTest`, to correctly access the `width` and `height` of the `Rectangle` object.

This is an example of code to access the `width` of the `Rectangle` object.

```
System.out.println("Width is " + myRect.getWidth());
```

Insert the similar code to access the `height` of the `Rectangle` object.

Compile `RectangleTest.java` and execute the program. Record the results.



```
        double theArea = myRect.area();
        System.out.println("My Rectangle has area "+theArea);
        System.out.println("Width is "+myRect.getWidth());
        System.out.println("Height is "+myRect.getHeight());
```

```
run:
My Rectangle has area 125.0
Width is 12.5
Height is 10.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

`

1.4 Mutator Methods

12. The completed method should be added to the `Rectangle` class:

```
public void setWidth(double w)
{
   width = w;
}
```

Add the methods `setWidth` and `setHeight` to the `Rectangle` class. Make changes to `RectangleTest` to correctly use these methods. Your `Rectangle` class should now be:

```
class Rectangle
{
    private double width, height;

    public double area()
    {
        double theArea;
        theArea = height * width;
        return theArea;
    }

    public double getWidth()
    {
       return width;
    }

    public double getHeight()
    {
        return height;
    }

    public double setWidth(double w)
    {
        return width;
    }

    public void setHeight(double h)
    {
        height = h;
    }
 }
```

`

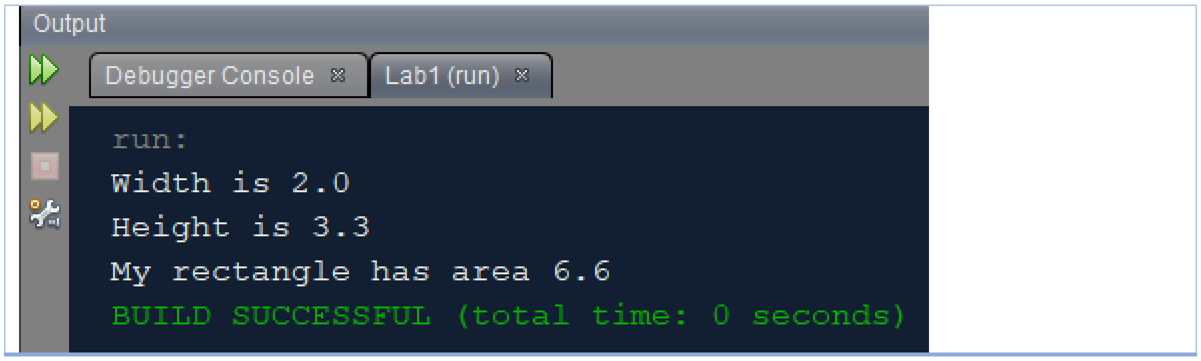
And, your `RectangleTest` class should now be:

```
class RectangleTest
{
     public static void main(String[] args)
     {
          Rectangle myRect = new Rectangle();

          myRect.setWidth(2.0);
          myRect.setHeight(3.3);
          double theArea = myRect.area();

          System.out.println("Width is " + myRect.getWidth());
          System.out.println("Height is " + myRect.getHeight());
          System.out.println("My rectangle has area " + theArea);
     }
}
```

Compile `RectangleTest`. Execute the program and record the results.



```
run:
Width is 2.0
Height is 3.3
My rectangle has area 6.6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Modify the `Rectangle` class by adding the constructor, placing it after the declaration of the instance variables, and before the definitions of the existing methods. This location is not mandatory, but it makes the code more readable.

Also, do the following to modify the class `RectangleTest` class

a. Comment out the two statements in main that invoke the set methods.
b. Change the statement that creates the `Rectangle` object from

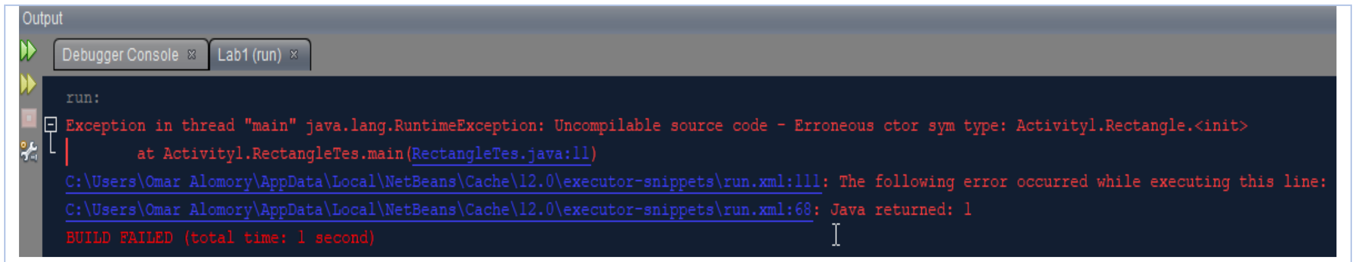
```
Rectangle myRect = new Rectangle();
```

to

```
Rectangle myRect = new Rectangle(12.5, 10);
```

`

Compile and execute the modified program. Record the results



```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous ctor sym type: Activity1.Rectangle.<init>
        at Activity1.RectangleTes.main(RectangleTes.java:11)
C:\Users\Omar Alomory\AppData\Local\NetBeans\Cache\12.0\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Omar Alomory\AppData\Local\NetBeans\Cache\12.0\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 1 second)
```

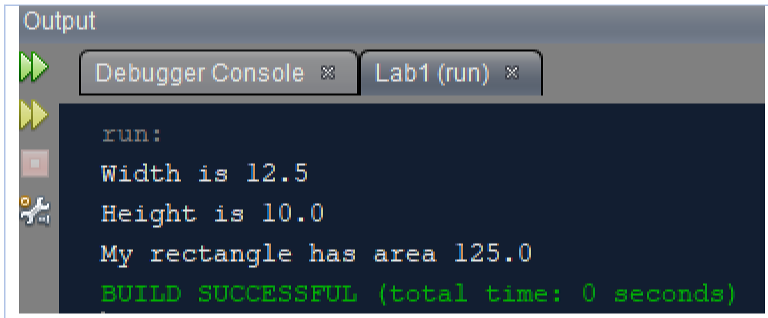
1.5 Writing a Constructor

13. A constructor must have the same name as the class name. Therefore, a constructor used to construct a `Rectangle` object, must be named Rectangle. We say that a constructor is a special type of method because it does not have a return type and because it can only be used in conjunction with the new operator. A constructor that initializes the height and width of a `Rectangle` object would take the form

```
public Rectangle(double w, double h)
{
    width = w;
    height = h;
}
```

Add the constructor to the `Rectangle.java`.

Compile and execute the modified program. Record the results



```
run:
Width is 12.5
Height is 10.0
My rectangle has area 125.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

`

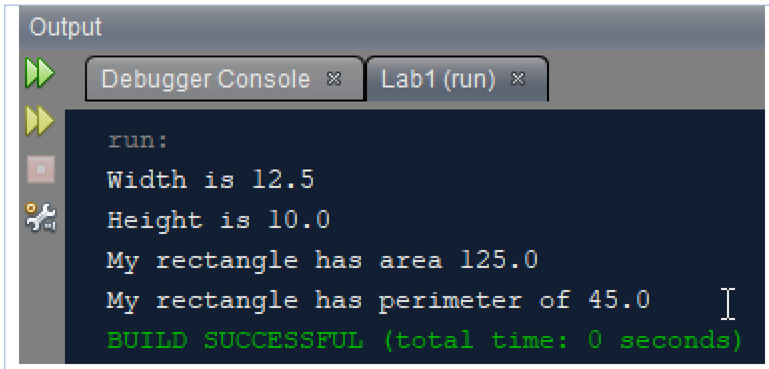
1.6 Completing the Rectangle class

Now, complete the `Rectangle` class by

- Adding a method to find the perimeter of a Rectangle object with the header that calculates and returns the perimeter of the rectangle.

```
public double perimeter()
```

Compile and execute the modified program. Record the results



```
run:
Width is 12.5
Height is 10.0
My rectangle has area 125.0
My rectangle has perimeter of 45.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

`

TASK 2: POST-LABORATORY PROBLEM

OBJECTIVE

To test the understanding of basic concepts and terminologies in object and class.

TASK DESCRIPTION

This task is basically to test the understanding of basic concepts and terminologies in object and class.

ESTIMATED TIME

[60 Minutes]

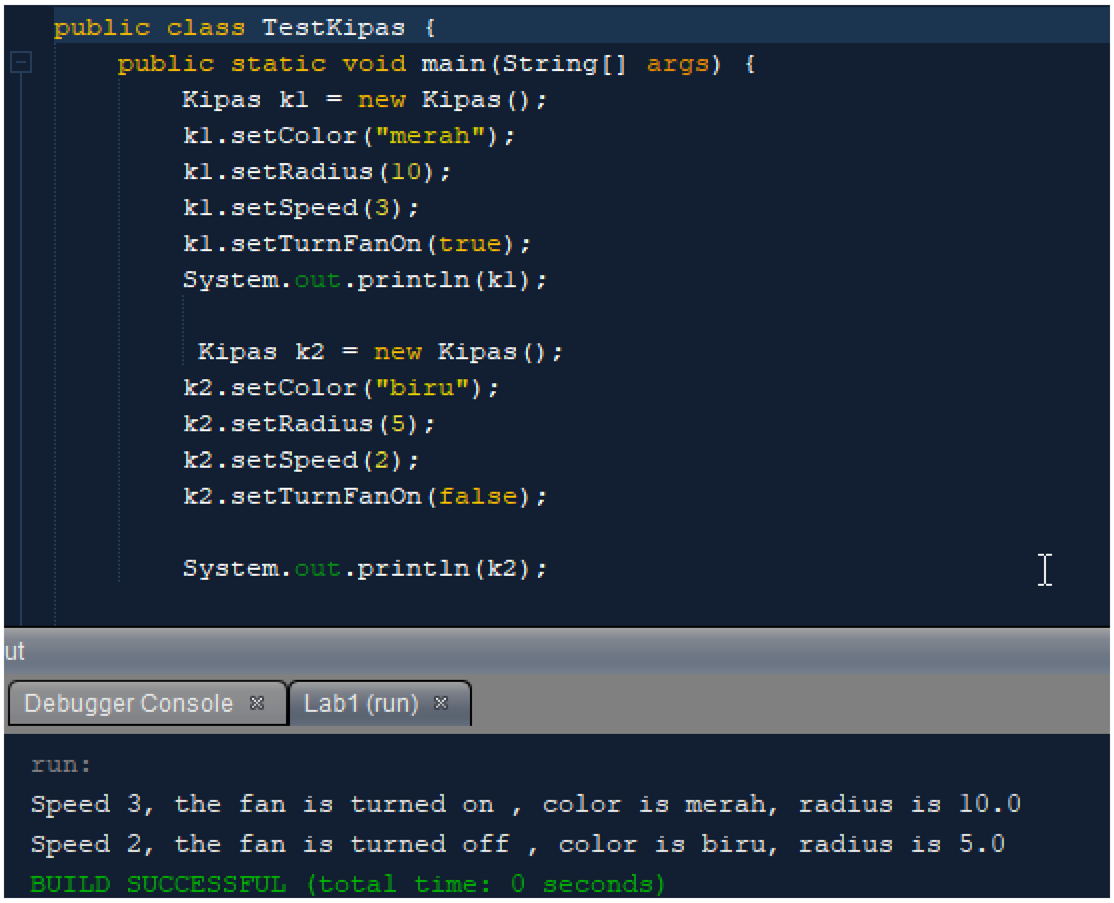
Design a class named `Kipas` to represent a fan. The class contains:

a. Three constants named `PERLAHAN`, `SEDERHANA`, and `LAJU` with the values 1, 2, and 3 to denote the fan speed.
b. A `private int` data field named speed that specifies the speed of the fan (the default is `PERLAHAN`).
c. A private `boolean` data field named on that specifies whether the fan is on (the default is false).
d. A `private double` data field named `radius` that specifies the radius of the fan (the default is 5).
e. A string data field named `colour` that specifies the colour of the fan (the default is `biru`).
f. The accessor and mutator methods for all four data fields.
g. A no-arg constructor that creates a default fan.
h. A method named `toString()` that returns a string description for the fan. If the fan is on, the method returns the fan `speed`, `colour`, and `radius` in one combined string. If the fan is not on, the method returns the fan color and radius along with the string "`fan is off`" in one combined string.

   Write a test program that creates two `Fan` objects. Assign maximum speed, radius 10, color merah, and turn it on to the first object. Assign medium speed, radius 5, color biru, and turn it off to the second object. Display the objects by invoking their toString method.

Both `Fan.java` and `TestFan.java` should be submitted via **epembelajaran.**

`



```
public class TestKipas {
    public static void main(String[] args) {
        Kipas k1 = new Kipas();
        k1.setColor("merah");
        k1.setRadius(10);
        k1.setSpeed(3);
        k1.setTurnFanOn(true);
        System.out.println(k1);

         Kipas k2 = new Kipas();
        k2.setColor("biru");
        k2.setRadius(5);
        k2.setSpeed(2);
        k2.setTurnFanOn(false);

        System.out.println(k2);
```

```
run:
Speed 3, the fan is turned on , color is merah, radius is 10.0
Speed 2, the fan is turned off , color is biru, radius is 5.0
BUILD SUCCESSFUL (total time: 0 seconds)
```