**Practical No. 19: Develop a program for implementation of multiple inheritance.**

1. **Practical Significance:**

Deriving a new class form multiple super/parent classes is known as multiple inheritance. Java doesn’t allow multiple inheritance to avoid the ambiguity caused by the overriding methods of the classes. Interface is used to achieve multiple inheritance in java.

1. **Relevant Course Outcome:**

Apply concept of inheritance for code reusability.

1. **Practical Outcome:**

Develop program for implementation of multiple inheritance.

1. **Minimum Theoretical Background:**
2. **Interfaces:**

Java does not support multiple inheritance.

Classes in java cannot have more than one superclass.

A large number of real-time applications requires use of multiple inheritance where no. of methods are inherited form several classes. Multiple inheritance proves difficult and adds complexity to the language. Java provides an alternative approach called as interfaces to support to the concept of multiple inheritance.

**Defining Interfaces**

An interface is like a class. Interfacesalso contain methods and variables but with a major difference. The difference is that interfaces define only abstract methods and final fields.

Interfaces do not specify any code to implement these methods and data fields contain only constants.

**Syntax:**

Interface InterfaceName

{

Variables declaration;

Methods declaration;

}

Where interface is the keyword and InterfaceName is any valid java variables.

Variable are declared as:

Static final type VariableName = Value;

Example: return\_type methodName1(parameter\_list);

**Interface Definition:**

Interface Item

{

static final int id = 100;

static final Strinf name = “ABC”;

void display();

}

**Extending Interfaces**

Interfaces can be extended. The new subinterface will inherit all the members of the subinterface.

Interface name2 extends name1

{

body of name2

}

**Example:**

interface ItemConstants

{

All constants in one interface

int id = 100;

String name = “ABC”;

}

interface Item extends ItemConstants

{

All methods in other interface

void display();

}

1. **Multiple Interfaces:**

interface ItemConstants

{

int id = 100;

String name = “ABC”;

}

interface ItemMethods

{

void display();

}

interface Item extends ItemConstants, ItemMethods

{

………………………………….

………………………………….

}

1. **Implementing Interfaces:**

Interfaces are used like superclasses whose features/properties are inherited by classes.

It is mandatory to define a class that inherits the given interface.

class classname implements interfacename

{

body of classname

}

The class classname “implements” the interface interfacename.

class classname extends superclass implements interface1, interface2,…

{

body of classname

}

1. **Program Code:**

**Example 1:**

interface A

{

void Student();

String ns = "Dhumal Aniket Harishchandra";

}

interface B

{

void Class();

String nc = "Java Programming";

}

interface C extends A, B

{

void Teacher();

String nt = "Alinka Shinde";

}

class D implements C

{

public void Student()

{

System.out.println("Student name is : " + ns);

}

public void Class()

{

System.out.println("Class name is : " + nc);

}

public void Teacher()

{

System.out.println("Teacher name is : " + nt);

}

}

class E extends D

{

public static void main(String[] agrs)

{

E Info = new E();

Info.Student();

Info.Class();

Info.Teacher();

}

}

**Example 2:**

interface A

{

void Student();

String ns = "Dhumal Aniket Harishchandra";

}

interface B

{

void Class();

String nc = "Java Programming";

}

interface C

{

void Teacher();

String nt = "Alinka Shinde";

}

class MIDemo implements A, B, C

{

public void Student()

{

System.out.println("Student name is : " + ns);

}

public void Class()

{

System.out.println("Class name is : " + nc);

}

public void Teacher()

{

System.out.println("Teacher name is : " + nt);

}

public static void main(String[] agrs)

{

E Info = new E();

Info.Student();

Info.Class();

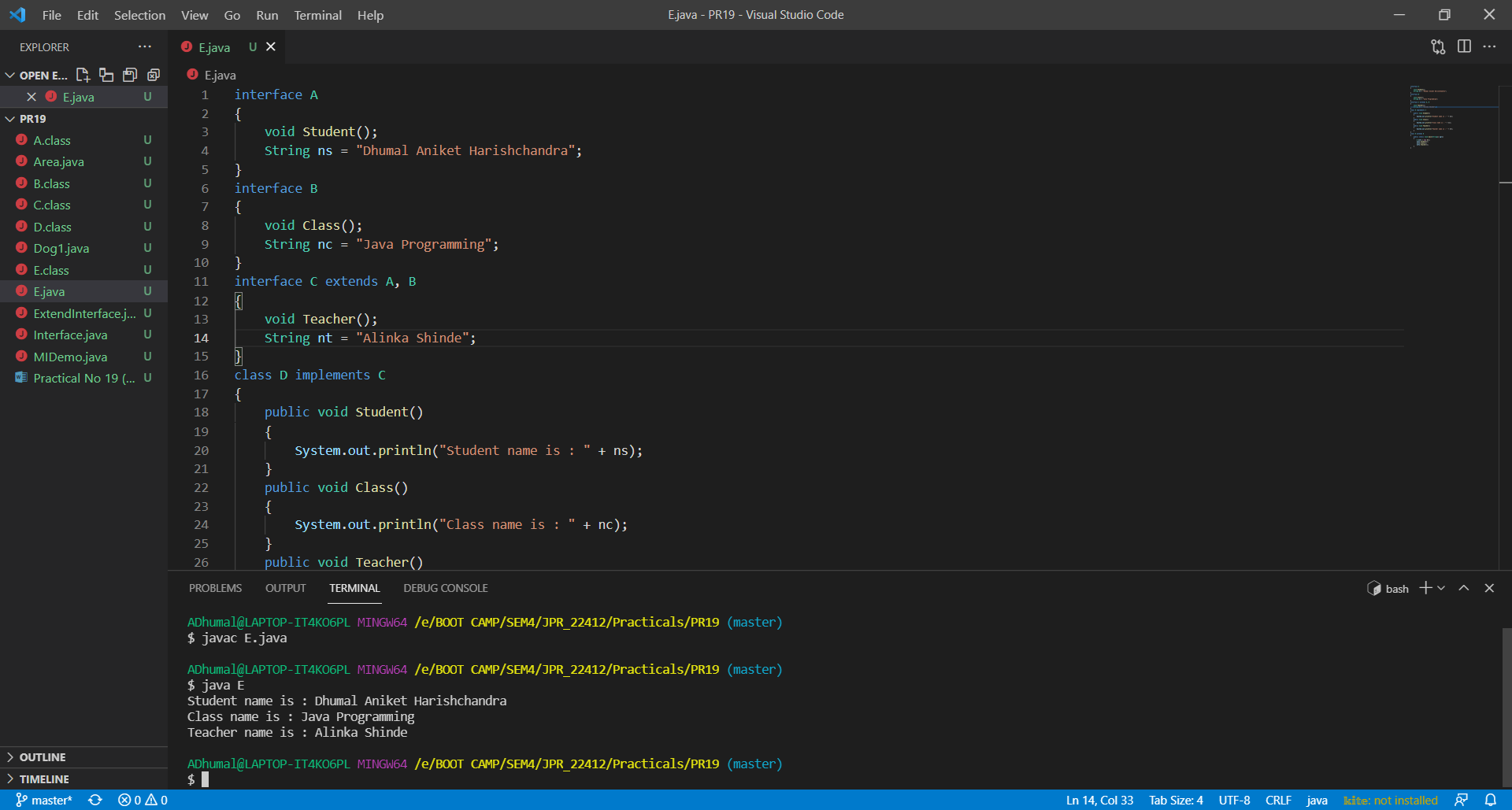
Info.Teacher();

}

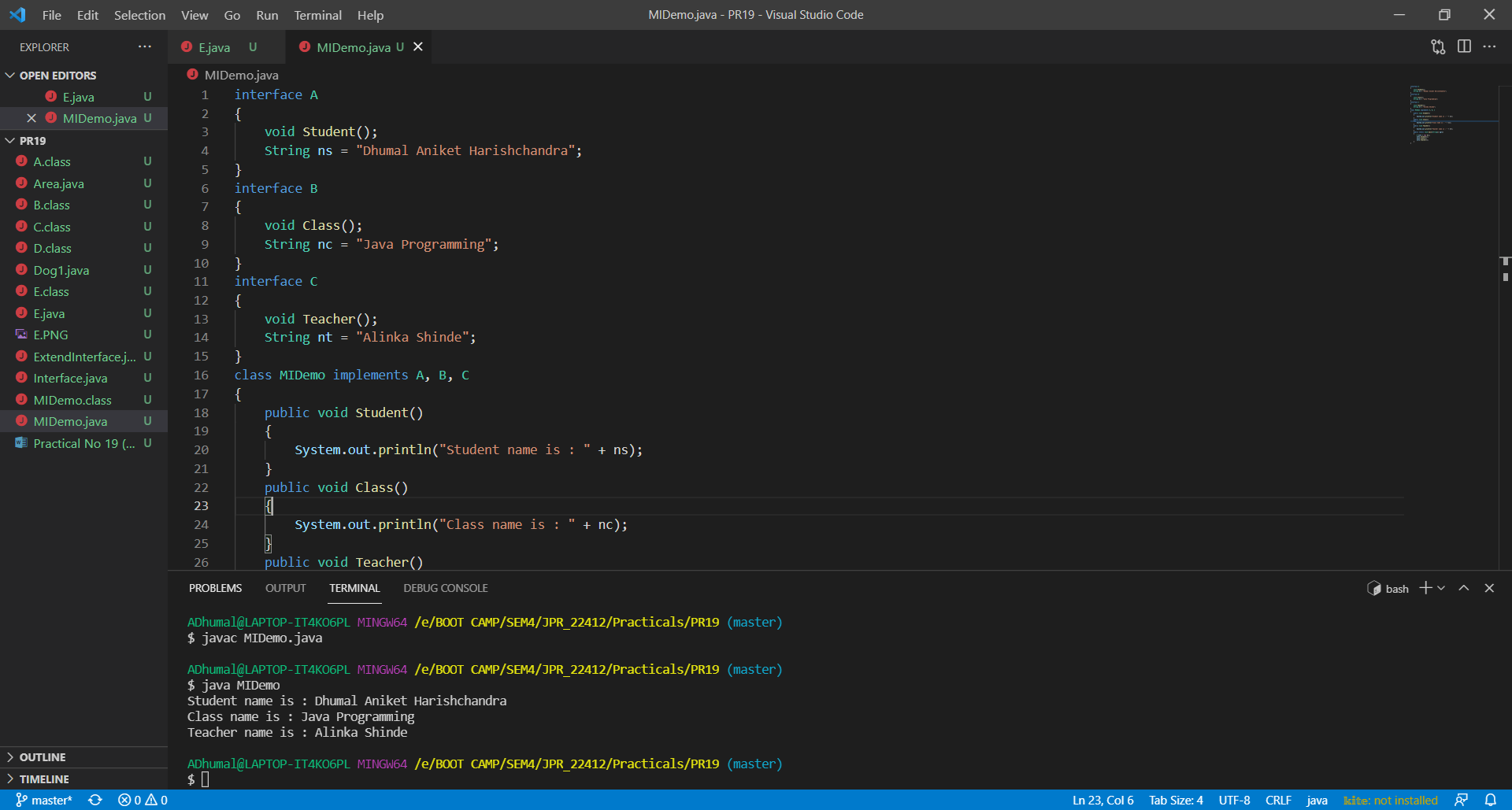
}

1. **Result:**

**Example 1:**



**Example 2:**



1. **Practical Related Questions:**
2. **Differentiate between Class and Interface.**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Class** | **Interface** |
| 1 | A class describes the attributes and behaviours of an object. | An interface contains behaviours that a class implements. |
| 2 | A class may contain abstract methods, concrete methods. | An interface contains only abstract methods. |
| 3 | Members of a class can be public, private, protected or default. | All the members of the interface are public by default. |

1. **Write similarities between interfaces and classes.**

* Both are java basic object types.
* Both can contain variables and methods (class methods have Implementation code whereas the interface methods can only have declarations).
* Both be inherited using Inheritance (extends keyword for class and implements keyword for interfaces).

1. **Advantages of interfaces.**

* It is used to achieve total abstraction.
* Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance.
* It is also used to achieve loose coupling.
* Interfaces are used to implement abstraction. So the question arises why use interfaces when we have abstract classes?

1. **Will this code compile successfully? (Assume Suitable Class)**

public interface FamousLine

{

void ShowLine()

{

System.out.println("Show Line");

}

}

**Assumed Suitable Class**

public interface FamousLine

{

void ShowLine()

{

System.out.println("Show Line");

}

}

class Interface implements FamousLine

{

public static void main(String agrs[])

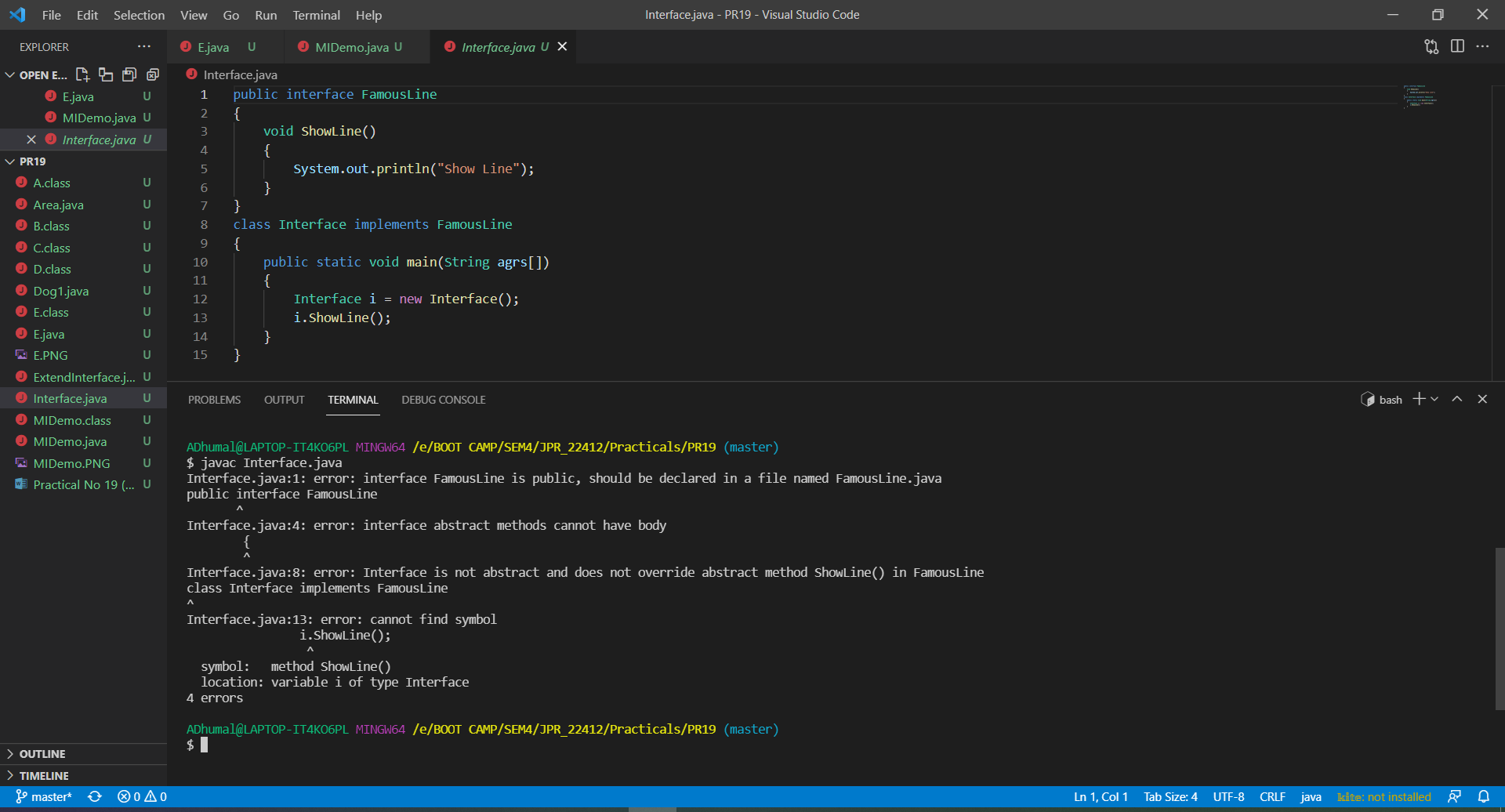
{

Interface i = new Interface();

i.ShowLine();

}

}



1. **Exercise:**
2. **Correct the code to rectify the compile time error thrown.**

interface NewShape

{

void draw();

}

interface Circle extends NewShape

{

void getRadius();

int radius = 10;

}

class NewCircle implements Circle

{

public void getRadius()

{

System.out.println(radius);

}

}

class ExtendInterface extends NewCircle

{

public static void main(String agrs[])

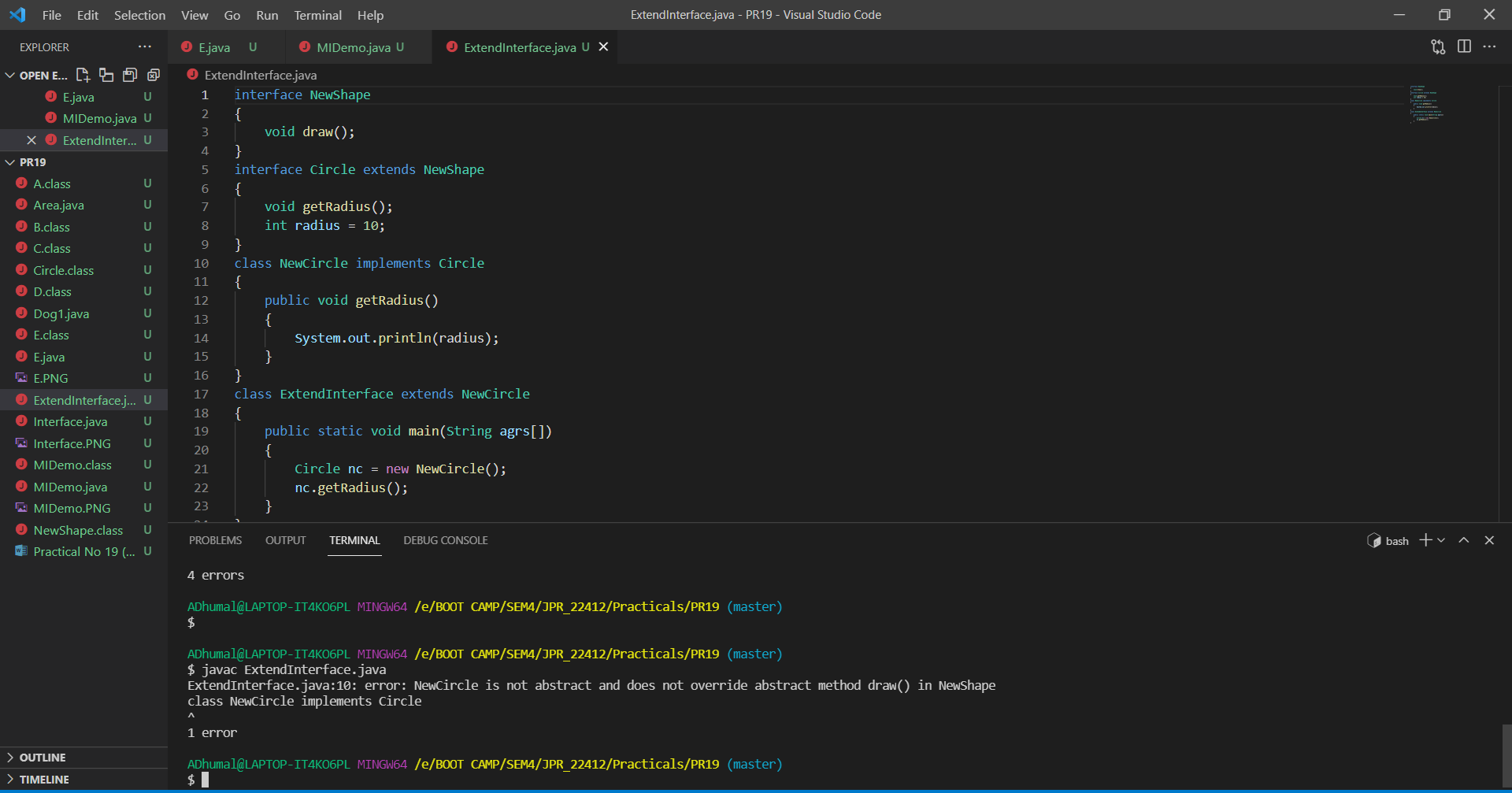
{

Circle nc = new NewCircle();

nc.getRadius();

}

}



interface NewShape

{

void draw();

}

interface Circle extends NewShape

{

void getRadius();

int radius = 10;

}

class NewCircle implements Circle

{

public void draw()

{

System.out.println("Draw a circle of radious 10cm.");

}

public void getRadius()

{

System.out.println(radius);

}

}

class ExtendInterface extends NewCircle

{

public static void main(String agrs[])

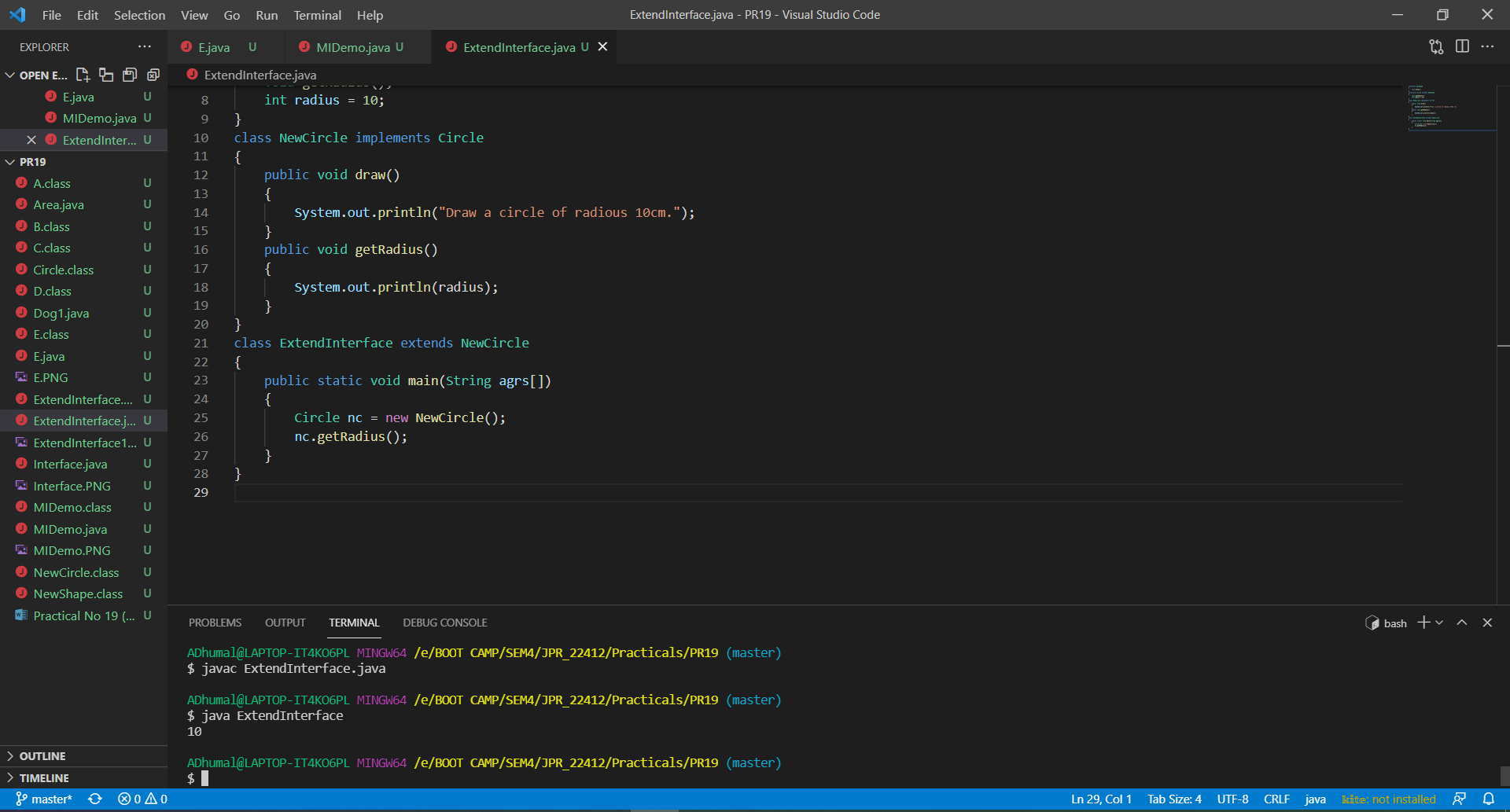
{

Circle nc = new NewCircle();

nc.getRadius();

}

}



1. **Develop a program to find area of rectangle and circle using interfaces.**

interface Circle

{

void CircleArea();

}

interface Rectangle

{

void RectangleArea();

}

class Area implements Circle, Rectangle

{

public void CircleArea()

{

int r = 17;

double pi = 3.14, area;

area = pi \* r\* r;

System.out.println("Radius is : " + r);

System.out.println("Area of circle is : " + area);

}

public void RectangleArea()

{

int l = 17, b = 24;

double area;

area = l \* b;

System.out.println("Length is : " + l);

System.out.println("Breadth is : " + b);

System.out.println("Area of rectangle is : " + area);

}

public static void main(String args[])

{

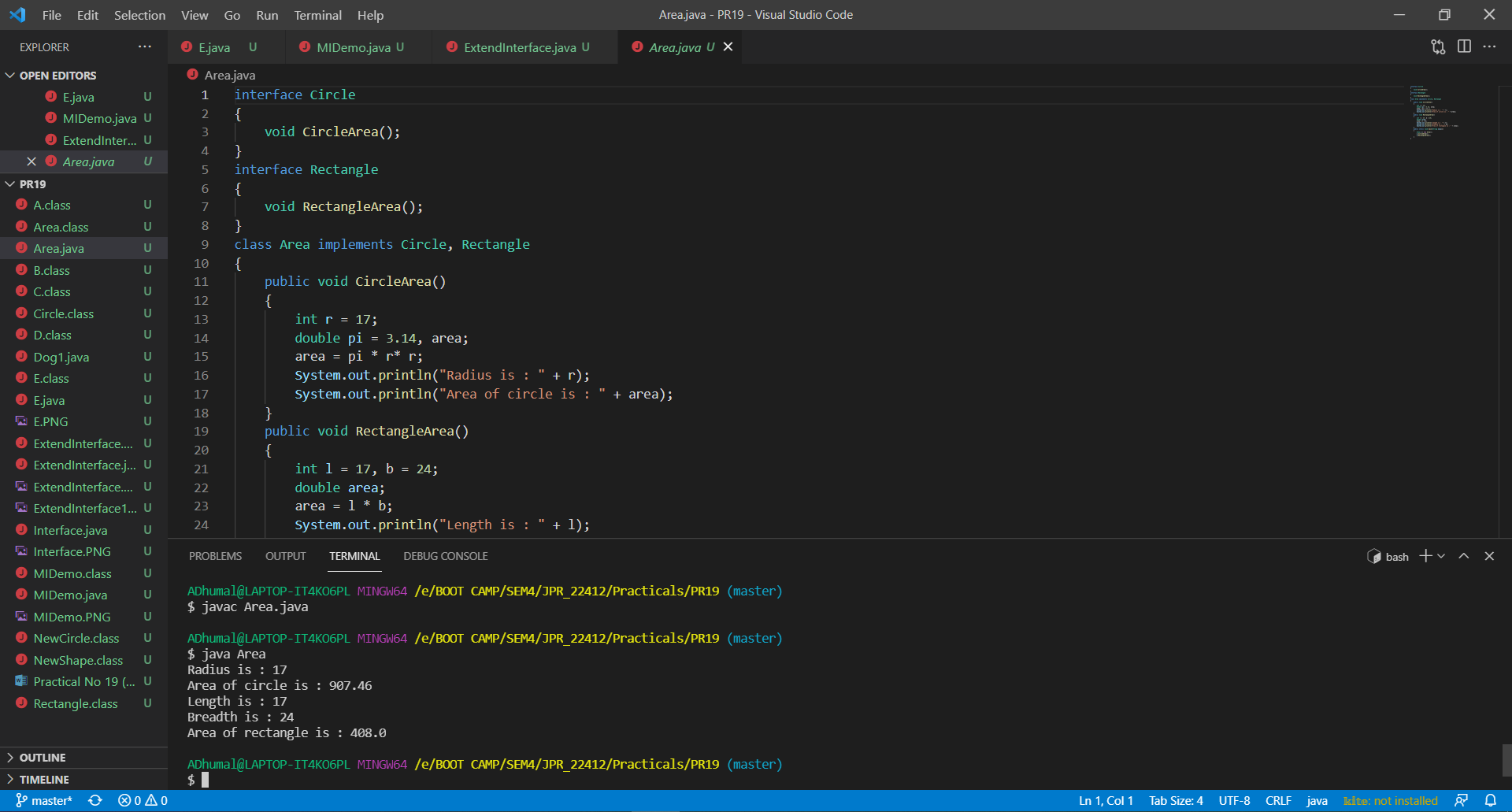
Area a = new Area();

a.CircleArea();

a.RectangleArea();

}

}



1. **Write output of the program.**

