

IQRA NATIONAL UNIVERSITY

FINAL ASSIGNMENT BS SOFTWARE ENGINEERING

NAME: AHMED JUNAID ID: 15815

SEMESTER 2nd BS(SE)

SUBJECT: Object Oriented Programming

Question1: A Why access modifiers are used in java, explain in detail Private and Default?

Access Modifiers:

As we know all the programming language has Access Modifiers but as Java is Object Oriented the declaration of Access Modifiers is Important for each field has An Access Modifier .we don't want some data to be controlled or manipulated by other classes

There are what is known as 4 different access Modifiers and each of them provides different accessibility

- 1- Default
- 2- Private
- 3- Protected
- 4- Public

I will go over Default and private in this question as the others are not asked according to the question :

Default Access Modifiers:

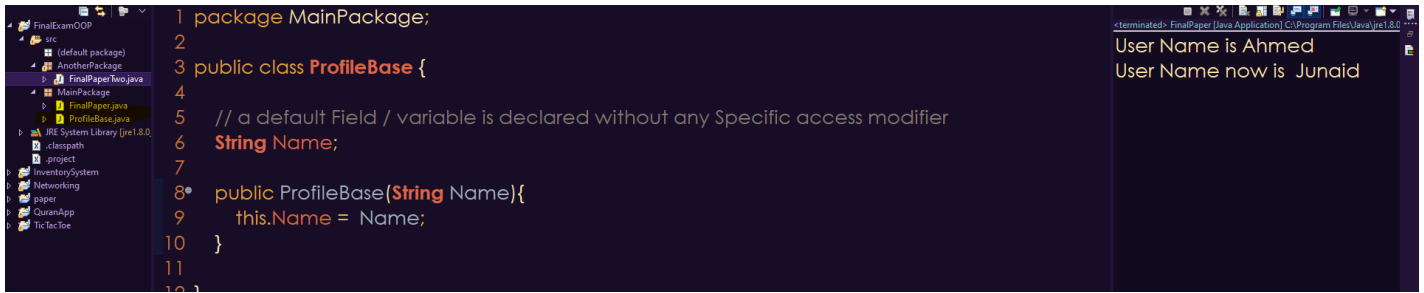
Default Assigned Access Modifiers are those data field or function which can be accessed in mostly everywhere except other packages classes and non-subclasses. A package is Kind of a folder which holds the classes together software can have many packages in java according to the need like for example if we were to build a java game using OpenGL API we would have different packages like for example input system with all the input controllers for the game and another package named Shaders which hold shader classes I will explain this in detail in the practical part of this question.

Private Access Modifiers:

Private Access Modifiers are those data field or function which can not be accessed outside the main Class body to put it simply the data which has private access is same as it sounds it is private sometime we don't want some data to be access by other classes so we make them private fields and private functions again I will go over this in practical

b. Write a specific program of the above mentioned access modifiers in java.

```
1 package MainPackage;
2
3 public class ProfileBase {
4
5     // a default Field / variable is declared without any Specific access modifier
6     String Name;
7
8     public ProfileBase(String Name){
9         this.Name = Name;
10    }
11
12 }
```

The screenshot shows an IDE window with a project named 'FinalExamOOP'. The 'MainPackage' folder contains 'FinalPaperTwo.java' and 'ProfileBase.java'. The code in 'ProfileBase.java' defines a class with a public field 'Name' and a constructor. The console output shows 'User Name is Ahmed' and 'User Name now is Junaid'.

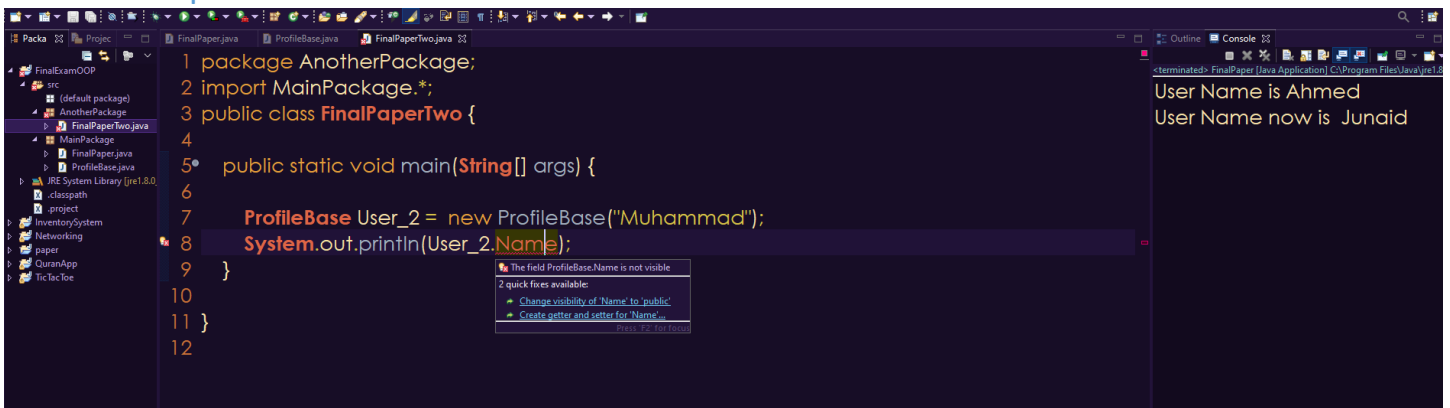
- 1- Firstly I go ahead and created our ProfileBase Class which has an Default Field Name
- 2- I made a constructor to take a String object as the parameter and assign it to the Name Default field

```
1
2 public class FinalPaper {
3
4     public static void main(String[] args) {
5
6         // Creating an object Based on the created Class
7         ProfileBase User_1 = new ProfileBase("Ahmed");
8
9         // Printing Constructor passed Name String
10        System.out.println("User Name is " + User_1.Name);
11
12        // we can see we can access this field directly like if was a Public field
13        User_1.Name = "Junaid";
14
15        //Printing out newly changed value
16        System.out.println("User Name now is " + User_1.Name);
17
18    }
19
20 }
```

The screenshot shows the 'FinalPaper' class in the 'FinalPaperTwo.java' file. It contains a main method that creates a 'ProfileBase' object with the name 'Ahmed', prints it, changes the name to 'Junaid', and prints it again. The console output shows 'User Name is Ahmed' and 'User Name now is Junaid'.

- 3- Then I went ahead and created our main FinalPaper class in which I initialized the object based on our class Profile and by passing the constructor name
- 4- Then we can see that we can access the field direct and change it just like a public field

```
1 package AnotherPackage;
2 import MainPackage.*;
3 public class FinalPaperTwo {
4
5     public static void main(String[] args) {
6
7         ProfileBase User_2 = new ProfileBase("Muhammad");
8         System.out.println(User_2.Name);
9     }
10
11 }
12
```

The screenshot shows the 'FinalPaperTwo' class in the 'AnotherPackage' folder. It imports 'MainPackage.*' and creates a 'ProfileBase' object with the name 'Muhammad', printing it. The console output shows 'User Name is Ahmed' and 'User Name now is Junaid'. An IDE error message is visible: 'The field ProfileBase.Name is not visible' with quick fixes: 'Change visibility of 'Name' to 'public'', 'Create getter and setter for 'Name'', and 'Press F1 for Focus'.

- 5- Now I went ahead and made another package named AnotherPackage in that package I created a class which runs just like FinalPaper class with main function and all
- 6- In the image above you can see we cant access Name field of that Object

Private Access Modifiers:

```

1 package MainPackage;
2
3 public class ProfileBase {
4
5 //Lets Make this Name field private by putting Private at the start
6 private String Name;
7
8 public ProfileBase(String Name){
9     this.Name = Name; // we can access it here
10 }
11
12 }

```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The field ProfileBase.Name is not visible
The field ProfileBase.Name is not visible
The field ProfileBase.Name is not visible
at MainPackage.FinalPaper.main(FinalPaper.java:12)

- 1- Now for the private example I will be using the same class Profile Base but this time I will mark the Name field Private
- 2- And as we can see we can access it in our constructor like it was before but this time in our main FinalPaper Class we see some interesting things

```

1 package MainPackage;
2
3 public class FinalPaper {
4
5
6 public static void main(String[] args) {
7
8 // Creating an object Based on the created Class
9 ProfileBase User_1 = new ProfileBase("Ahmed");
10
11 // Printing Constructor passed Name String
12 System.out.println("User Name is " + User_1.Name);
13
14 // we can see we can access this field directly as it is a public field
15 User_1.Name = "Junaid";
16
17 //Printing out newly changed value
18 System.out.println("User Name now is " + User_1.Name);
19
20 }
21
22 }
23

```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The field ProfileBase.Name is not visible
The field ProfileBase.Name is not visible
The field ProfileBase.Name is not visible
at MainPackage.FinalPaper.main(FinalPaper.java:12)

The field ProfileBase.Name is not visible
2 quick fixes available:
- Change visibility of 'Name' to 'package'
- Create getter and setter for 'Name'...

- 3- In this class I changed nothing and right away we can see now the output is filled with error saying The field ProfileBase.Name is not visible
- 4- So as I mentioned before it is private to its class body only

Q2. a. Explain in detail Public and Protected access modifiers?

Public Access Modifiers:

A public assigned Access Modifiers are those fields or methods which can be accessed in every class and subclass, it has permission as the name itself states public, simply put we have all the access of that particular data in all over the program more than the default which I explained earlier

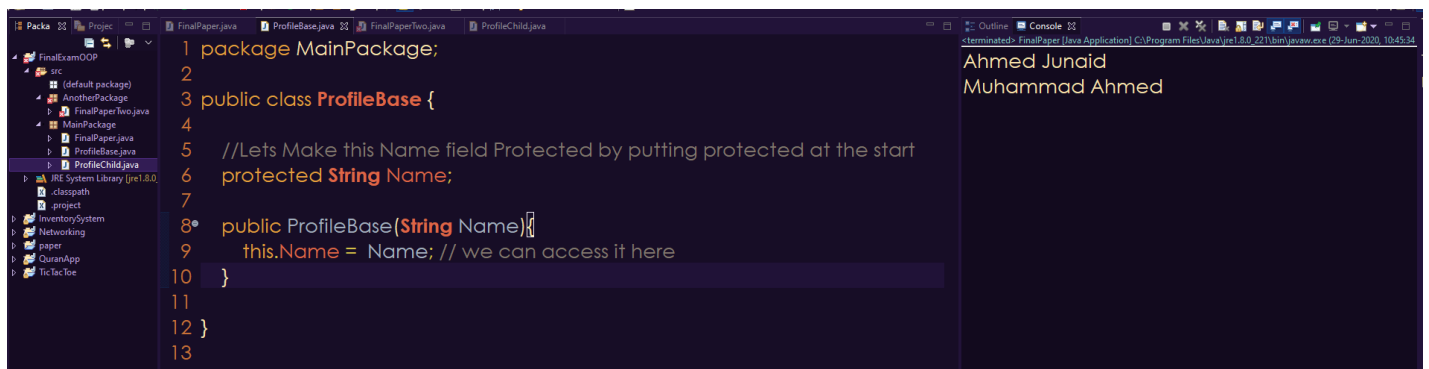
Protected Access Modifiers:

A protected assigned Access Modifiers are those Fields or methods which can be accessed by the classes of the same package which means other packages nonsubclass can't access this data, but the classes and function inside the same package can access it just like the public to put it simply A subclass or in other words inherited class can call this function can access it and it can be access via object but these fields or method are not accessible outside the package (if that make sense) I will make a program which will explain better

The only difference between this protected and public is protected are not accessed by outside package whereas the public is accessed by outside classes

b. Write a specific program of the above mentioned access modifiers in java.

Protected Access Modifier:

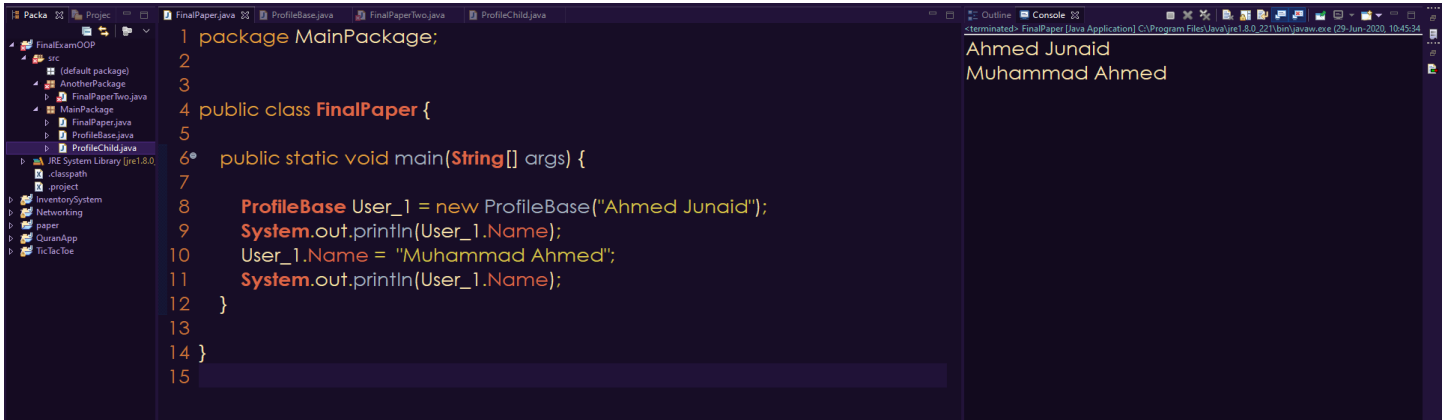


```
1 package MainPackage;
2
3 public class ProfileBase {
4
5     //Lets Make this Name field Protected by putting protected at the start
6     protected String Name;
7
8     public ProfileBase(String Name){
9         this.Name = Name; // we can access it here
10    }
11
12 }
13
```

The screenshot shows an IDE with a project named 'FinalExamOOP'. The 'MainPackage' folder contains 'FinalPaper.java', 'ProfileBase.java', and 'ProfileChild.java'. The 'ProfileBase.java' file is open, showing the code above. The console on the right displays the output: 'Ahmed Junaid' and 'Muhammad Ahmed'.

1- First I created the same Example class ProfileBase init I declared a Protected String Object Called Name

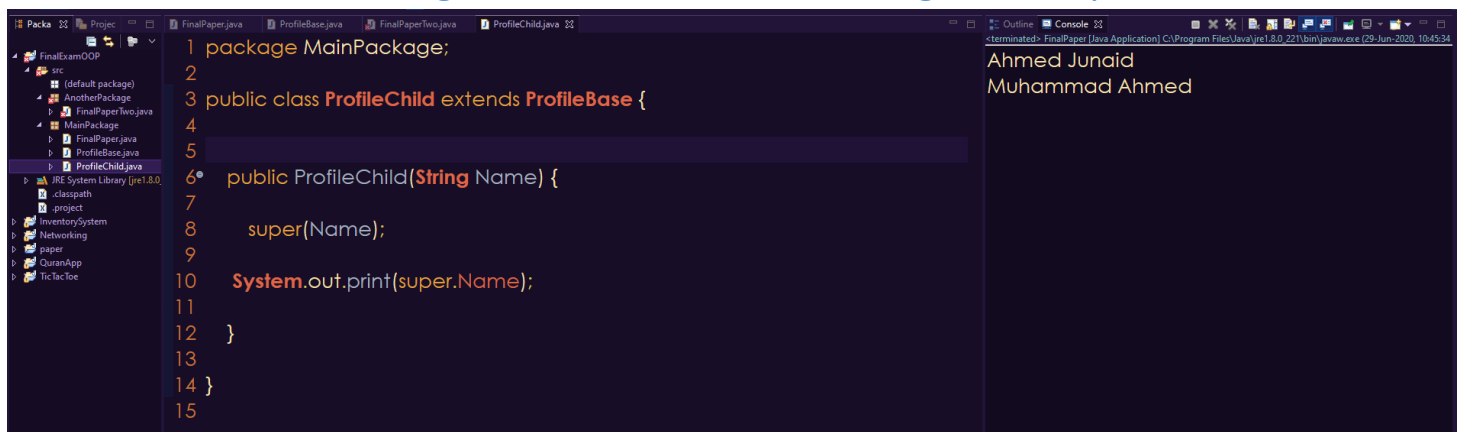
2- Inside the constructor I can access it



```
1 package MainPackage;
2
3
4 public class FinalPaper {
5
6     public static void main(String[] args) {
7
8         ProfileBase User_1 = new ProfileBase("Ahmed Junaid");
9         System.out.println(User_1.Name);
10        User_1.Name = "Muhammad Ahmed";
11        System.out.println(User_1.Name);
12    }
13
14 }
15
```

Ahmed Junaid
Muhammad Ahmed

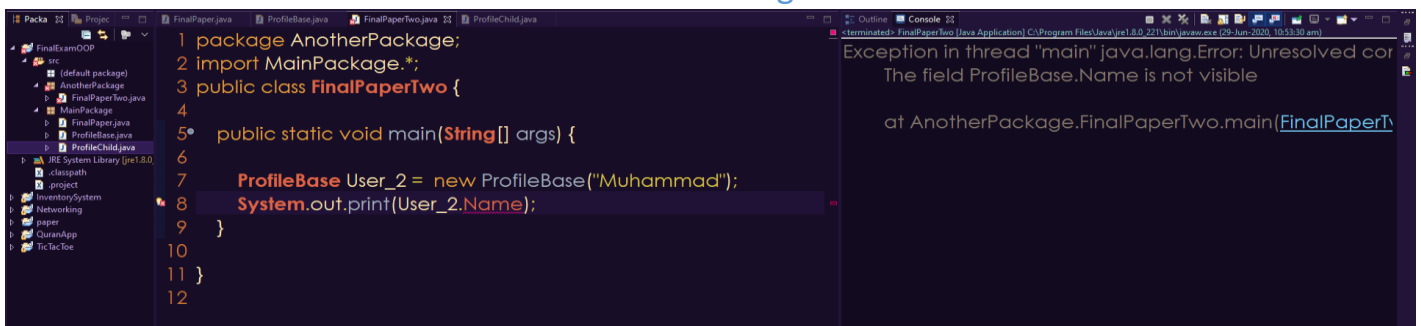
3- Now inside of main FinalPaper class which is inside the same package MainPackage I can access and able to change it directly



```
1 package MainPackage;
2
3 public class ProfileChild extends ProfileBase {
4
5
6     public ProfileChild(String Name) {
7
8         super(Name);
9
10        System.out.print(super.Name);
11    }
12
13
14 }
15
```

Ahmed Junaid
Muhammad Ahmed

4- Even in the inherited class I can access it and see it by calling super class and Name but inside the other Package

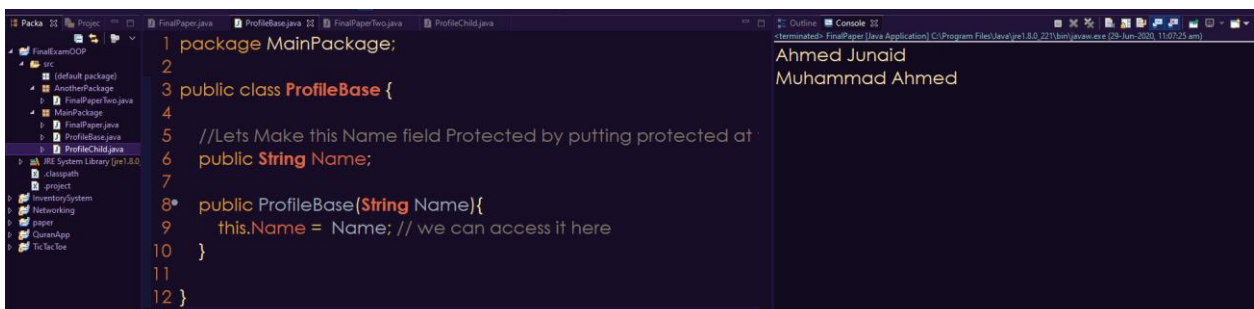


```
1 package AnotherPackage;
2 import MainPackage.*;
3 public class FinalPaperTwo {
4
5     public static void main(String[] args) {
6
7         ProfileBase User_2 = new ProfileBase("Muhammad");
8         System.out.print(User_2.Name);
9     }
10
11 }
12
```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The field ProfileBase.Name is not visible
at AnotherPackage.FinalPaperTwo.main(FinalPaperTwo.java:8)

5- Inside this class FinalPaperTwo Which is inside another package we can not access the Name field as the error in the console says it is not visible

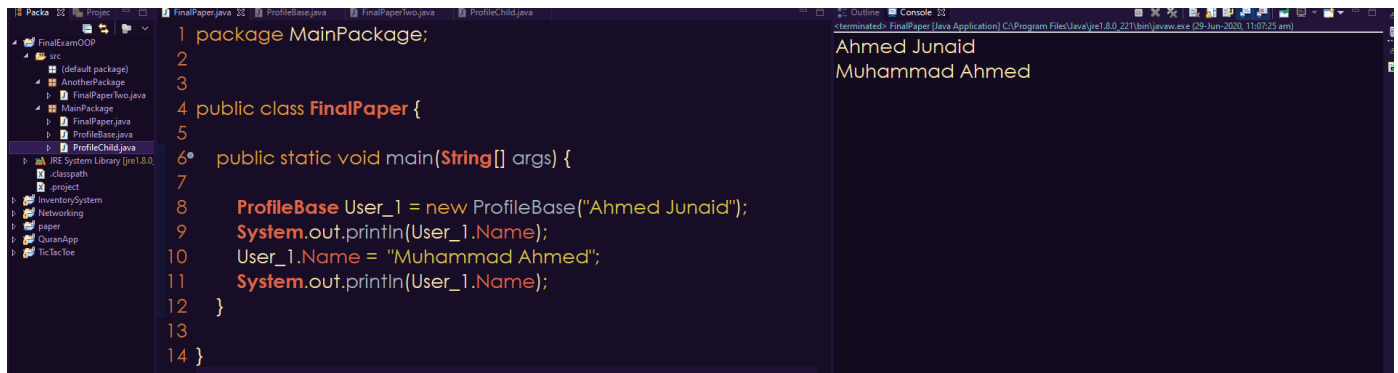
Public Access Modifier:



```
1 package MainPackage;
2
3 public class ProfileBase {
4
5     //Lets Make this Name field Protected by putting protected at
6     public String Name;
7
8     public ProfileBase(String Name){
9         this.Name = Name; // we can access it here
10    }
11
12 }
13
```

Ahmed Junaid
Muhammad Ahmed

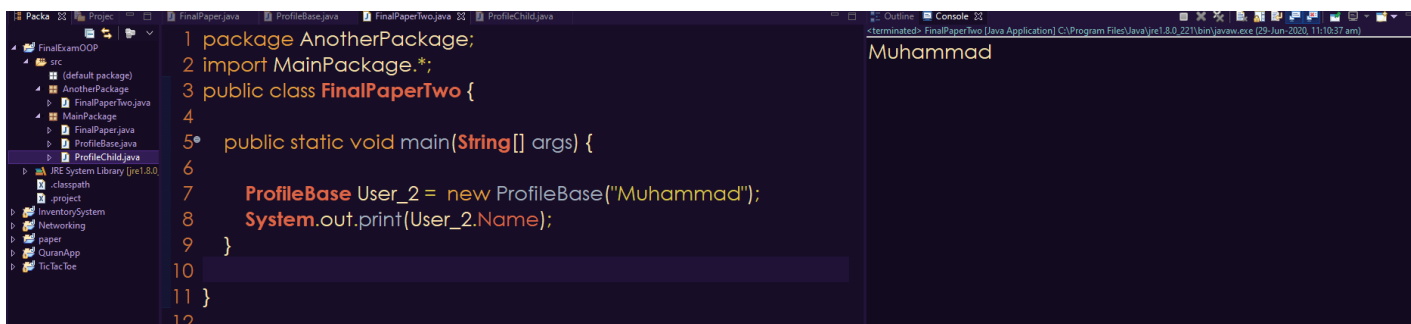
- 1- I Just changed the access modifier of the previous example to public instead of protected now as we can see we can access it from within the class



```
1 package MainPackage;
2
3
4 public class FinalPaper {
5
6     public static void main(String[] args) {
7
8         ProfileBase User_1 = new ProfileBase("Ahmed Junaid");
9         System.out.println(User_1.Name);
10        User_1.Name = "Muhammad Ahmed";
11        System.out.println(User_1.Name);
12    }
13
14 }
```

Ahmed Junaid
Muhammad Ahmed

- 2- And as an object we can still edit it within the same package



```
1 package AnotherPackage;
2 import MainPackage.*;
3 public class FinalPaperTwo {
4
5     public static void main(String[] args) {
6
7         ProfileBase User_2 = new ProfileBase("Muhammad");
8         System.out.print(User_2.Name);
9     }
10
11 }
12
```

Muhammad

- 3- And we can access it from within another class in an Another package as well and can change it

Q3. a. What is inheritance and why it is used, discuss in detail ?

Inheritance:

Inheritance concept is the same as it is an inheritance of something to the child. In OOP inherited child class has the functions of the parent class or also known as SuperClass, we use the keyword extends followed by the superclass name

Now to make it clear let's take an example of the superclass of wood

Superclass wood has few property/variables like for example

Type, hardness, color and a function stats to display all this

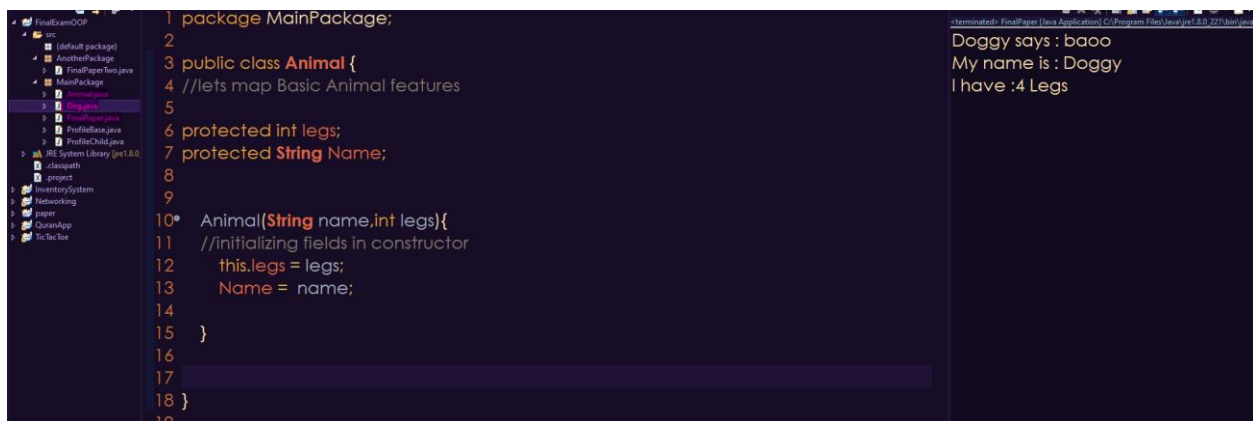
Now let's imagine an inherited class House which extends from wood class

Now that house has the values of the wood class like hardness and type and color however we can add more properties according to house object like windows,

doors etc etc but wood property still exist if we say satus on that House class we can invoke the function of its superclass wood which will display its value

This is the concept of inheritance in OOP and is very usefull as the example above explains how usefull that is again this is one of many uses of inheritance.

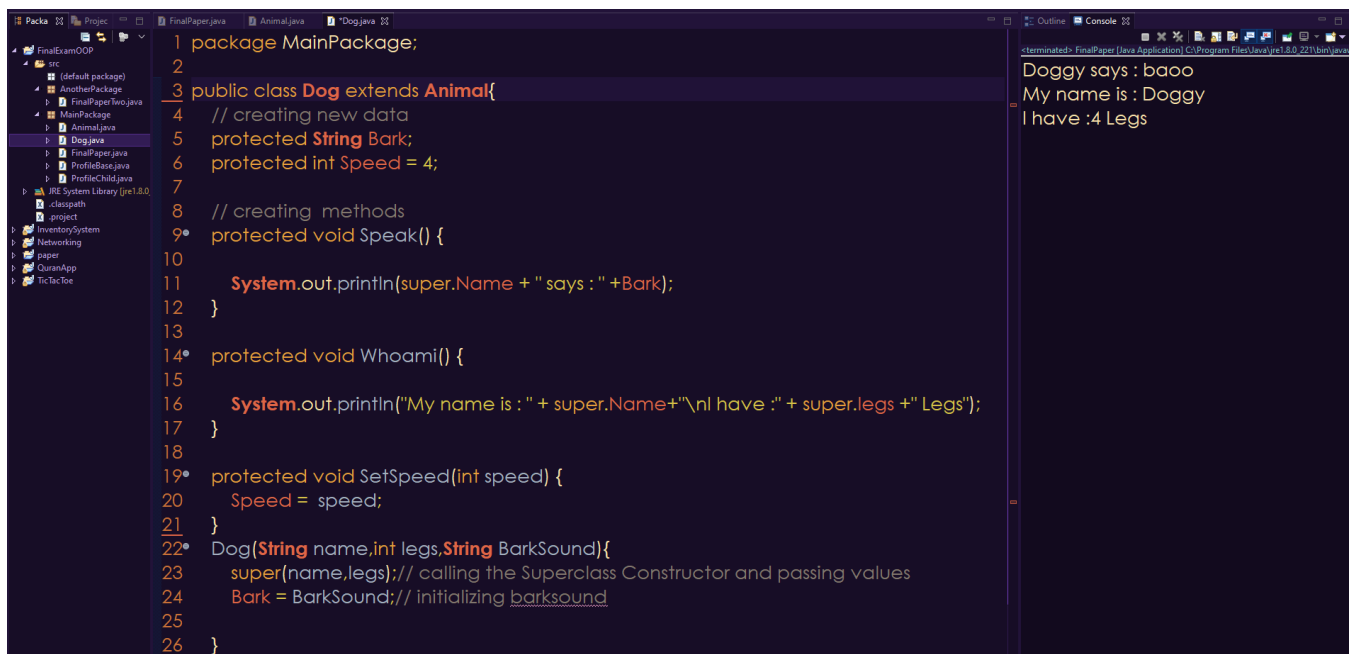
b. Write a program using Inheritance class on Animal in java.



```
1 package MainPackage;
2
3 public class Animal {
4 //lets map Basic Animal features
5
6 protected int legs;
7 protected String Name;
8
9
10 Animal(String name,int legs){
11 //initializing fields in constructor
12 this.legs = legs;
13 Name = name;
14
15 }
16
17
18 }
19
```

Doggy says : baoo
My name is : Doggy
I have :4 Legs

1- I first created Animal Base class and inside this class I defined some variables like common attribute of Animal and inside the constructor I initialized those variable fields

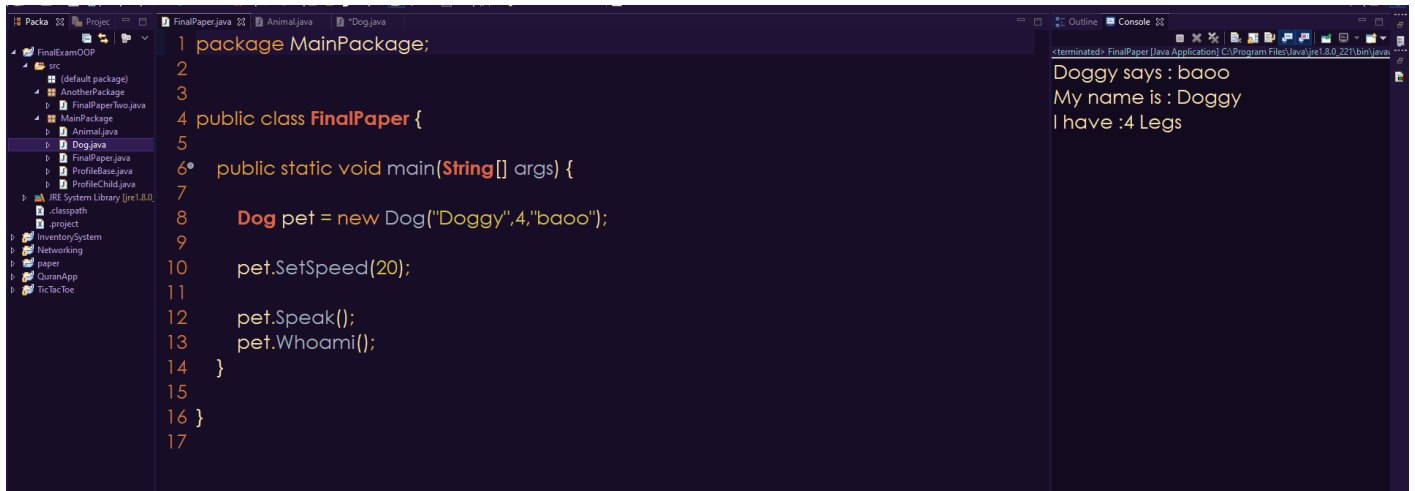


```
1 package MainPackage;
2
3 public class Dog extends Animal{
4 // creating new data
5 protected String Bark;
6 protected int Speed = 4;
7
8 // creating methods
9 protected void Speak() {
10
11 System.out.println(super.Name + " says : " + Bark);
12 }
13
14 protected void Whoami() {
15
16 System.out.println("My name is : " + super.Name+"\nI have : " + super.legs + " Legs");
17 }
18
19 protected void SetSpeed(int speed) {
20 Speed = speed;
21 }
22 Dog(String name,int legs,String BarkSound){
23 super(name,legs);// calling the Superclass Constructor and passing values
24 Bark = BarkSound;// initializing barksound
25
26 }
```

Doggy says : baoo
My name is : Doggy
I have :4 Legs

2- Then I created the child class named dog which is has inherited from Animal class and inside this class I created few more variable which was suited for dog and also few methods to later call from the object then

inside the constructor I used the superclass constructor which we call as super() and inside this we passed appropriate arguments to the constructor Notice how we everytime want to access some data from superclass we write super followed by period . to access the field or function if any



```
1 package MainPackage;
2
3
4 public class FinalPaper {
5
6     public static void main(String[] args) {
7
8         Dog pet = new Dog("Doggy",4,"baoo");
9
10        pet.SetSpeed(20);
11
12        pet.Speak();
13        pet.Whoami();
14    }
15
16 }
17
```

Console Output:
Doggy says : baoo
My name is : Doggy
I have :4 Legs

- 3- Then I instantiated / declared the Dog class object and pass appropriate constructor values
- 4- I set the speed to 20 which I forgot to show in the image
- 5- I called speak function
- 6- Followed by whoami function

Q4. a. What is polymorphism and why it is used, discuss in detail ?

Polymorphism :

Polymorphism consist of two words poly and morphism, poly means many , morphism means behavior , so many behavior we call this function overloading , is an example of polymorphism

Again once I create the program it will get clear

Pseudocode

We can have two classes one base class and one superclass and inside it we have a method like for example sayhello() now in both of them we put different body :

Superclass which is named Human

```
System.out.println("hi its me");
```

Child Class which is named Boy

It has this as the body of sayhello()

```
System.out.println("Yo how ya doin ");
```

Now if we make an object in main class like this

```
Human human = new Human();
```

Now if we call

```
human.sayhello();
```

it will output : hi its me

whereas if we make an object like this

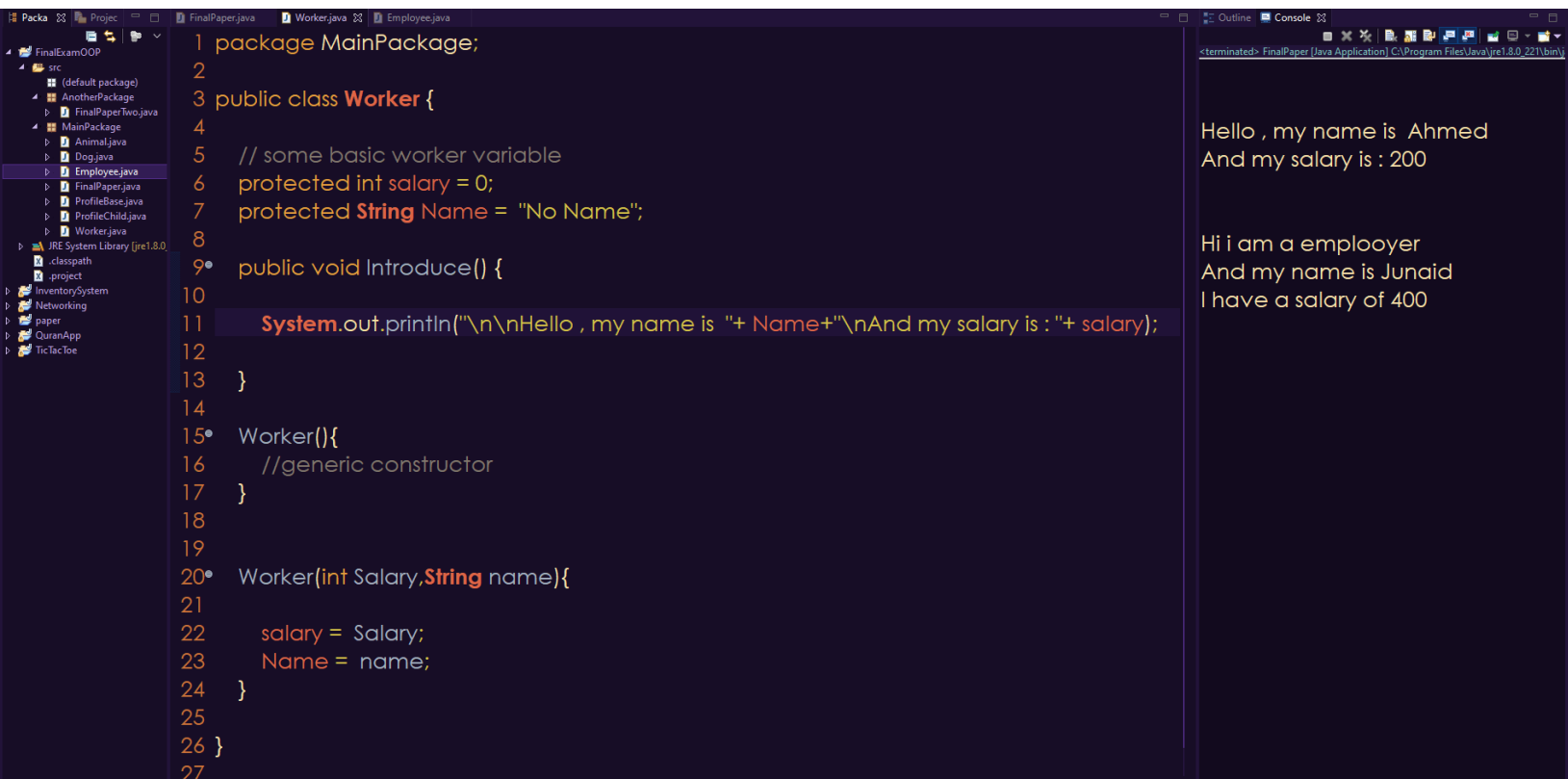
```
Human boy = new Boy();
```

And then we call that Method sayhello()

It will output: Yo how ya doing

So even though the object is Human Class Based on declaration it has override Function of Boy class the child class

b. Write a program using polymorphism in a class on Employee in java.



```
1 package MainPackage;
2
3 public class Worker {
4
5     // some basic worker variable
6     protected int salary = 0;
7     protected String Name = "No Name";
8
9     public void Introduce() {
10
11         System.out.println("\n\nHello , my name is "+ Name+"\n\nAnd my salary is : "+ salary);
12
13     }
14
15     Worker(){
16         //generic constructor
17     }
18
19
20     Worker(int Salary,String name){
21
22         salary = Salary;
23         Name = name;
24     }
25
26 }
27
```

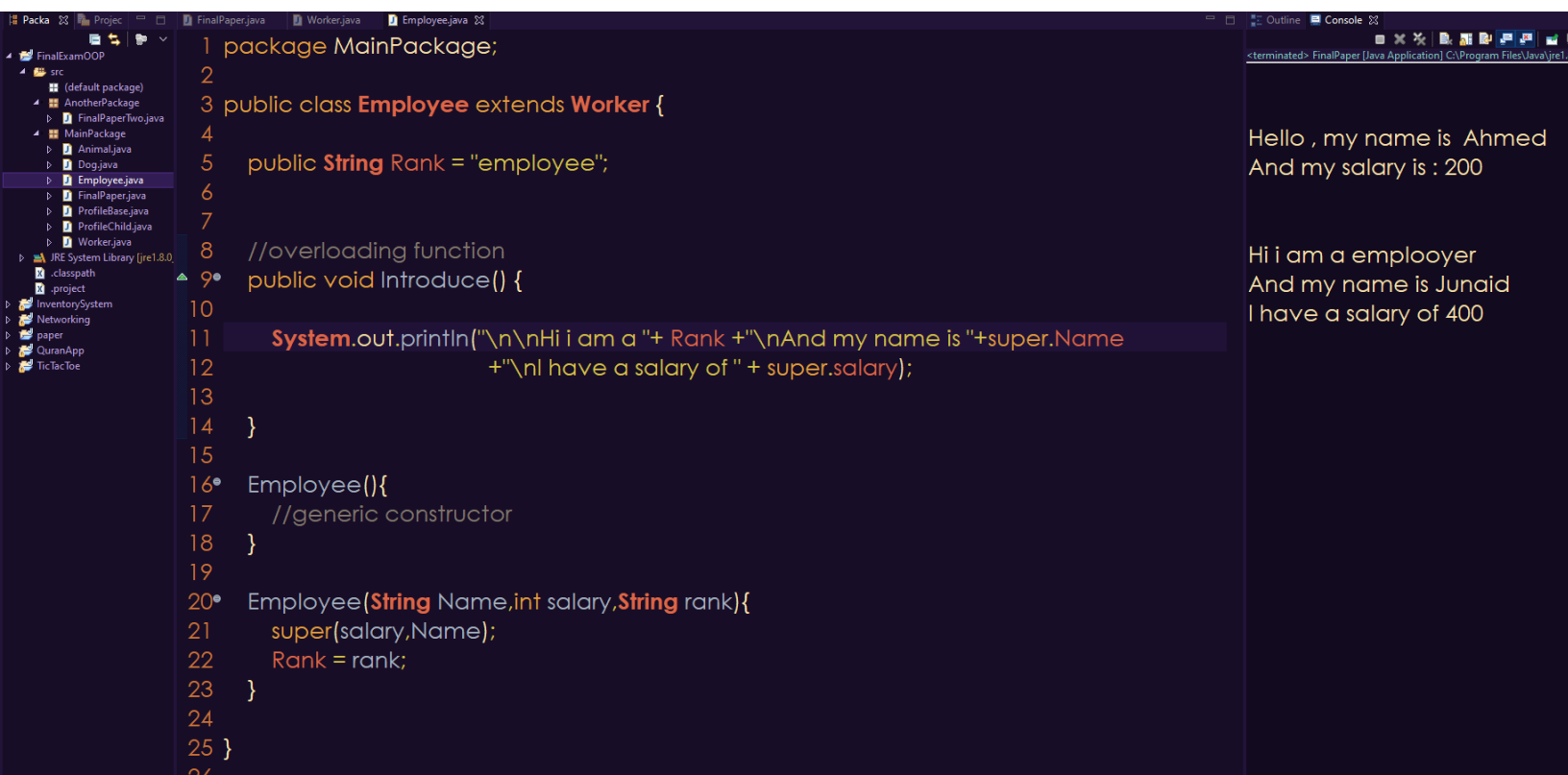
Hello , my name is Ahmed
And my salary is : 200

Hi i am an employeer
And my name is Junaid
I have a salary of 400

1- I created a worker superclass and declared few appropriate variables I made a method called introduce which will get overload and so is called polymorphism

we have a system out the statement inside it and so this will be changed according to over object

2- I created a generic and a constructor with params to initialize



```
1 package MainPackage;
2
3 public class Employee extends Worker {
4
5     public String Rank = "employee";
6
7
8     //overloading function
9     public void Introduce() {
10
11         System.out.println("\n\nHi i am a "+ Rank +"\n\nAnd my name is "+super.Name
12             +"\n\nI have a salary of " + super.salary);
13
14     }
15
16     Employee(){
17         //generic constructor
18     }
19
20     Employee(String Name,int salary,String rank){
21         super(salary,Name);
22         Rank = rank;
23     }
24
25 }
26
```

Console Output:

```
<terminated> FinalPaper [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\java.exe
Hello , my name is Ahmed
And my salary is : 200

Hi i am a employoer
And my name is Junaid
I have a salary of 400
```

3- Now I created An Child class called Employee which extends from Worker class inside this class I have a new field called Rank which is an additional thing I added

4-Then I made the same name Method Introduce which is, in this case, overloading for this particular class instantiated object inside this method I changed many things like the message with the rank

5-Lastly I defined one generic and one with parameter constructor to pass the value to the super constructor

```
1 package MainPackage;
2
3
4 public class FinalPaper {
5
6     public static void main(String[] args) {
7
8         Worker worker_1 = new Worker(200,"Ahmed");
9
10        worker_1.Introduce();// calling superclass method
11
12        Worker worker_2 = new Employee("Junaid",400,"employer");
13
14        worker_2.Introduce();// calling overloaded method
15    }
16 }
17
18
19 }
20
```

Hello , my name is Ahmed
And my salary is : 200

Hi i am a emplooyer
And my name is Junaid
I have a salary of 400

Then I created two objects for demonstration purpose the first one as a normal Superclass

And The other one with the Object type of Worker but casting it as Employee constructor / calling employee constructor

And then calling both of their introduce methods

Q5. a. Why abstraction is used in OOP, discuss in detail?

Abstraction :

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces

An abstract class is same as an interface in java but the abstract class has the method and fields defined/declared doesn't necessarily need to be implemented to the inherited class and unlike interface where the method can't have a body the abstract class can have body created in the abstract class itself. it is just a restricted class that cannot be used to create objects on, to access it, it must be inherited from another class as a template

Use of this abstract class is to extend or add more functions to a class

A normal class can't have multiple superclasses but it can have multiple interface or abstract classes

Sometimes we need multiple interfaces and abstract classes to a single class which can happen and it is important to keep this in mind

There are three ways to abstract data

- 1- Abstract class
- 2- Abstract methods
- 3- Interface

We might get confused between data abstraction and encapsulating data, I used to be confused but abstraction is defined as :

Abstraction means implementation hiding whereas encapsulation means information hiding

Encapsulation groups together data and methods that act upon the data, data Abstraction deals with exposing the interface to the user and hiding the details of implementation

Now what is the benefit of Abstraction in OOP:

- 1- It reduces the complexity of viewing things.
- 2- Avoids code duplication and increases reusability
- 3- It helps to increase the security of an application or program as only important details are provided to the user.

I will go ahead and give an example of both abstract class and template to give you an idea of what is it.

b. Write a program on abstraction in java.

```
abstract class Shape
{
    String color;

    // these are abstract methods
    abstract double area();
    public abstract String toString();

    public Shape(String color) {
        this.color = color;
    }

    // this is a concrete method
    public String getColor() {
        return color;
    }
}
```

- 1- Firstly I created abstract shape Class and in side I have abstract methods and a constructor to take the color and assign it to the variable color
- 2- I made another String returned method getColor which just returns color variable this is what known as concrete method

```
class Circle extends Shape
{
    double radius;

    public Circle(String color,double radius) {

        super(color);
        this.radius = radius;
    }

    @Override
    double area() {
        return Math.PI * Math.pow(radius, 2);
    }

    @Override
    public String toString() {
        return "Circle color is " + super.color +
            "and area is : " + area();
    }
}
```

- 3- Then I created a circle class which extends form the abstract class inside this class I have different set of details but the superclass which is shape abstract class implements 2

methods which we made abstract these methods need to be implemented into the base/child class

- 4- But we don't need to implement getColor which was a concrete method we defined or OVERRIDE area method and ToString Method because we want to implement this function according to our need

```
class Rectangle extends Shape{  
  
    double length;  
    double width;  
  
    public Rectangle(String color,double length,double width) {  
        super(color);  
        this.length = length;  
        this.width = width;  
    }  
  
    @Override  
    double area() {  
        return length*width;  
    }  
  
    @Override  
    public String toString() {  
        return "Rectangle color is " + super.color +  
            "and area is : " + area();  
    }  
}
```

- 5- I made another rectangle class which also extends from the shape abstract class
- 6- Inside this class I declared variable according to that rectangle shape like as we declared radius in the circle class inside this class I declared length and width because we know to find area of the rectangle we need length and width instead of radius

```
public class FinalPaper  
{  
    public static void main(String[] args)  
    {  
        Shape ShapeC = new Circle("Blue ", 5);  
        Shape ShapeR= new Rectangle("Orange ", 10, 15);  
  
        System.out.println(ShapeC.toString());  
        System.out.println(ShapeR.toString());  
    }  
}
```

- 7- Now inside of our FinalPaper class I initialized two shape abstract class object but upon calling the constructor I used different constructors for each appropriately
- 8- Then I called their toString() method which resulted in giving me the output with proper details

OUTPUT

```
Circle color is Blue and area is : 78.5  
Rectangle color is Orange and area is : 150.0
```

The interface is same as abstract class but the only difference is that as we seen in abstract class and method the method some of the function didn't required implementation but in interface class we must implement it there is nothing like a concrete method as in abstract class

We use "extends" for abstract class implementation
Whereas in interface we use "implements"