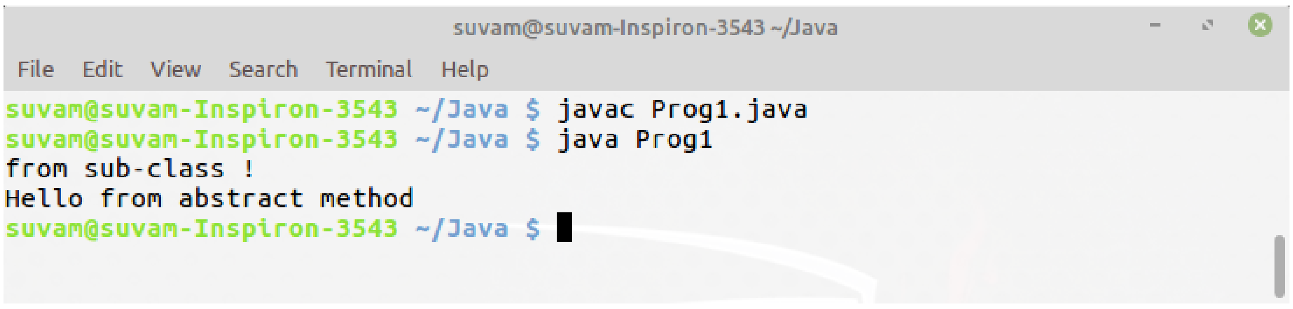
24. Abstract class.

Code:

```
abstract class A{
     abstract void callme();
     void callmetoo(){
          System.out.println("Hello from abstract method");
     }
}
class B extends A{
     void callme(){
          System.out.println("from sub-class !");
     }
}
public class Prog1 {
     public static void main(String[] args){
          B b = new B();

          b.callme();
          b.callmetoo();
     }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

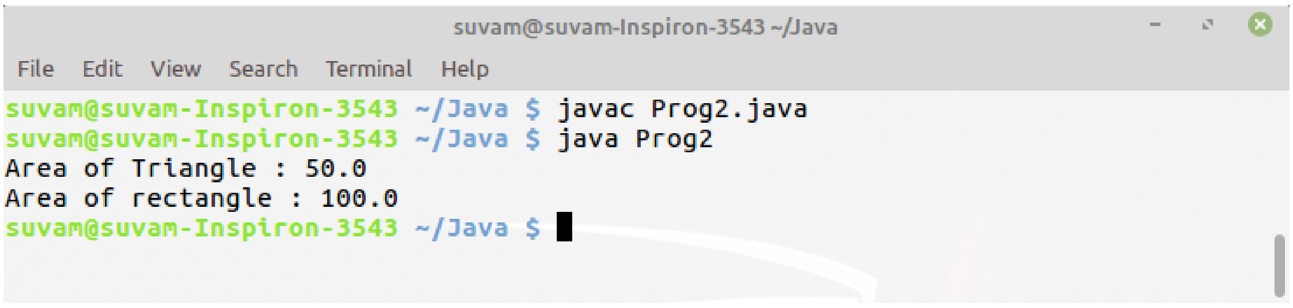
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog1.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog1
from sub-class !
Hello from abstract method
suvam@suvam-Inspiron-3543 ~/Java $
```

25. Use of Abstract class.

Code:

```
abstract class Figure{
     double dim1,dim2;
     Figure(double a, double b){
          dim1 = a;
          dim2 = b;
     }
     abstract double area();
}
class Rectangle extends Figure{
     Rectangle(double a, double b){
          super(a,b);
     }

     double area(){
          return (dim1 * dim2);
     }
}
class Triangle extends Figure{
     Triangle(double a, double b){
          super(a,b);
     }
     double area(){
          return (dim1 * dim2)/2;
     }
}
public class Prog2 {
     public static void main(String[] args) {
          Triangle t = new Triangle(10,10);
          Rectangle r = new Rectangle(10,10);
          Figure figure;
          figure = t;
          System.out.println("Area of Triangle : " +
figure.area());
          figure = r;
          System.out.println("Area of rectangle : " +
figure.area());

     }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

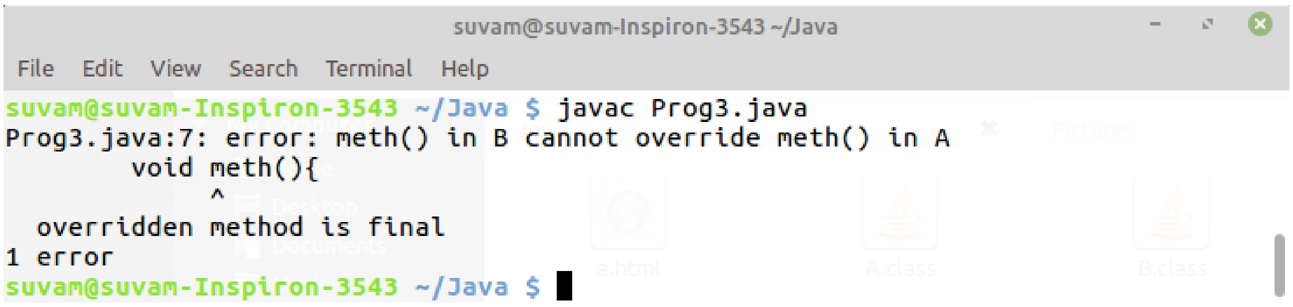
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog2.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog2
Area of Triangle : 50.0
Area of rectangle : 100.0
suvam@suvam-Inspiron-3543 ~/Java $
```

26. Using final to Prevent Overrriding.

Code:

```
class A{
     final void meth(){
          System.out.println("Final method!");
     }
}
class B extends A{
     void meth(){
          System.out.println(" ???? ");
     }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

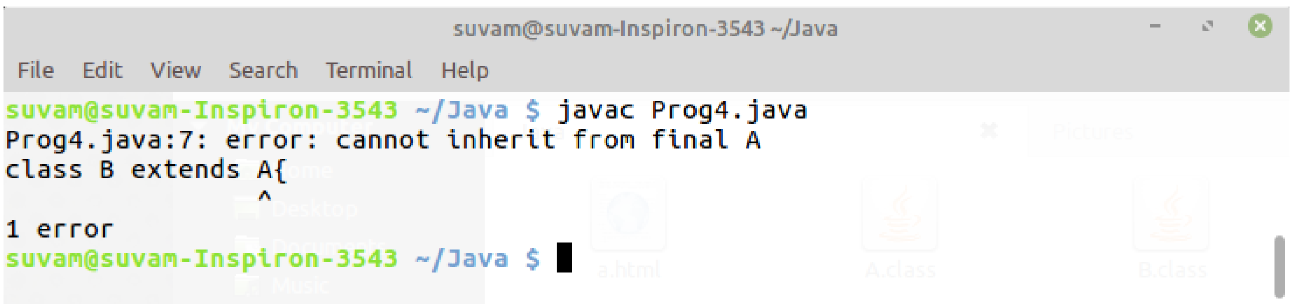
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog3.java
Prog3.java:7: error: meth() in B cannot override meth() in A
        void meth(){
             ^
  overridden method is final
1 error
suvam@suvam-Inspiron-3543 ~/Java $
```

27. Using final to Prevent Inheritance.

Code:

```
final class A{
     void print(){
          System.out.println("hello world !");
     }
}

class B extends A{

}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

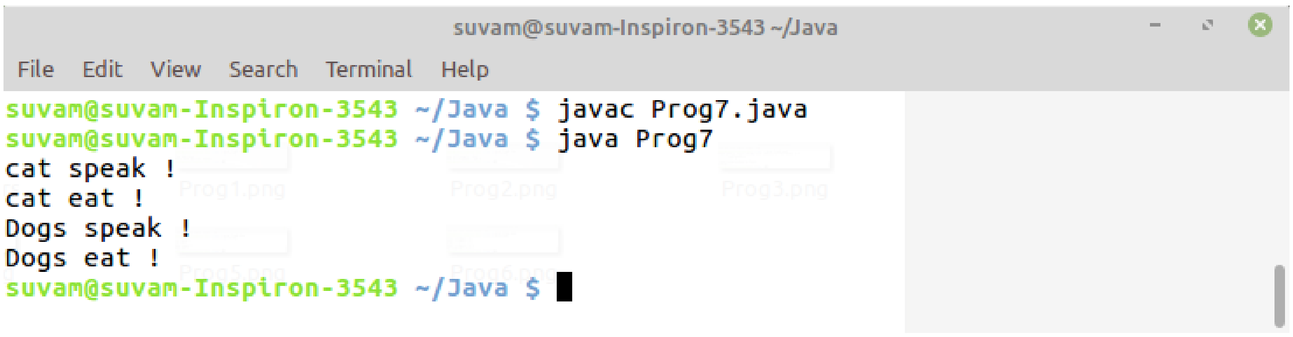
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog4.java
Prog4.java:7: error: cannot inherit from final A
class B extends A{
                 ^
1 error
suvam@suvam-Inspiron-3543 ~/Java $ 
```

28. Abstract class variable Can reference a Subclass Object.

Code:

```
abstract class Animal{
	abstract void eat();
	abstract void speak();
}
class Dog extends Animal{
	void eat(){
		System.out.println("Dogs eat !");
	}
	void speak(){
		System.out.println("Dogs speak !");
	}
}
class Cat extends Animal{
	void eat(){
		System.out.println("cat eat !");
	}
	void speak(){
		System.out.println("cat speak !");
	}
}
public class Prog7 {
	public static void main(String[] args){
		Cat c = new Cat();
		Dog d = new Dog();

		c.speak();
		c.eat();
		d.speak();
		d.eat();
	}
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

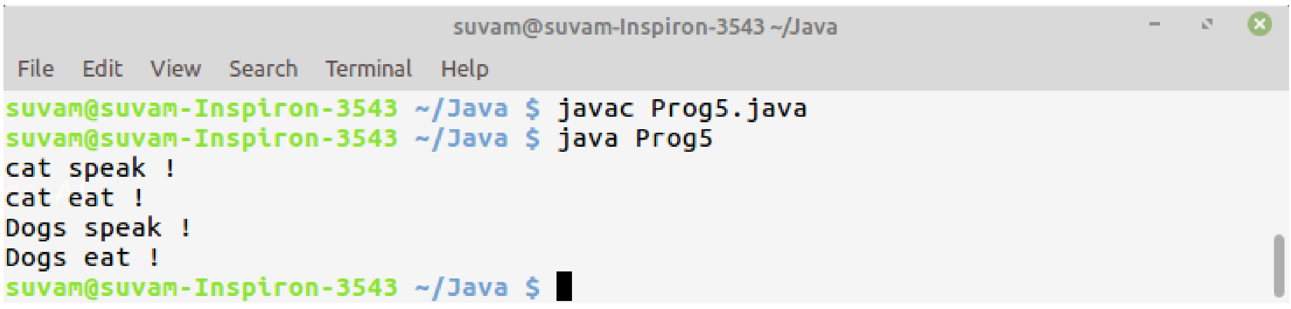
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog7.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog7
cat speak !
cat eat !
Dogs speak !
Dogs eat !
suvam@suvam-Inspiron-3543 ~/Java $
```

29. Call subclass Methods using Method.

Code:

```
abstract class Animal{
     abstract void eat();
     abstract void speak();
}
class Dog extends Animal{
     void eat(){
          System.out.println("Dogs eat !");
     }
     void speak(){
          System.out.println("Dogs speak !");
     }
}
class Cat extends Animal{
     void eat(){
          System.out.println("cat eat !");
     }
     void speak(){
          System.out.println("cat speak !");
     }
}

public class Prog5 {
     void make(Animal a){
          a.speak();
          a.eat();
     }
     public static void main(String[] args){
          Animal c = new Cat();
          Animal d = new Dog();


          Prog5 m = new Prog5();

          m.make(c);
          m.make(d);
     }

}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

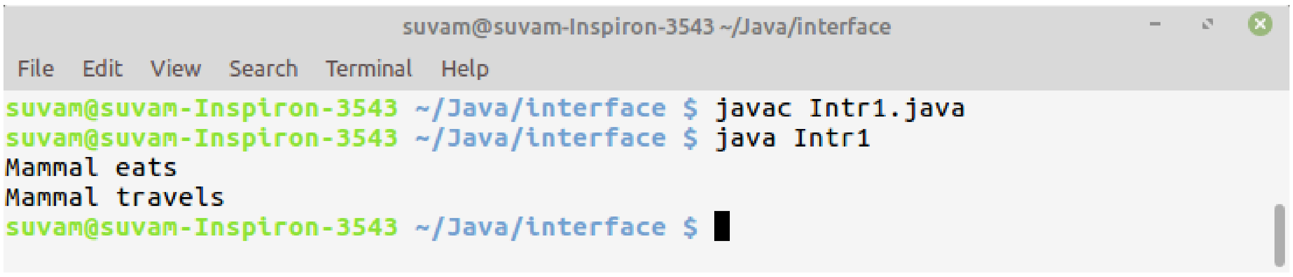
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog5.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog5
cat speak !
cat eat !
Dogs speak !
Dogs eat !
suvam@suvam-Inspiron-3543 ~/Java $ 
```

30. Interface Bacis.

Code:

```
interface Animal {
   public void eat();
   public void travel();
}
class Mammal implements Animal {
   public void eat() {
      System.out.println("Mammal eats");
   }
   public void travel() {
      System.out.println("Mammal travels");
   }
   public int noOfLegs() {
      return 0;
   }
}
class Intr1{
     public static void main(String args[]) {
      Mammal m = new Mammal();
      m.eat();
      m.travel();
   }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java/interface
File Edit View Search Terminal Help

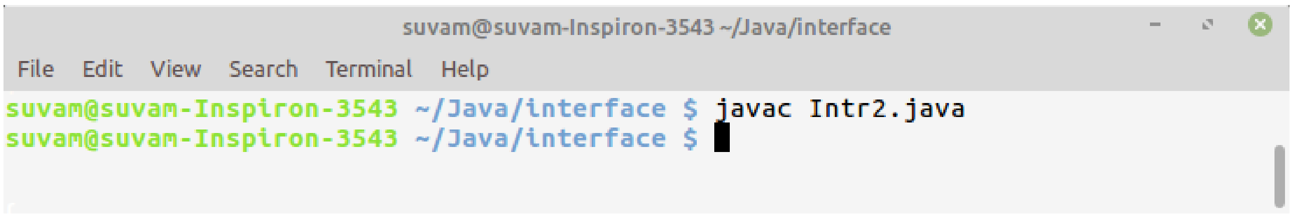
```
suvam@suvam-Inspiron-3543 ~/Java/interface $ javac Intr1.java
suvam@suvam-Inspiron-3543 ~/Java/interface $ java Intr1
Mammal eats
Mammal travels
suvam@suvam-Inspiron-3543 ~/Java/interface $ 
```

31. Extending Interface.

Code:

```
interface Sports {
   public void setHomeTeam(String name);
   public void setVisitingTeam(String name);
}

interface Football extends Sports {
   public void homeTeamScored(int points);
   public void visitingTeamScored(int points);
   public void endOfQuarter(int quarter);
}

interface Hockey extends Sports {
   public void homeGoalScored();
   public void visitingGoalScored();
   public void endOfPeriod(int period);
   public void overtimePeriod(int ot);
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java/interface
File Edit View Search Terminal Help

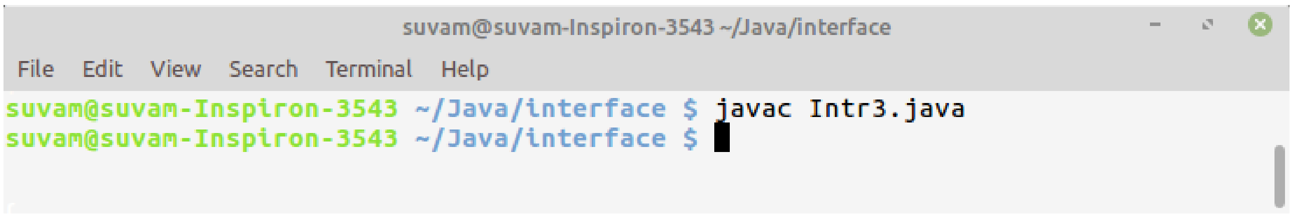
```
suvam@suvam-Inspiron-3543 ~/Java/interface $ javac Intr2.java
suvam@suvam-Inspiron-3543 ~/Java/interface $ 
```

32. Extending Multiple interface.

Code:

```
interface Sports {
   public void setHomeTeam(String name);
   public void setVisitingTeam(String name);
}

interface Football extends Sports {
   public void homeTeamScored(int points);
   public void visitingTeamScored(int points);
   public void endOfQuarter(int quarter);
}

interface Hockey extends Sports {
   public void homeGoalScored();
   public void visitingGoalScored();
   public void endOfPeriod(int period);
   public void overtimePeriod(int ot);
}

interface Today extends Sports, Hockey, Football{
     public void displayGame(String s);
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java/interface
File Edit View Search Terminal Help

```
suvam@suvam-Inspiron-3543 ~/Java/interface $ javac Intr3.java
suvam@suvam-Inspiron-3543 ~/Java/interface $ 
```

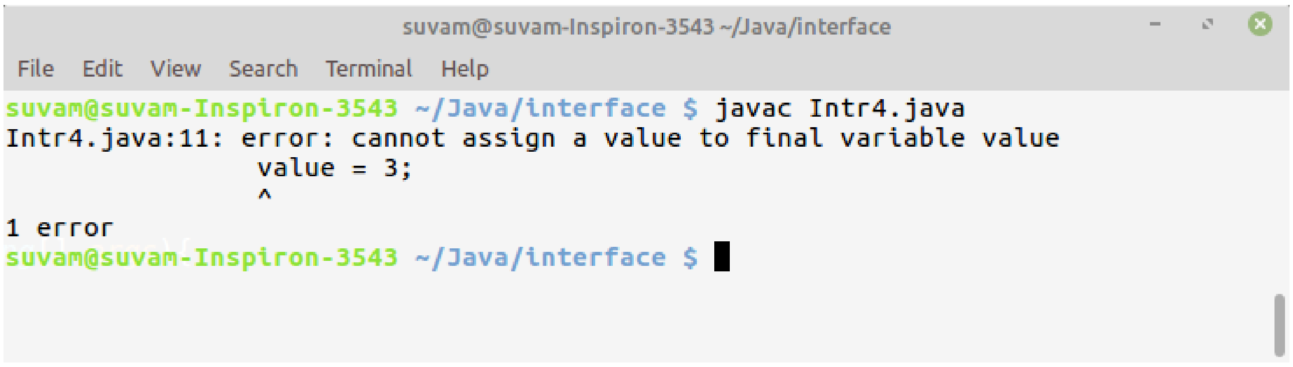
33. Extending Multiple interface.

Code:

```
interface MyInterface{
     int value = 30;
     public void display();
}
```

```
class Me implements MyInterface{
	public void display(){
		System.out.println("Display : " + value);
	}
	public void tryToChange(){
		value = 3;
	}
}

class Intr4{
	public static void main(String[] args){
		Me ob = new Me();
		ob.display();
	}
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java/interface
File Edit View Search Terminal Help

```
suvam@suvam-Inspiron-3543 ~/Java/interface $ javac Intr4.java
Intr4.java:11: error: cannot assign a value to final variable value
                value = 3;
                ^
1 error
suvam@suvam-Inspiron-3543 ~/Java/interface $
```

34. Packages in Java.

Code:

File path: */java/package1/package2*

```
package package1.package2;
public class ClassC{
	public void displayC(){
		System.out.println("ClassC from package2");
	}
}
```

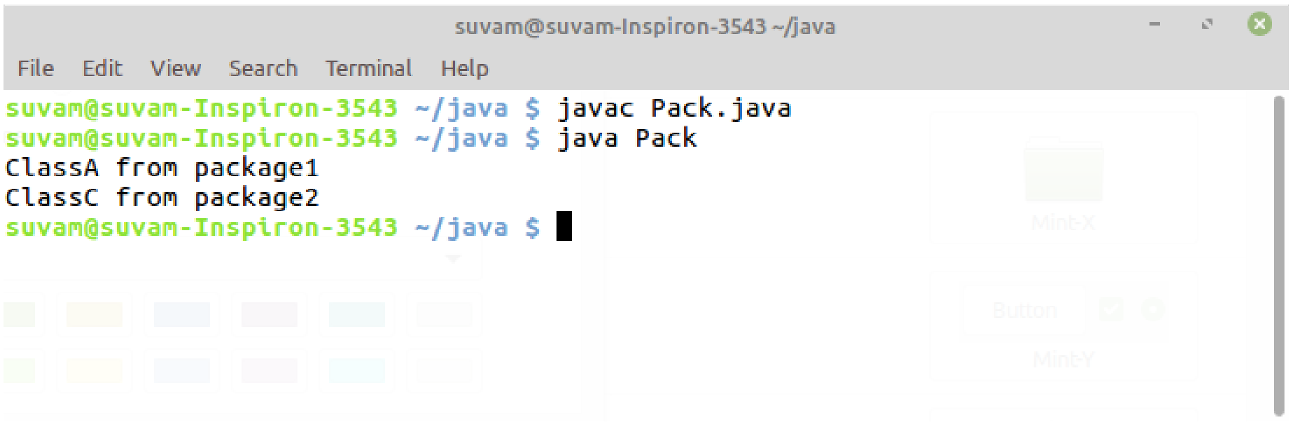
File path: */java/package1*

```
package package1;
public class ClassA{
     public void displayA(){
          System.out.println("ClassA from package1 ");
     }
}
```

File path: */java*

```
import package1.*;
import package1.package2.*;

class Pack{
     public static void main(String[] args){
          ClassA obj= new ClassA();
          obj.displayA();
          ClassC ob = new ClassC();
          ob.displayC();
     }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/java
File Edit View Search Terminal Help

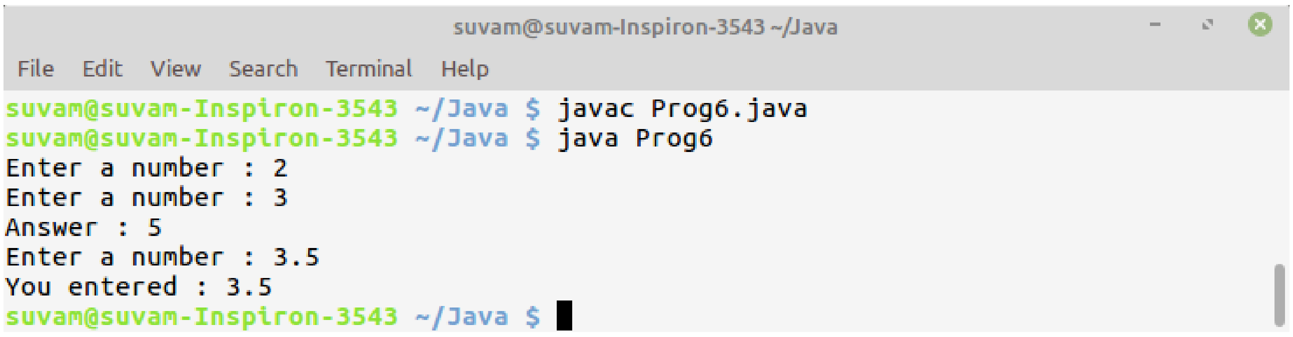
```
suvam@suvam-Inspiron-3543 ~/java $ javac Pack.java
suvam@suvam-Inspiron-3543 ~/java $ java Pack
ClassA from package1
ClassC from package2
suvam@suvam-Inspiron-3543 ~/java $
```

35. Input values using Java Scanner Class.

Code:

```
import java.util.Scanner;

public class Prog6 {
     public static void main(String[] args){
          System.out.print("Enter a number : ");
          Scanner scan = new Scanner(System.in);
          int num1 = scan.nextInt();

          System.out.print("Enter a number : ");
          int num2 = scan.nextInt();

          int ans  = num1 + num2;
          System.out.println("Answer : " + ans);

          System.out.print("Enter a number : ");
          double num3 = scan.nextDouble();
          System.out.print("You entered : " + num3);
     }
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

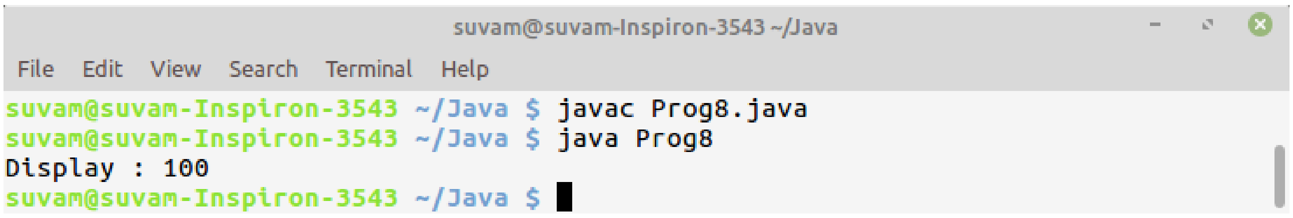
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog6.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog6
Enter a number : 2
Enter a number : 3
Answer : 5
Enter a number : 3.5
You entered : 3.5
suvam@suvam-Inspiron-3543 ~/Java $
```

36. Inner Class.

Code:

```
class Outer{
	int outer_x = 100;
	void test(){
		Inner inner = new Inner();
		inner.display();
	}
	class Inner{
		void display(){
			System.out.println("Display : " + outer_x);
		}
	}
}

public class Prog8 {
	public static void main(String[] args){
		Outer a = new Outer();
		a.test();
	}
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

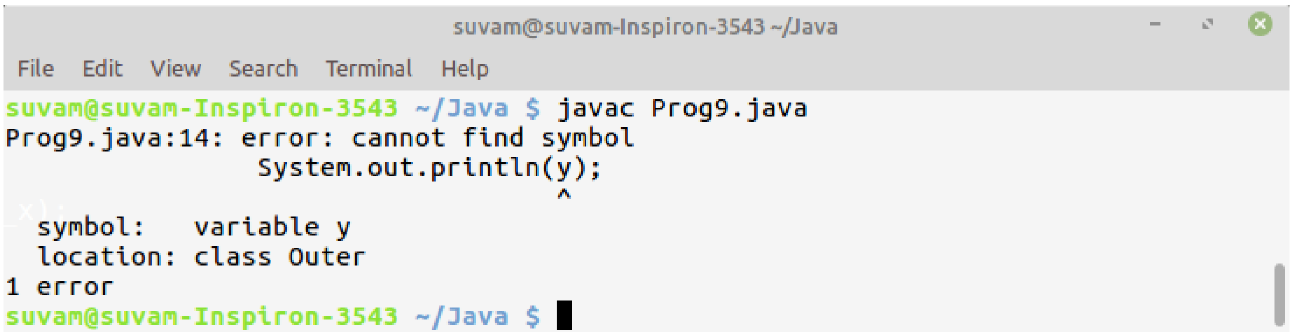
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog8.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog8
Display : 100
suvam@suvam-Inspiron-3543 ~/Java $
```

37. Inner Class can Access Outer Class Variables.

Code:

```
class Outer{
	int outer_x = 100;
	void test(){
		Inner inner = new Inner();
		inner.display();
	}
	class Inner{
		int y  = 10;
		void display(){
			System.out.println("Display : " + outer_x);
		}
	}
	void show(){
		System.out.println(y);
	}
}
public class Prog9 {
	public static void main(String[] args){
		Outer a = new Outer();
		a.test();
	}
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

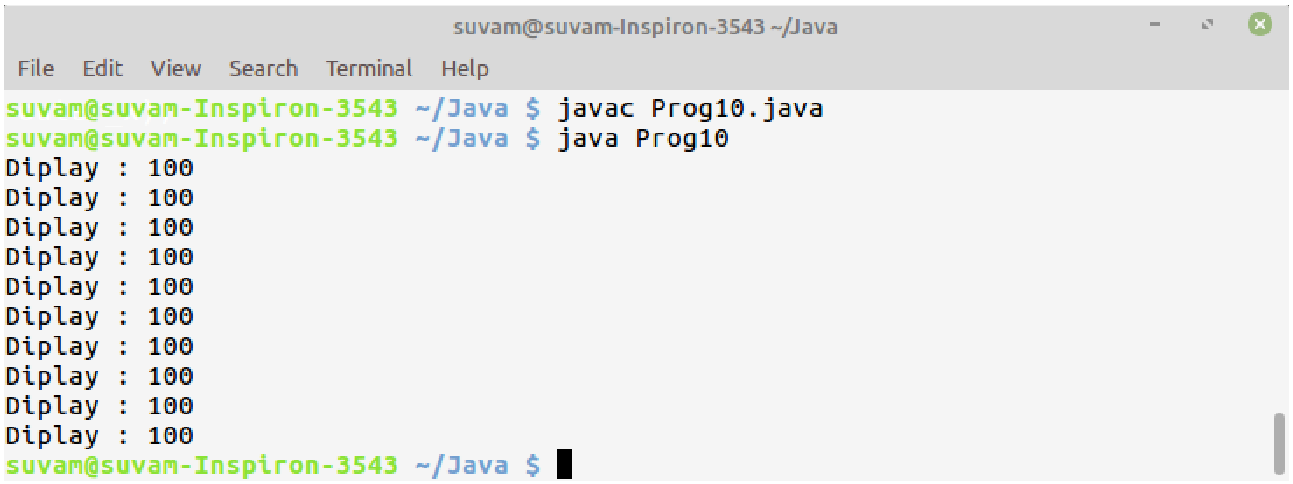
```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog9.java
Prog9.java:14: error: cannot find symbol
                System.out.println(y);
                                   ^
  symbol:   variable y
  location: class Outer
1 error
suvam@suvam-Inspiron-3543 ~/Java $ 
```

38. Another Example os Inner Class where the class is defined inside a for loop.

Code:

```
class Outer{
	int outer_x = 100;
	void test(){
		for (int i = 0; i<10;i++){
			class Inner{
				void display(){
					System.out.println("Diplay : "+ outer_x);
				}
			}
			Inner inner = new Inner();
			inner.display();
		}
	}
}

class Prog10{
	public static void main(String[] args){
		Outer outer = new Outer();
		outer.test();
	}
}
```

Output:


suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog10.java
suvam@suvam-Inspiron-3543 ~/Java $ java Prog10
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
Diplay : 100
suvam@suvam-Inspiron-3543 ~/Java $ 
```

39. Anonymous Inner Class.

Code:

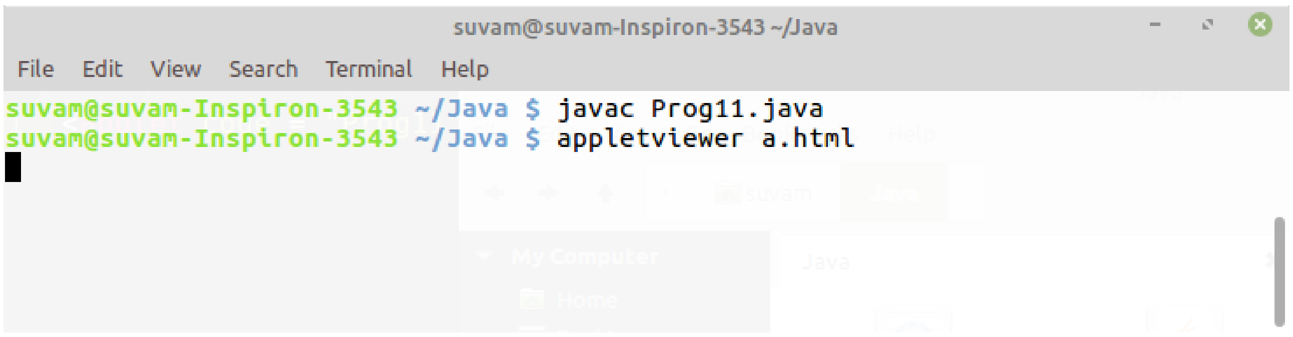
Java File:

```
import java.applet.*;
import java.awt.event.*;

public class Prog11 extends Applet{
     public void init(){
          addMouseListener(new MouseAdapter(){
               public void mousePressed(MouseEvent me){
                    showStatus("Mouse pressed !");
               }
          });
     }
}
```

HTML File:

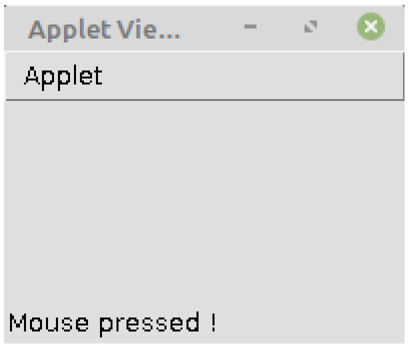
```
<applet code = "Prog11" width=200 height=100></applet>
```

Output:



suvam@suvam-Inspiron-3543 ~/Java
File Edit View Search Terminal Help

```
suvam@suvam-Inspiron-3543 ~/Java $ javac Prog11.java
suvam@suvam-Inspiron-3543 ~/Java $ appletviewer a.html
```



```
Mouse pressed !
```