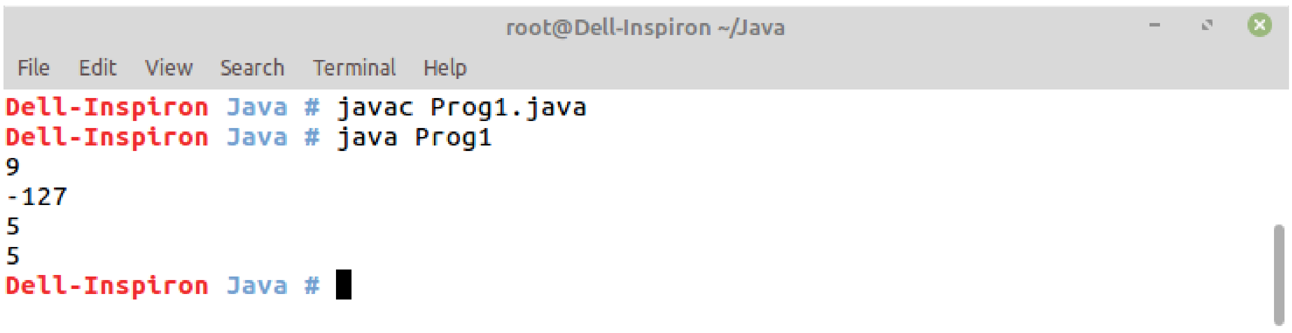
8. Type conversion & casting incompatible types.

Code:

```
class Prog1{
public static void main(String[] args){
     byte x=9;
     System.out.println(x);

     x=(byte)129;
     System.out.println(x);

     int y=5;
     x=(byte)y;
     System.out.println(x);

     float z=8.9f;
     System.out.println(x);
     }
}
```

Output:


root@Dell-Inspiron ~/Java
File Edit View Search Terminal Help

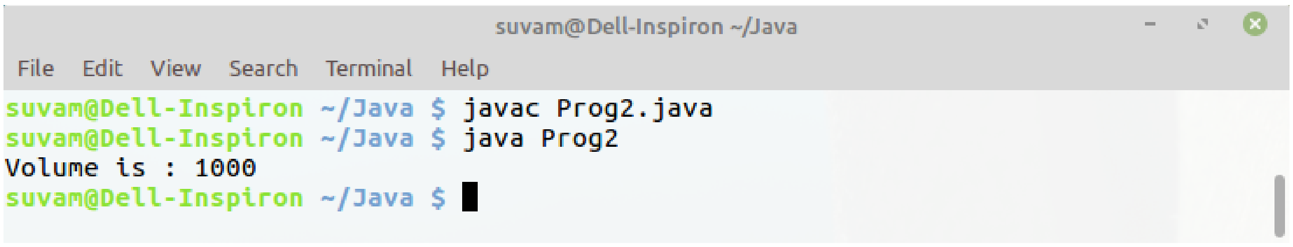
```
Dell-Inspiron Java # javac Prog1.java
Dell-Inspiron Java # java Prog1
9
-127
5
5
Dell-Inspiron Java #
```

9. A simple class & Adding a method to the class.

Code:

```
class Box{
	int width;
	int height;
	int length;

	void volume(){
		System.out.print("Volume is : ");
		System.out.println(width*length*height);
	}
}

class Prog2{
	public static void main(String[] args){
		Box myBox = new Box();

		myBox.width = 10;
		myBox.height = 10;
		myBox.length = 10;

		myBox.volume();
	}
}
```

Output:


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

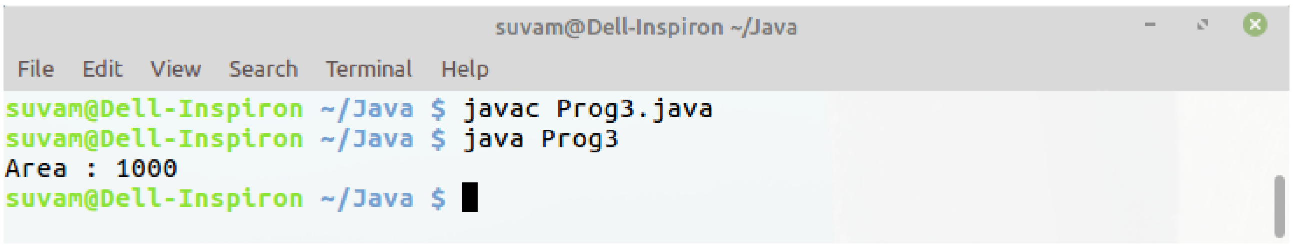
```
suvam@Dell-Inspiron ~/Java $ javac Prog2.java
suvam@Dell-Inspiron ~/Java $ java Prog2
Volume is : 1000
suvam@Dell-Inspiron ~/Java $ 
```

10. Adding a method to the class which returns values.

Code:

```
class Box{
     int width;
     int height;
     int length;

     int volume(){
          return (width*length*height);
     }
}

class Prog3{
     public static void main(String[] args){
          Box myBox = new Box();

          myBox.width = 10;
          myBox.height = 10;
          myBox.length = 10;

          System.out.println("Area : " + myBox.volume());
     }
}
```

Output:


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

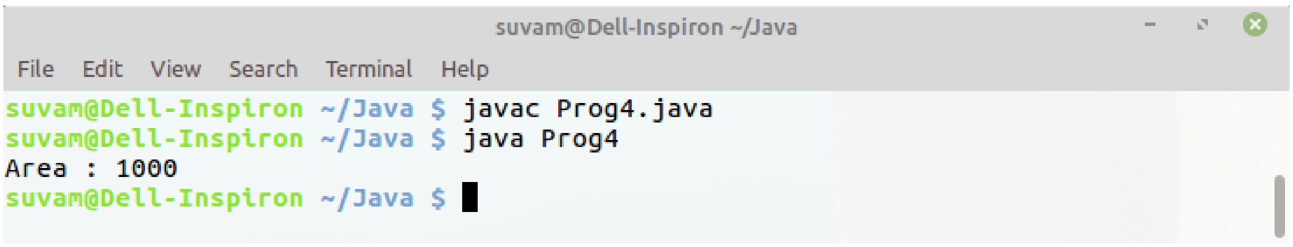
```
suvam@Dell-Inspiron ~/Java $ javac Prog3.java
suvam@Dell-Inspiron ~/Java $ java Prog3
Area : 1000
suvam@Dell-Inspiron ~/Java $
```

11. Adding a Parameterized method to the class which Returns values.

Code:

```
class Box{
     int volume(int width,int length,int height){
          return (width*length*height);
     }
}

class Prog4{
     public static void main(String[] args){

          Box myBox = new Box();

          System.out.println("Area : " + myBox.volume(10,10,10));
     }
}
```

Output:


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

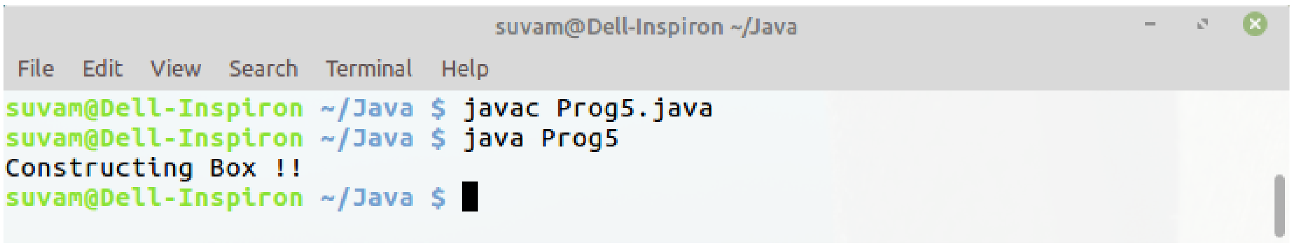
```
suvam@Dell-Inspiron ~/Java $ javac Prog4.java
suvam@Dell-Inspiron ~/Java $ java Prog4
Area : 1000
suvam@Dell-Inspiron ~/Java $ 
```

12. Use of Constructor.

Code:

```
class Box{
	Box(){
		System.out.println("Constructing Box !!");
	}
}

class Prog5{
	public static void main(String[] args){

		Box myBox = new Box();

	}
}
```

Output:


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

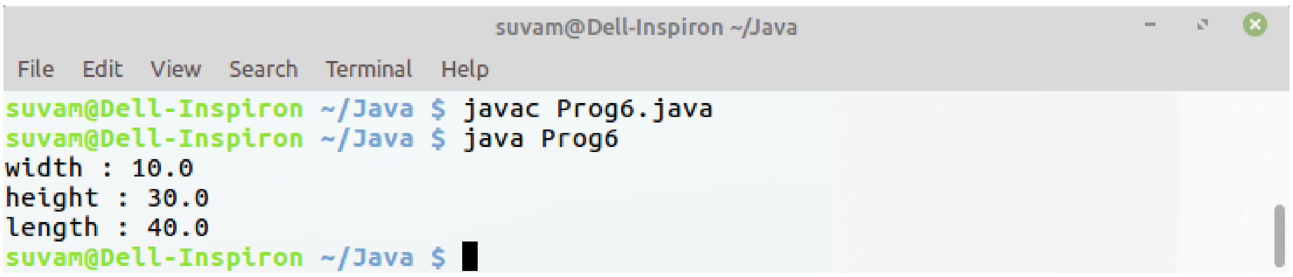
```
suvam@Dell-Inspiron ~/Java $ javac Prog5.java
suvam@Dell-Inspiron ~/Java $ java Prog5
Constructing Box !!
suvam@Dell-Inspiron ~/Java $ 
```

13. Parameterized Constructor.

Code:

```
class Box{
	double width;
	double height;
	double length;

	Box(double width, double height,  double length){
		this.width = width;
		this.height = height;
		this.length = length;
	}
}

class Prog6{
	public static void main(String[] args){
		Box myBox = new Box(10,30,40);

		System.out.println("width : "+myBox.width);
		System.out.println("height : "+myBox.height);
		System.out.println("length : "+myBox.length);
	}
}
```

Output:


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

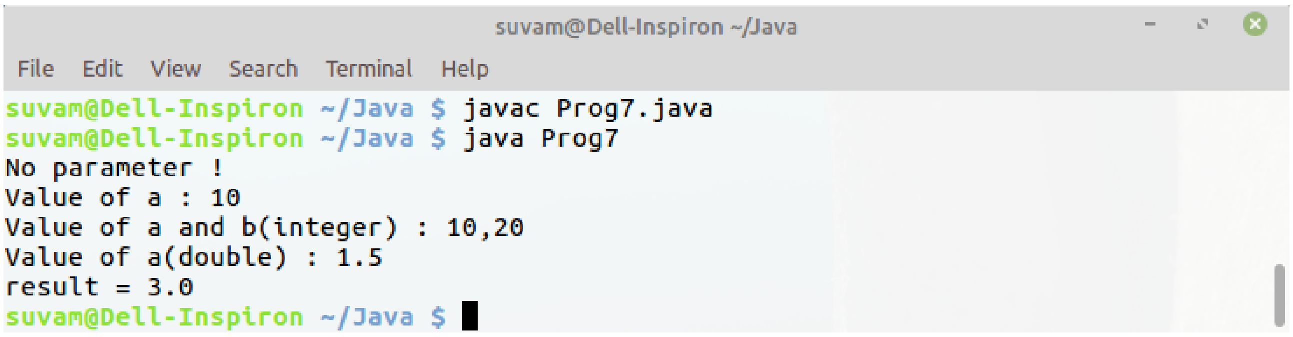
```
suvam@Dell-Inspiron ~/Java $ javac Prog6.java
suvam@Dell-Inspiron ~/Java $ java Prog6
width : 10.0
height : 30.0
length : 40.0
suvam@Dell-Inspiron ~/Java $ 
```

14. Methods overloading.

Code:

```
class Overloading{
	void test(){
		System.out.println("No parameter !");
	}
	void test(int a){
		System.out.println("Value of a : " + a);
	}
	void test(int a, int b){
		System.out.println("Value of a and b(integer) : " + a +
"," + b);
	}
	double test(double a){
		System.out.println("Value of a(double) : " + a);
		return (a+a);
	}
}

class Prog7{
	public static void main(String[] args){
		Overloading ob = new Overloading();
		double result;

		ob.test();
		ob.test(10);
		ob.test(10,20);
		result = ob.test(1.5);

		System.out.println("result = " + result);
	}
}
```

Output:


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

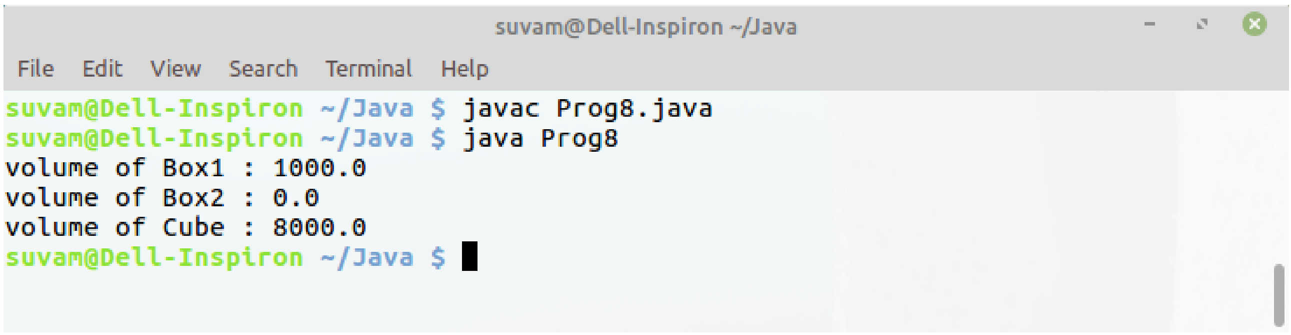
```
suvam@Dell-Inspiron ~/Java $ javac Prog7.java
suvam@Dell-Inspiron ~/Java $ java Prog7
No parameter !
Value of a : 10
Value of a and b(integer) : 10,20
Value of a(double) : 1.5
result = 3.0
suvam@Dell-Inspiron ~/Java $ 
```

15. Constructor overloading.

Code:

```
class Box{
	double width;
	double height;
	double length;
	Box(){
		width = 0;
		height = 0;
		length = 0;
	}
	Box(int len){
		width = length = height = len;
	}
	Box(double width, double height,  double length){
		this.width = width;
		this.height = height;
		this.length = length;
	}
	double volume(){
		return (width*length*height);
	}
}
class Prog8{
	public static void main(String[] args){
		Box myBox1 = new Box(10,10,10);
		Box myBox2 = new Box();
		Box myCube = new Box(20);
		System.out.println("volume of Box1 : "+myBox1.volume());
		System.out.println("volume of Box2 : "+myBox2.volume());
		System.out.println("volume of Cube : "+myCube.volume());
	}
}
```

Output:


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

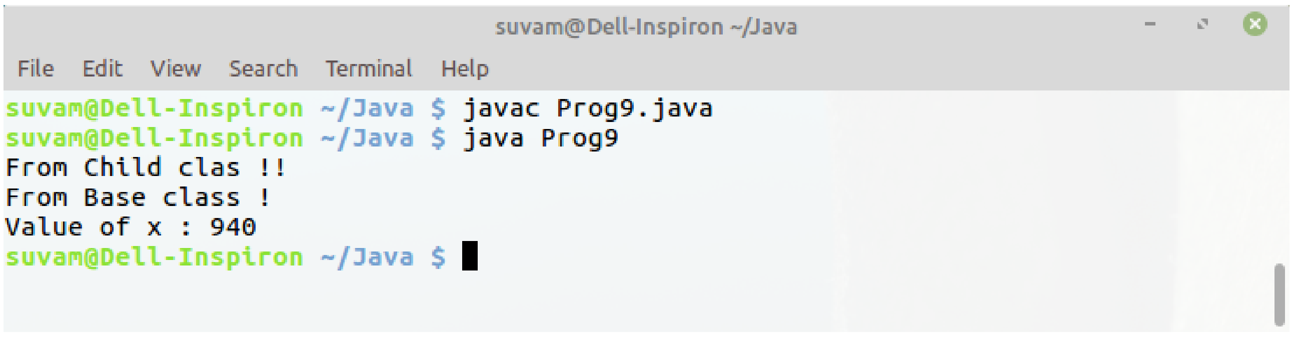
```
suvam@Dell-Inspiron ~/Java $ javac Prog8.java
suvam@Dell-Inspiron ~/Java $ java Prog8
volume of Box1 : 1000.0
volume of Box2 : 0.0
volume of Cube : 8000.0
suvam@Dell-Inspiron ~/Java $
```

16. A program using simple Inheritance.

Code:

```
class Base{
     int x = 940;

     void display(){
          System.out.println("From Base class !");
     }
}

class Child extends Base{
     void show(){
          System.out.println("From Child clas !!");
     }
     void value(){
          System.out.println("Value of x : " + x);

     }
}

class Prog9{
     public static void main(String[] args){
          Child ob = new Child();

          ob.show();
          ob.display();
          ob.value();
     }
}
```

Output:



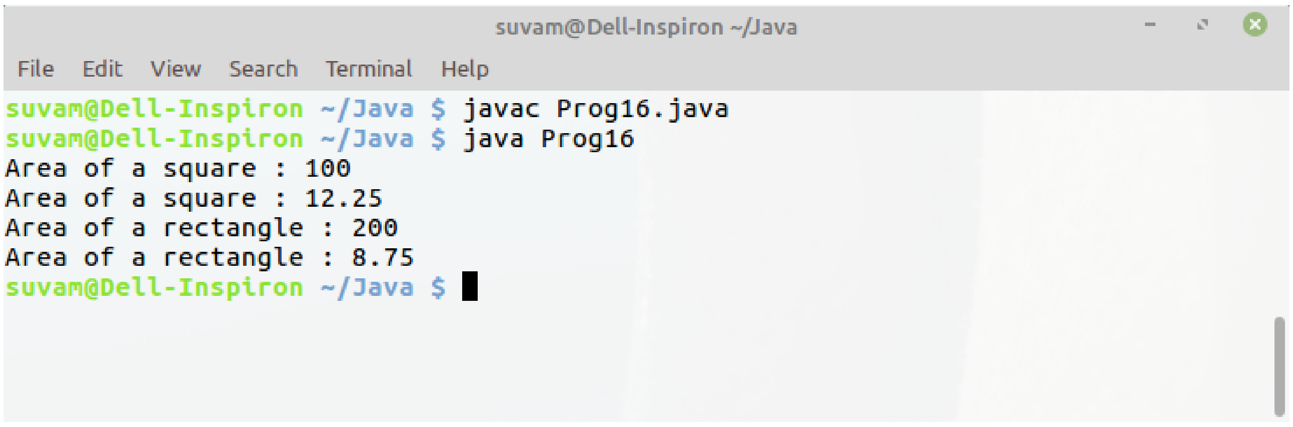
```
suvam@Dell-Inspiron ~/Java $ javac Prog9.java
suvam@Dell-Inspiron ~/Java $ java Prog9
From Child clas !!
From Base class !
Value of x : 940
suvam@Dell-Inspiron ~/Java $
```

17. A program for area calculation using Inheritance.

Code:

```
class Area{
	int getArea(int x){
		return (x*x);
	}
	double getArea(double x){
		return (x*x);
	}
}
class AreaAll extends Area{
	int getArea(int x, int y){
		return (x*y);
	}
	double getArea(double x,double y){
		return (x*y);
	}
}

class Prog16{
	public static void main(String[] args){
		AreaAll ob = new AreaAll();
		System.out.println("Area of a square : " + 
ob.getArea(10));
		System.out.println("Area of a square : " + 
ob.getArea(3.5));
		System.out.println("Area of a rectangle : " + 
ob.getArea(10,20));
		System.out.println("Area of a rectangle : " + 
ob.getArea(2.5,3.5));
	}
}
```

Output:


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

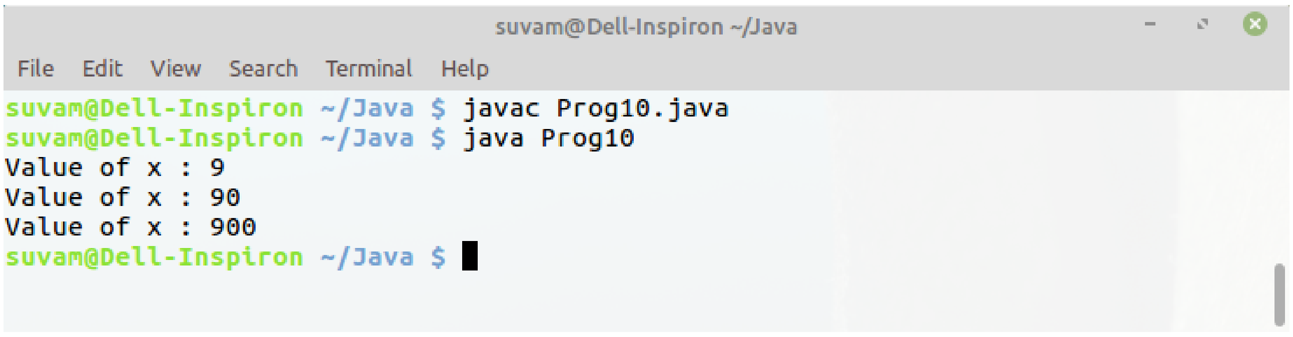
```
suvam@Dell-Inspiron ~/Java $ javac Prog16.java
suvam@Dell-Inspiron ~/Java $ java Prog16
Area of a square : 100
Area of a square : 12.25
Area of a rectangle : 200
Area of a rectangle : 8.75
suvam@Dell-Inspiron ~/Java $ 
```

18. Use of ‘this’ & ‘super’ keyword.

Code:

```
class Base{
     int x = 900;
}

class Child extends Base{
     int x = 90;

     void values(){
          int x = 9;
          System.out.println("Value of x : " + x);
          System.out.println("Value of x : " + this.x);
          System.out.println("Value of x : " + super.x);
     }
}

class Prog10{
     public static void main(String[] args){
          Child ob = new Child();

          ob.values();
     }
}
```

Output:


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

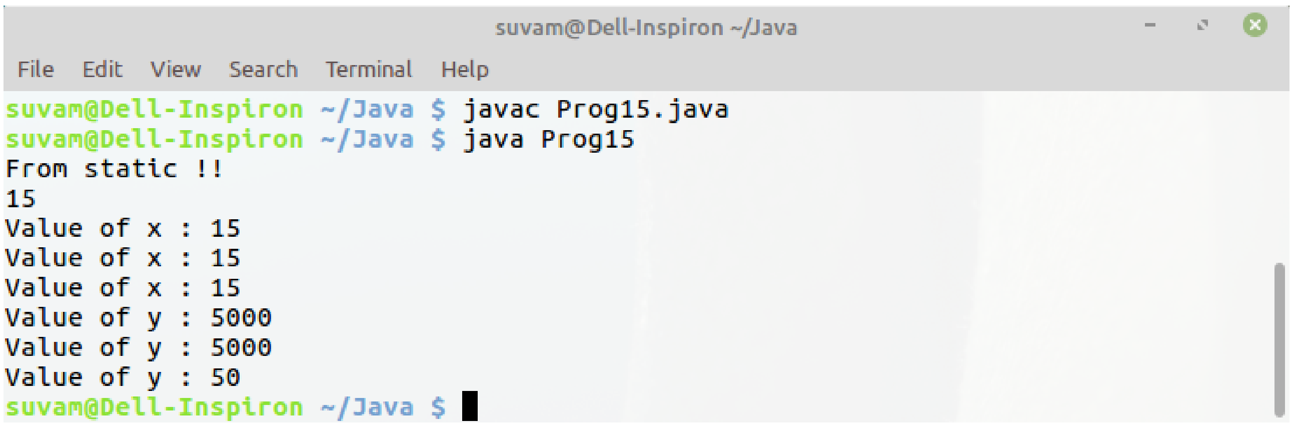
```
suvam@Dell-Inspiron ~/Java $ javac Prog10.java
suvam@Dell-Inspiron ~/Java $ java Prog10
Value of x : 9
Value of x : 90
Value of x : 900
suvam@Dell-Inspiron ~/Java $ 
```

19. Understanding the function of 'this', 'super' keyword & static method and static variable in java.

Code:

```
class Base{
	static int x = 15;
	int y = 50;
	static void show(){
		System.out.println("From static !! ");
	}
}

class Child extends Base{
	int y = 5000;
	void disply(){
		System.out.println("Value of x : " + x);
		System.out.println("Value of x : " + this.x);
		System.out.println("Value of x : " + super.x);

		System.out.println("Value of y : " + y);
		System.out.println("Value of y : " + this.y);
		System.out.println("Value of y : " + super.y);
	}
}

class Prog15{
	public static void main(String[] args){
		Base.show();
		System.out.println(Base.x);
		Child ob = new Child();
		ob.disply();
	}
}
```

Output:


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

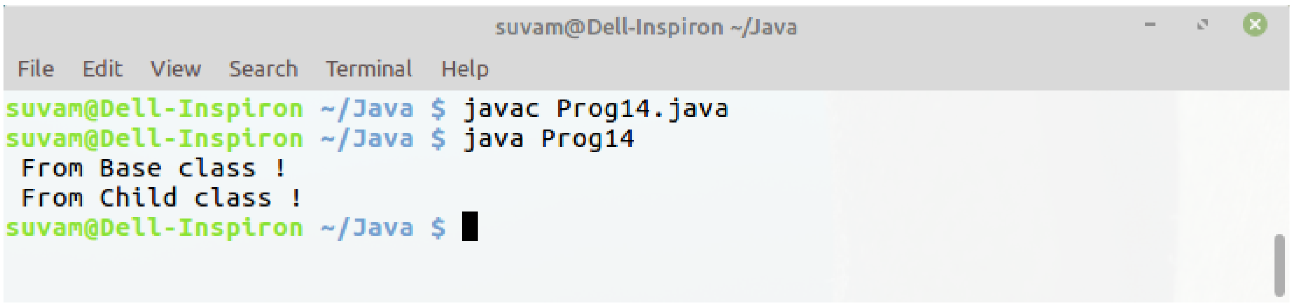
```
suvam@Dell-Inspiron ~/Java $ javac Prog15.java
suvam@Dell-Inspiron ~/Java $ java Prog15
From static !!
15
Value of x : 15
Value of x : 15
Value of x : 15
Value of y : 5000
Value of y : 5000
Value of y : 50
suvam@Dell-Inspiron ~/Java $
```

20. Method overriding & Dynamic Method Dispatch.

Code:

```
class Base{
	void show(){
		System.out.println(" From Base class !");
	}
}

class Child extends Base{
	void show(){
		System.out.println(" From Child class !");
	}
}

class Prog14{
	public static void main(String[] args){
		Base ob = new Base();
		ob.show();

		ob = new Child();
		ob.show();
	}
}
```

Output:


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

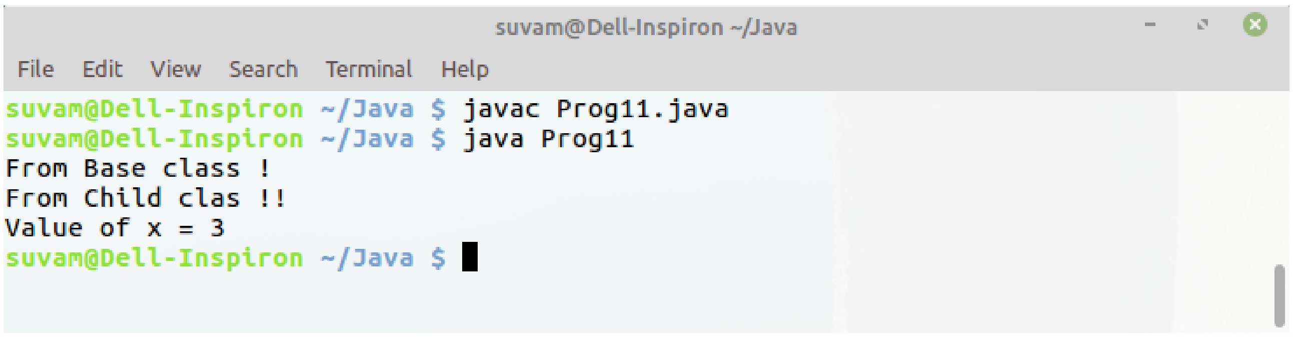
```
suvam@Dell-Inspiron ~/Java $ javac Prog14.java
suvam@Dell-Inspiron ~/Java $ java Prog14
 From Base class !
 From Child class !
suvam@Dell-Inspiron ~/Java $ 
```

21. Another example of Method overriding where child class constructor is parameterized.

Code:

```
class Base{
	Base(){
		System.out.println("From Base class !");
	}
	void show(){
		System.out.println("Nothing to show !");
	}
}

class Child extends Base{
	int x;
	Child(int a){
		x = a;
		System.out.println("From Child clas !!");
	}

	void show(){
		System.out.println("Value of x = " + x);
	}
}

class Prog11{
	public static void main(String[] args){
		Child ob = new Child(3);
		ob.show();
	}
}
```

Output:


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

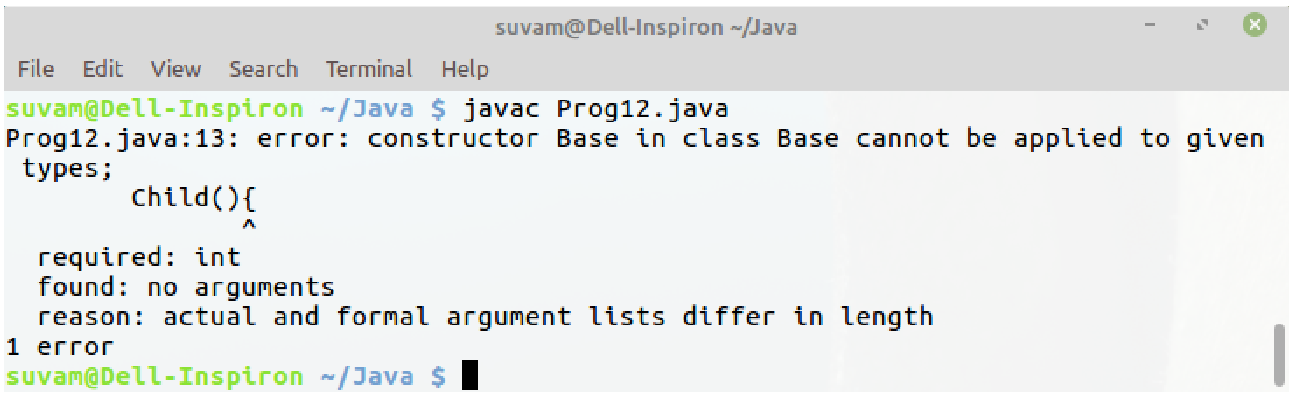
```
suvam@Dell-Inspiron ~/Java $ javac Prog11.java
suvam@Dell-Inspiron ~/Java $ java Prog11
From Base class !
From Child clas !!
Value of x = 3
suvam@Dell-Inspiron ~/Java $
```

22. Error in Method overriding where Base class Constructor is parameterized but Child class Constructor is not parameterized.

Code:

```
class Base{
     int c;
     Base(int a){
          c = a;
          System.out.println(" From Base class !");
     }
     void show(){
          System.out.println("value of c = "+c);
     }
}
class Child extends Base{
     int x = 10;
     Child(){
          System.out.println(" From Child class !");
     }
     void show(){
          System.out.println("value of x = "+x);
     }
}
class Prog12{
     public static void main(String[] args){
          Child ob = new Child();

          ob.show();
     }
}
```

Output:


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

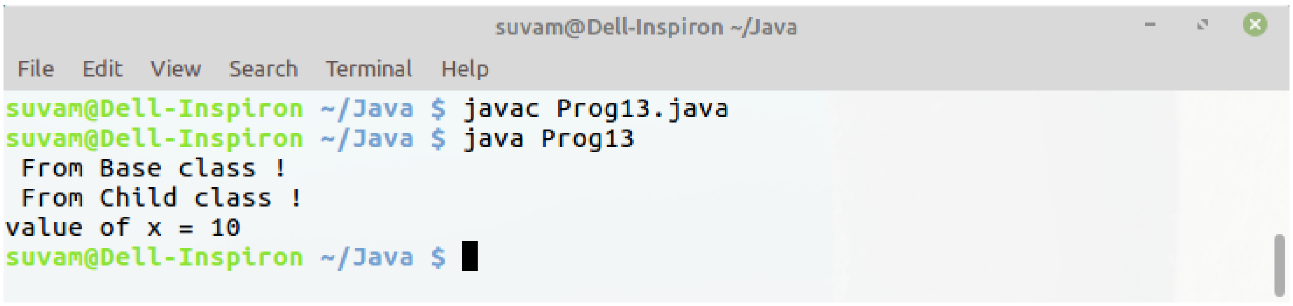
```
suvam@Dell-Inspiron ~/Java $ javac Prog12.java
Prog12.java:13: error: constructor Base in class Base cannot be applied to given
 types;
        Child(){
              ^
  required: int
  found: no arguments
  reason: actual and formal argument lists differ in length
1 error
suvam@Dell-Inspiron ~/Java $
```

23. Another use of ‘super’ keyword when Base class constructor is parameterized but Child class constructor is not parameterized.

Code:

```
class Base{
	Base(int a){
		System.out.println(" From Base class !");
	}
	void show(){
		System.out.println("Nothing !");
	}
}

class Child extends Base{
	int x =10;

	Child(){
		super(20);
		System.out.println(" From Child class !");
	}

	void show(){
		System.out.println("value of x = "+x);
	}
}

class Prog13{
	public static void main(String[] args){

		Child ob = new Child();
		ob.show();
	}
}
```

Output:


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

```
suvam@Dell-Inspiron ~/Java $ javac Prog13.java
suvam@Dell-Inspiron ~/Java $ java Prog13
 From Base class !
 From Child class !
value of x = 10
suvam@Dell-Inspiron ~/Java $ 
```