Var-args:

--variable number of argumetns.

--introduced in JAva 1.5 v

--this concept is developed based on array concept only but in
much more simplified way.

example:

```
package com.masai;

public class Demo {

	public static int add(int... arr) {
		int result=0;
		for(int i:arr) {
			result+=i;
		}
		return result;
	}

	public static void main(String[] args) {

		int[] nums= {10,20,30,40};

		System.out.println(add(10,20,30,40));
		System.out.println(add());

		//System.out.println(add(null));

		System.out.println(add(nums));
	}
}
```

Note: we can access var-args elements using zero based index, simillar to
normal array.

rules of using var-args:
---
1. if we want to declare any other parameters along with the var-args
then this
var-arg parameter must be the last parameter.
example:

```
package com.masai;
public class Demo {

	public static int add(String name, int... arr) {
		int result=0;
		for(int i:arr) {
```

```
                result+=i;
            }
            return result;
      }

      public static void main(String[] args) {
            System.out.println(add("Hello"));
            System.out.println(add("Hello",10));
            System.out.println(add("Hello",20,30));
            System.out.println(add("Hello",null));
      }
}
```

2. inside a single method we can not have 2 var-args as a parameter.even for
different types also.

3. if multiple overloaded methods are there then var-args gets the least priority.
example:

```
package com.masai;
public class Demo {
      public static int add(int... arr) {
            int result=0;
            for(int i:arr) {
                  result+=i;
            }
            return result;
      }

      public static int add(int i) {
            System.out.println("inside add(int i)");
            return 0;
      }

      public static void main(String[] args) {
            System.out.println(add(10));// second add metthod will be
called
      }

}
```

4. along with var-args we can not define another overloaded method which takes array of the same type parameter. it will become duplicate method defination.

example2:

```
package com.masai;
public class Demo {
      public static void fun1(Student... students) {
            for(Student s:students) {

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

```
            }


      }


      public static void main(String[] args) {


            Student[] stduetns= {

                        new Student(10, "n1", 780),
                        new Student(12, "n2", 780),
                        new Student(13, "n3", 780),
                        new Student(14, "n4", 780),



            };


            fun1(stduetns);


      }

}
```

Enum:
======

--with the help of enum we can create our own datatype (enumerated data type)

--in java we have 8 primitive data types are their.

boolean b= true/ false

byte b = -128 to 127 = 256 // here 256 values are allowed.

--if we want to create a datatype where we allow only certain set of values
then we should use enum.

enum of Month :

Month.java

```
package com.masai;

public enum Month {

      JAN,FEB,MAR,APR,MAY;


}
```

--enum can be created in 2 ways:

1.as a seperate .java file like a class or interface (we can compile this enum and can generate a .class file also)

2.as inner enum inside a class(like a inner class.)

--enum contants should be in upper case (naming convension)

Demo.java:

```
package com.masai;

public class Demo {

      public static void main(String[] args) {

                  Month m=Month.JAN;

                 System.out.println(m);


      }

}
```

---enum concept is introduced in java 1.5v

--every enum contant is implemented by using class concept.

--every enum constant is always "public static final".

--every enum constant represented an object of type enum.

--the above enum Month will converted internally as follows:

```
public final class Month extends Enum{

public static final JAN= new Month();
public static final FEB= new Month();
public static final MAR= new Month();
public static final APR= new Month();
public static final MAY= new Month();

}
```

java.lang.Enum class is a predefiend an abstract class.
--this Enum class already implements Comparable and Serializable interface internally.

Note: inheritance concept is not applicable with an enum.
but an enum can implements any number of interfaces.

--inlike other languages enum in java is more powerfull because, we can have variables, methods, constructors inside an enum also.

--even we can place main method inside an enum and execute that as similar to class.

example:

```
package com.masai;


public enum Month  {

      JAN,FEB,MAR,APR,MAY;

      public static void main(String[] args) {

            System.out.println("inside main of enum");

      }


}
```

values() and ordinal() method:
==========

--Every enum implicitly contains values() method that returns all the values presents
inside an enum.

--values() method will return all the values in the form of array .
this value() method is a static method which we can call on any enum.

--withing the enum every constant placed based on the order, and we can find the Ordinal values of enum constant by uisng ordinal() method.

--this ordinal values are zero based index value.

--ordinal() method is non-static method.

Demo.java:

```
package com.masai;

public class Demo {

	public static void main(String[] args) {

			Month[] months= Month.values();

			for(Month month:months) {

	System.out.println(month+"==========="+month.ordinal());

			}

	}

}
```

methods and variables inside an enum:
==

```
package com.masai;


public enum Month  {

	JAN,FEB,MAR,APR,MAY; // here if we place any other elements then
				//this semicolon is mandatory otherwise it is
optional

	int i=10;
```

```
	void fun1() {

		System.out.println("inside fun1");
	}


}
```

constructor in enum:

--we can place constructor also inside an enum, that constructor can only be
private whether we mension it or not.

--a constructor of an enum will be executed seperatly for every enum constant at the
time of enum class loaded into the memory.

Month.java:

```
package com.masai;


public enum Month  {

	JAN,FEB,MAR,APR,MAY;

	private Month() {
		System.out.println("isnide constructor of Month");
	}


}
```

Demo.java:

```
package com.masai;

public class Demo {

	public static void main(String[] args) {

			Month m= Month.JAN;


	}
```

```
}
```

output:

```
isnide constructor of Month
isnide constructor of Month
isnide constructor of Month
isnide constructor of Month
isnide constructor of Month
```

Example :

Month.java:

```
package com.masai;


public enum Month  {

	JAN,FEB(28),MAR(31),APR,MAY;

	private Month() {
		System.out.println("isnide constructor of Month");
	}


	private Month(int num) {
		System.out.println("isnide constructor of Month(int num)");
	}

}
```

--here for FEB and MAR, paramterized constructor and for remaining zero argument constructor will be executed.

Item.java:

```
package com.masai;

public enum Item {

	SUGER,SALT,PENCIL(5),PEN(10),SHARPNER(30);

	int price;

	private Item() {
		this.price=20;
	}
```

```
	private Item(int price) {
		this.price=price;
	}


	public int getPrice() {

		return this.price;
	}



}
```

Demo.java:

```
package com.masai;

public class Demo {

	public static void main(String[] args) {

		Item[] items= Item.values();

		for(Item item:items) {


			System.out.println(item+" price is : "+item.getPrice());
		}
	}

}
```

exmaple 2:

City.java:

```
package com.masai;

public enum City {

	DELHI{

		@Override
		public void message() {
			System.out.println("Welcome User you are in Capital");
```

```
                System.out.println("Tower in your area is
:"+numberOfTowers);


            }




      },MUMBAI,KOLKATA,CHENNAI("50 towers");


      String numberOfTowers;


      private City() {
            this.numberOfTowers="100 towers";
      }

      private City(String numberOfTowers) {
            this.numberOfTowers=numberOfTowers;
      }


      public void message() {
            System.out.println("Welcome User");
            System.out.println("Tower in your area is :"+numberOfTowers);

      }

}


Demo.java:
--------------

package com.masai;

import java.util.Scanner;

public class Demo {

      public void printCity(City city) {

            if(city != null) {


                  city.message();
            }
            else
                  System.out.println("invalid city");

      }
```

```
	public static void main(String[] args) {

		Demo d1= new Demo();

		Scanner sc=new Scanner(System.in);

		System.out.println("Enter the City name");
		String cityName= sc.next();

		//converting String to the appropriate enum type.

		City city= City.valueOf(cityName.toUpperCase());
		//for any invalid city , it will throw an exception

		d1.printCity(city);



	}

}
```