1.   Write a java program to display factors of a given number?

sol:

```
import java.util.Scanner;
public class Test1{
        public static void main(String[] args) {

                System.out.println("Enter any number: ");
                Scanner sc=new Scanner(System.in);
                int n;
                n=sc.nextInt();


                for(int i=1;i<=n;i++){
                        if(n%i==0) {
                                System.out.println(i+" ");
                        }
                }
        }
}
```

output
======

```
Enter any number:
15
1
3
5
15
```

2.   Write a java program to display the numbers in the Collatz series for any number?

sol:
----
3.   Write a java program to display the Fibonacci series of a given number?

sol:

```
class Test1{
        public static void main(String[] args) {
                int a=0;
                int b=1;
                for(int i=0;i<10;i++) {

                        int temp;
                        temp=a+i;
                        b=i+temp;
                        System.out.println(temp);
                }
        }
}
```

4.   Write a java program to display below loop pattern?

sol:

```
8
16   17
32   33   34   35
64       65   66   67   68   69   70   71
```

sol:
----

5.   Write a java program to display below loop pattern?

```
              2
             3     5
            7      11      13
17        19       23       29
```

sol:

```
class Test1{
        public static void main(String[] args) {
                for(int i=1;i<30;i++) {
                        int count =0;
                        for(int j=1;j<=i;j++) {
                                if(i%j==0) {
                                        count++;
                                }
                        }
                        if(count==2) {
                                System.out.print(i+" ");
                        }
                }

        }
}
```

6.   Write a program that prints the numbers from 1 to 100. But for multiples of three, print “Fizz” instead of the number, and for the multiples of five, print “Buzz”. For numbers that are multiples of both three and five, print “FizzBuzz”.

sol:

```
package mypack;

public class Test1 {
	public static void main(String [] args) {
		for(int i=1;i<=100;i++) {

			if(i%3==0&&i%5==0) {
				System.out.print("FizzBuzz ");
			}
			else if(i%3==0) {
				System.out.println("Fizz");
			}
			else if(i%5==0) {
				System.out.println("Buzz ");
			}
			else {
				System.out.print(i+" ");
			}
		}
	}

}
```

output
======

```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz
Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32 Fizz 34 Buzz Fizz 37 38 Fizz
Buzz 41 Fizz 43 44 FizzBuzz 46 47 Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56 Fizz 58
59 FizzBuzz 61 62 Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 FizzBuzz 76 77
Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 92 Fizz 94 Buzz Fizz
97 98 Fizz Buzz
```

7. Write a Java program to create an abstract class BankAccount with abstract methods deposit() and withdraw(). Create subclasses: SavingsAccount and CurrentAccount that extend the BankAccount class and implement the respective methods to handle deposits and withdrawals for each account type.

sol:

```
abstract class BankAccount{
	abstract void deposit();
	abstract void withdraw();
}
class SavingsAccount extends BankAccount{
	String accounttype;
	double deposite;
	double withdraw;


	public SavingsAccount(String accounttype, double deposite, double
withdraw) {
		super();
		this.accounttype = accounttype;
		this.deposite = deposite;
		this.withdraw = withdraw;
	}

	void deposit() {
		System.out.println("Account type : "+accounttype);
		System.out.println("deposite :"+deposite);

	}

	void withdraw() {
		System.out.println("withdraw :"+withdraw);
	}


	public String toString() {
		return "SavingsAccount [accounttype=" + accounttype + ",
deposite=" + deposite + ", withdraw=" + withdraw + "]";
	}

}
class CurrentAccount extends BankAccount{
	String accounttype;
	double deposite;
	double withdraw;

	public CurrentAccount(String accounttype, double deposite, double
withdraw) {
		super();
		this.accounttype = accounttype;
		this.deposite = deposite;
		this.withdraw = withdraw;
	}

	void deposit() {
		System.out.println("Account type : "+accounttype);
		System.out.println("deposite :"+deposite);

	}

	void withdraw() {
		System.out.println("withdraw :"+withdraw);

	}
	public String toString() {
		return "SavingsAccount [accounttype=" + accounttype + ",
deposite=" + deposite + ", withdraw=" + withdraw + "]";
	}

}
public class Test1 {
	public static void main(String [] args) {
```

```
                SavingsAccount sa=new SavingsAccount("savings", 10000,1000);
                sa.deposit();
                sa.withdraw();
                System.out.println();

                CurrentAccount ca=new CurrentAccount("current", 9000,2000);
                ca.deposit();
                ca.withdraw();
        }
}
```

```
output
======
Account type : savings
deposite :10000.0
withdraw :1000.0

Account type : current
deposite :9000.0
withdraw :2000.0
```

8. Write a Java program to create a class Employee with a method called calculateSalary(). Create two subclasses Manager and Programmer. In each subclass, override the calculateSalary() method to calculate and return the salary based on their specific roles.

sol:

```
class Employee{
        String salary;
        String domain;

        public Employee(String salary, String domain) {
                super();
                this.salary = salary;
                this.domain = domain;
        }

        void calculateSalary() {
                System.out.println("Salary :"+salary);
                System.out.println("Domain :"+domain);
        }

        @Override
        public String toString() {
                return "Employee [salary=" + salary + ", domain=" + domain +
"]";
        }

}
class Manager extends Employee{
```

```java
        String salary;
        String domain;

        public Manager(String salary, String domain, String salary2, String
domain2) {
                super(salary, domain);
                this.salary = salary2;
                this.domain = domain2;
        }

        void calculateSalary() {
                System.out.println("Salary :"+salary);
                System.out.println("Domain :"+domain);
        }

        @Override
        public String toString() {
                return "Employee [salary=" + salary + ", domain=" + domain +
"]";
        }
}
class Programmer extends Employee{


        String salary;
        String domain;


        public Programmer(String salary, String domain, String salary2, String
domain2) {
                super(salary, domain);
                this.salary = salary2;
                this.domain = domain2;
        }

        void calculateSalary() {
                System.out.println("Salary :"+salary);
                System.out.println("Domain :"+domain);
        }

        @Override
        public String toString() {
                return "Employee [salary=" + salary + ", domain=" + domain +
"]";
        }
}
class Test1{
        public static void main(String [] args) {
                Manager m=new Manager("90000","java","100000","python");
                System.out.println("Manager");
                System.out.println("========");
                m.calculateSalary();
                System.out.println();
                Programmer p=new Programmer("1000","python","19000","java");
                System.out.println("Programmer");
                System.out.println("===========");
                p.calculateSalary();
        }
}
```

```
output
======
Manager
========
Salary :100000
Domain :python

Programmer
===========
Salary :19000
Domain :java
```

9.   Implement a functional interface called NumberPredicate to check if a number satisfies certain conditions. Use lambda expressions for checking if a number is positive, negative, and even.

sol:

```
interface NumberPredicate {
    boolean test(int number);
}
public class Test1 {
    public static void main(String[] args) {

        NumberPredicate Positive = number -> number > 0;

        NumberPredicate Negative = number -> number < 0;

        NumberPredicate Even = number -> number % 2 == 0;


        int number =10;
            System.out.println("Number: " + number);
            System.out.println("Number is Positive :" + Positive.test(number));
            System.out.println("Number is Negative :" + Negative.test(number));
            System.out.println("Number is Even     :"     + Even.test(number));
    }
}
```

```
output:
=======
Number: 10
Number is Positive :true
Number is Negative :false
Number is Even     :true
```

10.   Write a Java program to create an interface Flyable with a method called fly_obj(). Create three classes Spacecraft, Airplane, and Helicopter that implement the Flyable interface. Implement the fly_obj() method for each of the three classes.

sol:

```
interface Flyable{
        void fly_obj();
}
class Spacecraft implements Flyable{

        public void fly_obj() {
                System.out.println("This is Spacecraft");
        }
}
class Airplane implements Flyable{
        public void fly_obj() {
                System.out.println("This is Airplane");
        }
}
class Helicopter implements Flyable{
        public void fly_obj() {
                System.out.println("This is Helicopter");
        }
}

public class Test1{
        public static void main(String[] args) {
                Spacecraft s=new Spacecraft();
                s.fly_obj();
                System.out.println();
                Airplane a=new Airplane();
                a.fly_obj();
                System.out.println();
                Helicopter h=new Helicopter();
                h.fly_obj();
        }
}
```

output
=======

```
This is Spacecraft

This is Airplane

This is Helicopter
```